User's Manual

Renesas Synergy™ Software Package（SSP）v1.5.0ユーザーズマニュアル（参考資料）

Renesas Synergy™ プラットフォーム

本資料は英語版 Rev.1.10 の第 1 章から第 5 章を日本語に翻訳した参考資料です。第 6 章の API リファレンス以降は英語版 Rev.1.10 をご参照ください。日本語版と英語版で内容に相違がある場合は英語版を優先します。資料によっては英語版のバージョンが更新され、内容が変わっている場合があります。日本語版は参考用としてご使用のうえ、最新および正式な内容については英語版のドキュメントをご参照ください。

資料番号 R11UM0091EU0110、リビジョン Rev.1.10、発行日 2018 年 11 月 15 日の翻訳版です。

Renesas Electronics
www.renesas.com

Rev.1.10　2019.03

第 1 章　Renesas Synergy Software Package (SSP)

1.1 Renesas Synergy™ ソフトウェアパッケージの概要

1.1.1 SSP ユーザーズマニュアルの概要

このマニュアルでは、Renesas Synergy Software Package を使用して、Synergy マイクロコントローラシリーズ対応のアプリケーションを作成する方法について説明します。下の図では、SSP ユーザーズマニュアルの API リファレンスコンポーネントが青で示されています。このマニュアルではまた、SSP でのプログラミング手順を理解するため、e^2 studio ISDE の説明やチュートリアルと例など、その他のコンポーネントについても記載されています。

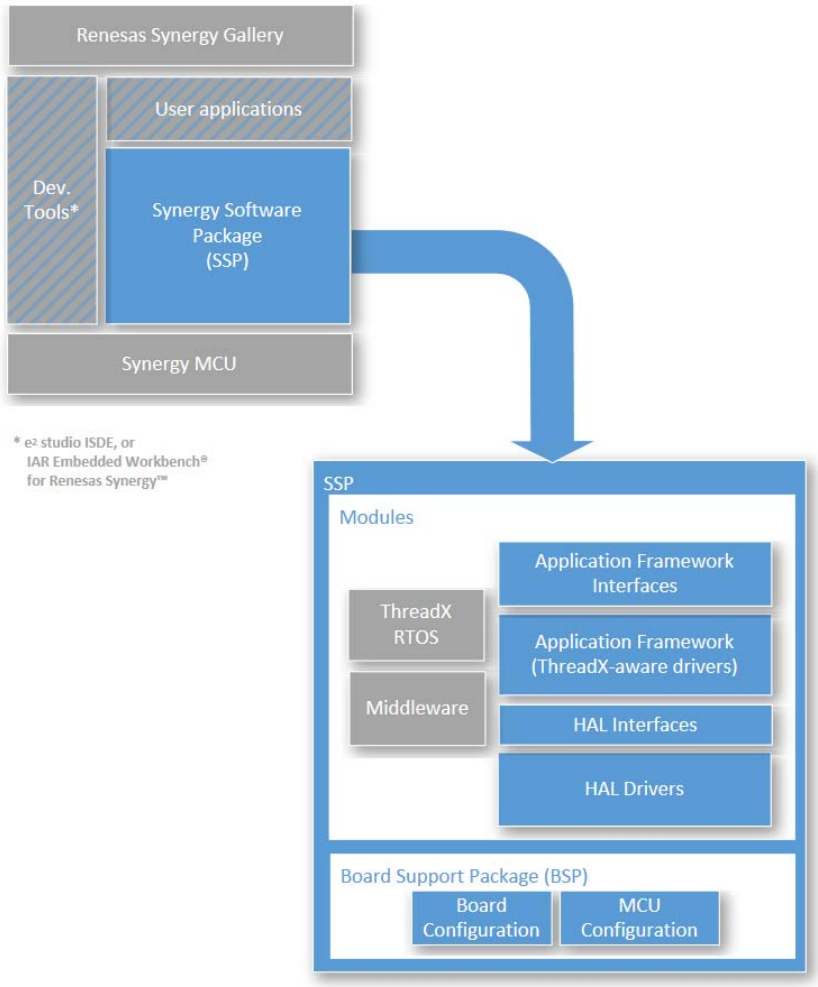


図 1: Synergy Software Package（SSP）のドキュメント

1.1.2 このマニュアルに含まれる内容

SSP アーキテクチャおよび SSP でのボードとチップレベルのサポートについては、以下を参照してください。

- SSP アーキテクチャ
- BSP アーキテクチャ

SSP を使用したプログラミングおよび e^2 studio ISDE の概要説明については、以下を参照してください。

e^2 studio ISDE ユーザーガイド

入門レベルのチュートリアルとアプリケーション例については、以下を参照してください。

- チュートリアル：Your First Synergy Project - Blinky

- チュートリアル：Using HAL Drivers - Programming the WDT

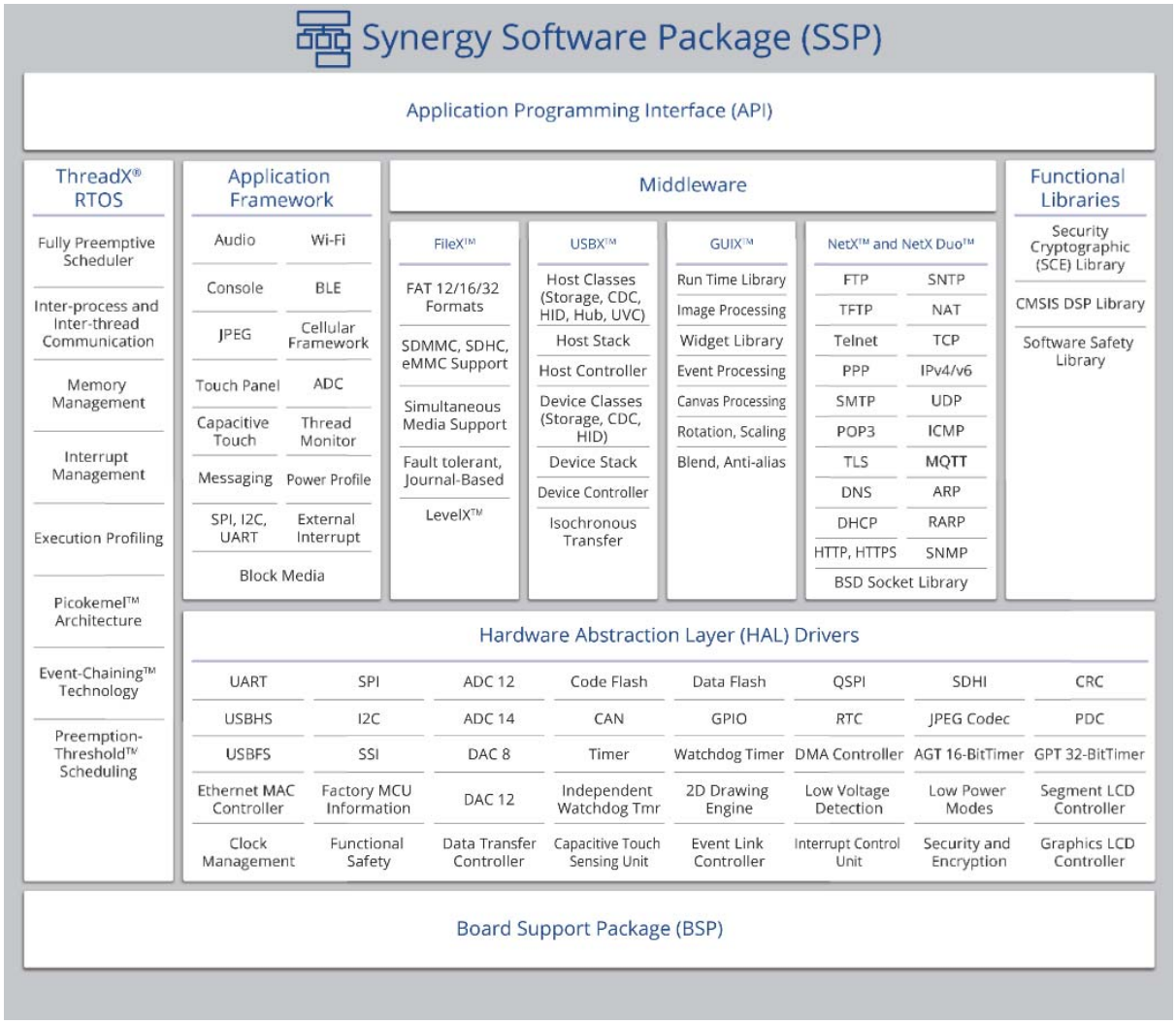
「モジュールの概要」では SSP モジュールについて説明します。以下を参照してください。

- フレームワークレイヤー
- HAL レイヤー
- Express Logic 社のモジュール

このドキュメントには、以下の SSP コンポーネントの API リファレンスドキュメントが含まれています。

- Framework Interfaces：ThreadX 対応のフレームワークモジュールへのインタフェース
- Framework Layer：ThreadX 対応のフレームワークドライバーモジュール
- HAL Interfaces：ハードウェア抽象化レイヤー（HAL）モジュールへのインタフェース
- HAL Layer：RTOS 独立の HAL ドライバーモジュール
- Board Support Package：ボード固有、かつマイクロコントローラ固有の設定モジュールが含まれたボードサポートパッケージ（BSP）

第 2 章　SSP の概要



Renesas Synergy™ プラットフォームの一部である Renesas Synergy™ Software Package（SSP）は、使いやすく拡張性に優れた高品質の組み込みシステム設計ソフトウェアを提供するための完全な統合ソフトウェアパッケージです。

SSP アーキテクチャについては以下を参照してください。

- SSP アーキテクチャ
- BSP アーキテクチャ

2.1 はじめに

2.1.1 目的

Renesas Synergy™ プラットフォームの一部である Renesas Synergy™ Software Package（SSP）は、使いやすく拡張性に優れた高品質の組み込みシステム設計ソフトウェアを提供するための完全な統合ソフトウェアパッケージです。Synergy ソフトウェアプラットフォームを使用することで、完全に統合された認証済みの組み込みソフトウェアプラットフォームが提供する、完全に最適化され、強化された業界屈指のリアルタイムオペレーティングシステム（RTOS）、ミドルウェア、通信スタック、機能ライブラリ、アプリケーションフレームワーク、ハードウェア抽象化された低レベルデバイスドライバーを利用でき、迅速な商品化が可能になります。

2.1.2 概要

SSP は、以下の 4 つの主要レイヤーで構成されています。

- Framework Interfaces : フレームワークライブラリは、ハードウェアを機能のユースケースとして抽象化する共通インタフェースを介して Synergy ハードウェアペリフェラルに接続します。「インタフェースレイヤー」は、関数とパラメータの定義が入ったヘッダーファイルのグループであり、コード用のスペースは含まれていません。
- Framework Layer :「フレームワークレイヤー」は、RTOS 統合されたドライバーおよび有益なアプリケーションコードで構成されています。
- HAL Interfaces:「HAL レイヤー」インタフェースは、RTOS独立のHAL レベルのドライバーに接続します。
- HAL Layer : HAL レイヤーのドライバーとハードウェアレジスタによって、インタフェースが実装されます。

2.1.3 使いやすさ

SSP は、統一的で直観的な API とそれに関する充実したドキュメントを提供しています。各モジュールには詳細なユーザードキュメントに加え、各関数のコードサイズや実行時間などを記載したソフトウェアデータシートが用意されています。

2.1.4 スケーラビリティ

ユーザーは、フレームワークレイヤー、インタフェースレイヤー、HAL レイヤーの中から、アプリケーションのニーズに最も適したレイヤーを使用してプラットフォーム機能と統合することができます。各モジュールをさらに拡張するには、パラメータチェッキングなどのビルド時オプションをコンパイルアウトすることで、小さく効率的なコードを作成できます。

2.2 SSP アーキテクチャ

2.2.1 SSP アーキテクチャ

このセクションでは、Renesas Synergy ソフトウェアパッケージ（SSP）アーキテクチャと SSP アプリケーションプログラミングインタフェース（API）の使用方法について説明します。

2.2.1.1 SSP の概要

マイクロコントローラの複雑さが増すにつれ、幅広い知識がないと想定どおりに動作させることが難しくなっています。SSP は、そのようななかで IoT アプリケーション用の組み込みソフトウェアの開発に革新をもたらすものです。SSP を使用すると、まったく新しい強力なソフトウェアインタフェースにより、コーディングにかかる時間を短縮しつつ、確実な開発プロセスを実現することができます。このソフトウェアでは、数か月を費やしてベースとなるコードを開発するのではなく、個別のアプリケーションコードを作成することによって、ハードウェアレベルのインタフェースを実装できます。

2.2.1.2 SSP 用語

Term	説明	Reference
Module	モジュールは、ペリフェラルドライバーから単なるソフトウェアまでのすべてに該当します。各モジュールは、ソースコード、ドキュメント、およびユーザーがコードを効率的に使用するために必要な要素を含む 1 つのフォルダで構成されます。モジュールは独立した単位ですが、他のモジュールに依存する場合があります。SSP モジュールの例として、UART ドライバー (UART Interface)、オーディオ再生フレームワーク（タイマ、DMA、DAC ドライバー (Audio Framework Interface) に依存する）、およびメッセージフレームワーク (Messaging Framework Interface) が挙げられます。複数のモジュールを組み合わせてユーザーに必要な機能を提供することで、アプリケーションを構築できます。	SSP モジュール
BSP	Board Support Package の略語。 SSP では、他の SSP モジュールが問題なく連動するために最低限必要な基礎を BSP によって提供します。	BSP アーキテクチャ

Term	説明	Reference
Callback Function	この用語はイベント発生時に呼び出される関数を意味します。たとえば、バスエラー割り込みハンドラは、r_bsp. 内に実装されています。ユーザーがバスエラーの発生を知りたいと思うことはよくあります。そのようなときは、r_bsp. にコールバック関数を提供すると、ユーザーに通知を送ることができます。実際にバスエラーが発生したときは、r_bsp が提供されたコールバック関数にジャンプして、ユーザーがそのエラーを処理できます。割り込みコールバック関数は長くならないように慎重に扱う必要があります。なぜなら、この関数が呼び出されると、現在の割り込みが中断されて MCU がまた別の割り込みに入ることになるため、保留中の割り込みがすべて後回しにされるからです。	-
Interface	「SSP インタフェース」セクションを参照してください（SSP インタフェース）。 SSP 内のすべてのインタフェースは、 API Reference: Framework Interfaces と API Reference: HAL Interfaces に記載されています。	SSP インタフェース
Instance	「SSP インスタンス」セクションを参照してください（SSPインスタンス）。	SSP インスタンス
Module Instance	あるモジュールの単一で独立した構成。	-
Application	ユーザーが所有者として維持管理するコード。アプリケーションコードは、Renesas から提供されるサンプルアプリケーションコードをベースに作成することができますが、責任はユーザーにあります。	An example for a simple application is included as tutorial in this manual: チュートリアル : Using HAL Drivers - Programming the WDT
Driver	ドライバーは、MCU 上のレジスタを直接変更する特殊なモジュールです。	-

Term	説明	Reference
Stacks	SSP アーキテクチャは、モジュールが連動してスタックを形成するように設計されています。最上位のモジュールから最下位のモジュールまでの依存関係が特定のスタックを形成します。	SSP スタック
Layer/Level	スタックは、複数のモジュールのレイヤーで構成されます。各レイヤーは、1 つ上のレイヤーの要件に応じて 1 つまたは複数のモジュールで構成されます。レイヤーとレベルは区別なく使用されます。	SSP 事前定義レイヤー

2.2.2 SSP モジュール

モジュールは SSP の中核的構成要素です。モジュールを使用すれば、さまざまなことを実行できますが、どのモジュールでも、上位に機能を提供し下位に機能を要求するという基本概念は共通しています。

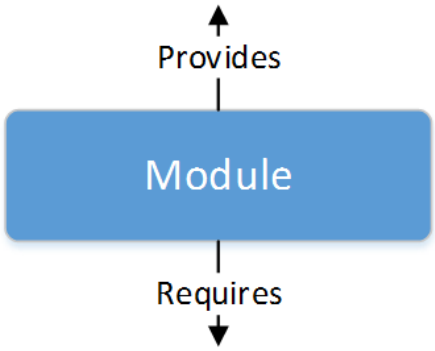


図 2: モジュール

モジュールによって提供される機能の量に制限はありませんが、通常は意味をなさなくなる限界点が存在します。提供される機能が多すぎると、将来、そのモジュールの再利用が困難になります。提供される機能が不十分だと、モジュールを想定どおりに機能させるための複雑さとオーバーヘッドが不必要に増加します。

最も単純な SSP アプリケーションは、最上位にユーザーアプリケーションを据えた 1 つのモジュールで構成されます。

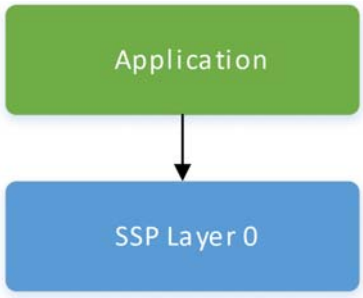


図 3: アプリケーションを含むモジュール

簡単にするために、現時点ではボードサポートパッケージ（BSP）を無視します。上の図では、BSP は最下位レイヤー（SSP Layer 0）の下に配置されています。

2.2.3 SSP スタック

モジュールが上下に重なると、SSP スタックが形成されます。このスタック処理は、あるモジュールが提供しているものと別のモジュールが必要としているものが一致したときに実行されます。たとえば、オーディオ再生フレームワークモジュールには転送インタフェースが必要ですが、それはデータトランスファコントローラ（DTC）ドライバーモジュールで実現できます。オーディオ再生モジュールに DTC コードが組み込まれる代わりに、それらが 2 つのモジュールに分割されています。これにより、基礎となるモジュールの再利用が可能になり、多くのメリットが得られます。

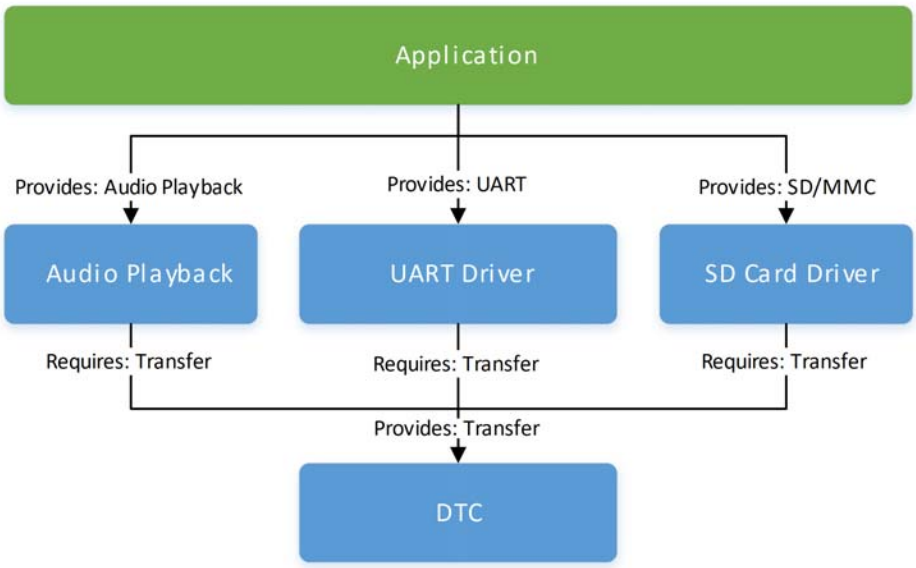


図 4: スタック – オーディオ再生

SSP モジュールを使ってレイヤーをスタックに追加し続けることにより、Synergy MCU と高レベルでやり取りすることができます。

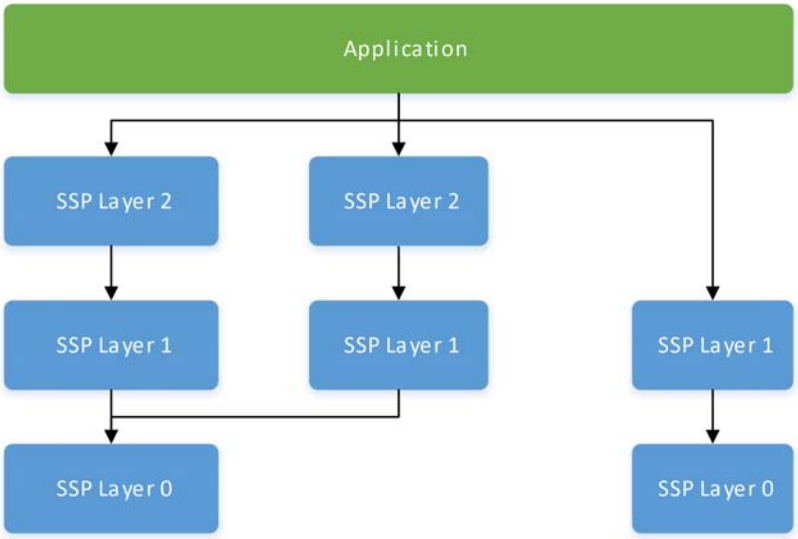


図 5: スタック

モジュールをスタックできると、アーキテクチャ全体の柔軟性が保証されるため、大きなメリットが得られます。モジュールが他のモジュールに直接依存している場合は、アプリケーション機能がさまざまなユーザー設計で機能できないときに問題が発生します。モジュールが再利用可能なことを保証するには、同じ機能を提供する他のモジュールに対してモジュールをスワップアウトできる必要があります。SSP アーキテクチャは、SSP インタフェースの使用を通じてモジュールをスワップインまたはスワップアウトできる柔軟性を提供しています。

2.2.4 SSP インタフェース

アーキテクチャレベルでは、モジュールはインタフェースを通じて共通機能を提供します。この共通性によって、同じインタフェースに接続されているモジュールを区別せずに使用できます。インタフェースは、2 つのモジュール間の契約と見なすことができます。モジュールは、契約内で同意された情報を使用して連動することに同意します。

Synergy MCU デバイスでは、別々のペリフェラルが重複していることがあります。たとえば、I2C 通信は IIC ペリフェラルまたは簡易 I2C モードの SCI ペリフェラルの使用を通して実現できます。両方のペリフェラルから提供される機能のレベルに違いがあります。I2C モードの SCI は、フル機能を備えた IIC の機能の一部しかサポートしません。

インタフェースは、ほとんどのユーザーが期待する共通機能のサポートを提供することを目的としています。つまり、IIC などの高度なペリフェラルのなかには、インタフェースで使用できないものがあります。このような機能のほとんどは、インタフェース拡張機能を介して使用することができます。

設計上、インタフェースはヘッダーファイルとして実装されます。すべてのインタフェースヘッダーファイル名は '_api.h'. で終わります。ここからは、インタフェースの構成要素について説明します。

2.2.4.1 SSP インタフェース列挙

可能な場合、インタフェースは関数パラメータと構造体メンバーに対して型指定列挙を使用します。以下に例を示します。

```
typedef enum e_i2c_addr_mode
{
  I2C_ADDR_MODE_7BIT = 1, ///< Use 7-bit addressing mode
  I2C_ADDR_MODE_10BIT ///<; Use 10-bit addressing mode
} i2c_addr_mode_t;
```

列挙を使用すると、パラメータで使用可能な値を決定する際の不確実性を取り除くことができます。また、列挙オプションは型の名前が使用可能なオプションに基づいてプレフィックスされる厳密な命名規則に従っていることにも注意してください。命名規則と e^2 studio で使用可能なオートコンプリート機能を組み合わせると、高度に読み取りやすいコードを維持しながら、コーディングが短時間で完成するメリットを得ることができます。

2.2.4.2 SSP インタフェースデータ構造体

少なくとも、すべての SSP インタフェースには、制御構造体、構成構造体、およびインスタンス構造体という 3 つのデータ構造体が含まれています。

制御構造体はモジュールを使用する場合の一意の識別子として使用されます。SSP モジュールが唯一のペリフェラルドライバーの場合は、この制御構造体がチャネル番号に置き換えられます。その場合は、モジュールに対するすべての関数呼び出しでチャネル番号が使用されるため、コードで動作対象のペリフェラルチャネルを特定できます。SSP モジュールはデバイスドライバーに制限されないため、制御構造体が使用されます。ユーザーは、制御構造体用のストレージを割り当ててから、その構造体へのポインタを open() 呼び出しに渡します。この時点で、モジュールが必要に応じて構造体を初期化します。その後、ユーザーがそれ以降のすべてのモジュール呼び出しに対して制御構造体へのポインタを送る必要があります。制御構造体の内容はモジュールによって使用されるものであるため、変更されないようにする必要があります。SSP リリース間でデータ構造が一定であるとは限らないため、制御構造体からのデータリードも避ける必要があります。一般的には、制御構造体をブラックボックスのように取り扱うようにします。

制御構造体の内容はインスタンスに固有です。つまり、インタフェースのインスタンスが 2 つある場合、その制御構造体の型はまったく異なるものになります。1 つのインタフェースに存在する制御構造体の名前は、<interface>_ctrl_t. です。次の例は I2C インタフェースの制御構造体の例です。

```
typedef void  i2c_ctrl_t;
```

インタフェース制御構造体の型はいずれも *void* なので、インタフェース制御構造体を割り当てることはできません。代わりに、これらの型はインスタンス制御構造体のプレースホルダーになります。インスタンス制御構造体はインスタンスヘッダーファイルで定義され、名前は <instance>_instance_ctrl_t. です。次の例は、IIC(r_riic) および SCI(r_sci_i2c) の I2C インスタンス制御構造体を示しています。

```
/** riic Instance control structure to be used with I2C Interface. */
typedef struct st_riic_instance_ctrl
```

```
{
  i2c_cfg_t info; ///< Information describing I2C device
  uint32_t open; ///< Flag to determine if the device is open
  void * p_reg; ///< Base register for this channel

/** More members specific to riic Instance. */

} riic_instance_ctrl_t;

/** sci_i2c Instance control structure to be used with I2C Interface. */

typedef struct st_sci_i2c_instance_ctrl
{
  i2c_cfg_t info; ///< Information describing I2C device
  uint32_t open; ///< Flag to determine if the device is open
  void * p_reg; ///< Base register for this channel

  /** More members specific to sci_i2c Instance. */

} sci_i2c_instance_ctrl_t;
```

インタフェースを使用する際は、インスタンス制御構造体を割り当て、インタフェース制御構造体の代わりに使用する必要があります。上の例を使って説明すると、SCI-I2C インスタンスを使用しているのであれば、型 sci_i2c_instance_ctrl_t の構造体を割り当て、インタフェースで i2c_ctrl_t が参照される場合には常にそちらを使用するようにします。統合ソリューション開発環境は、ユーザーに代わって正確な制御構造体の割り当てを行います。

malloc() 関数と free() 関数を使用した動的メモリ割り当ては SSP モジュールでは使用されません。

設定構造体は、open() 呼び出し中のモジュールの初期設定に使用されます。この構造体は、チャネル、割り込み優先順位、ビットレート、動作モードなどのメンバーで構成されます。この構造体は、モジュールへの入力にのみ使用されます。この構造体は一意にする必要がなく、必要に応じてユーザーが初期化後に破棄することができます。

```
/** I2C configuration block */
typedef struct st_i2c_cfg
{
  /** Generic configuration */
  ///< Identifier recognizable by implementation
  uint8_t channel;
  ///< Device's maximum clock rate from enum i2c_rate_t
  i2c_rate_t rate;
  uint16_t slave; ///< The address of the slave device
  ///< Indicates how slave fields should be interpreted
  i2c_addr_mode_t addr_mode;
  uint16_t sda_delay; ///< The SDA output delay
  uint8_t rxi_ipl; ///< Receive interrupt priority
  uint8_t txi_ipl; ///< Transmit interrupt priority
  uint8_t tei_ipl; ///< Transmit end interrupt priority
  uint8_t eri_ipl; ///< Error interrupt priority
```

```
  /** DTC/DMA support */
  ///< DTC instance for I2C transmit.Set to NULL if unused.
  transfer_instance_t const * p_transfer_tx;
  ///< DTC instance for I2C receive. Set to NULL if unused.
  transfer_instance_t const * p_transfer_rx;

  /** Parameters to control software behavior */
  ///< Pointer to callback function
  void (* p_callback)(i2c_callback_args_t * p_args);
  ///< Pointer to the user-provided context
  void const * p_context;

  /** Implementation-specific configuration */
  ///< Any configuration data needed by the hardware
  void const * p_extend;

} i2c_cfg_t;
```

上は I2C インタフェースの設定構造体の例です。最後の 3 つの構造体メンバー (p_callback、p_context、および p_extend) は、ほぼすべてのモジュール設定に共通です。

p_callback および *p_context* メンバーについては、「SSP インタフェース：コールバック関数」のセクションで説明します。

p_extend メンバーは、特定のインスタンスの現在のインタフェースを拡張するために使用されます。インタフェースは最も一般的な機能をサポートするように設計されています。インタフェースのインスタンスがそれ自体を正しく構成するために、特別な情報が必要な場合があります。そうした情報が不要な場合もありますが、ユーザーが特定のアプリケーションに合わせてモジュールを調整するために必要になる場合もあります。その場合、ユーザーは *p_extend* メンバーを通じて、基礎となるインスタンスに追加の構成情報を渡すことができます。このメンバーを介して渡される情報は基礎となるインスタンスによって定義されるため、ユーザーはその構造体に忠実に従う必要があります。無効な情報が基礎となるドライバーに渡された場合は、インスタンスでそのデータを正しく処理できないため、正常な動作が保証されません。詳細については、「インタフェース拡張機能」の項を参照してください。

構成構造体に現在のインタフェースに適用されるメンバーだけが入っていることも重要です。同じスタック内の複数のレイヤーで同じ設定パラメータが定義されていると、オプションをどこで変更するかを知るのが難しくなります。たとえば、UART のボーレートはドライバーレイヤーでのみ定義されます。UART インタフェースを使用するすべてのレイヤーは、ドライバーレイヤーで指定されているボーレートに依存し、独自の構成構造体でボーレートを提供することはありません。

2.2.4.3 SSP インタフェースコールバック関数

コールバック関数を使用すれば、モジュールはイベント発生時に非同期的にユーザーアプリケーションに通知することができます。イベントの例には、UART チャネル経由でのバイト受信があります。ユーザーアプリケーションコードで割り込みに対処するには、コールバックが必要です。SSP モジュールが Synergy MCU ペリフェラルの割り込みを定義して処理します。ユーザーが SSP モジュールと同時に割り込みサービスルーチンを定義しようとしても、そのコードはビルドされません。そのため、SSP モジュールを使用し、割り込

み発生時に呼び出す関数を登録しておくことで、ユーザーアプリケーションが割り込みに対処できるようになります。

コールバック関数はユーザーアプリケーション内で定義する必要があります。この関数は常に *void* を返し、1 つのパラメータに対して 1 つの構造体を使用します。構造体の typedef はモジュールのインタフェースで指定され、<interface>_callback_args_t. という名前になります。構造体の内容はインタフェースによって異なる可能性がありますが、*event* と *p_context* の 2 つは共通です。

event メンバーは、アプリケーションが、コールバックが呼び出された理由を特定するために使用されます。前の UART の例では、バイトが受信された、すべてのバイトが送信された、またはフレーミングエラーが発生した場合にコールバックがトリガされた可能性があります。*event* メンバーはインタフェースで指定される列挙です。

p_context メンバーは、ユーザーが指定したデータをコールバック関数に渡すために使用されます。多くの場合、コールバック関数は、複数のチャネルまたはモジュールインスタンス間で共有されます。コールバックが発生すると、それを処理するコードがコールバック対象のモジュールインスタンスを特定するためにコンテキスト情報を必要とします。たとえば、コールバックで SSP API コールを発行する場合は、少なくともコールバックで制御構造体を使用する必要があります。これを簡単にするために、ユーザーは制御構造体へのポインタを *p_context* として指定できます。コールバックが発生するときは、コールバック構造体で渡される制御構造体を使用できます。

コールバック関数は割り込みサービスルーチン内から呼び出されます。そのため、ユーザーのシステムのリアルタイム性能に影響を与えないようにコールバック関数はできるだけ短くする必要があります。フラッシュインタフェースコールバックのスケルトン関数の例を以下に示します。

```
static void flash_callback (flash_callback_args_t * p_args)
{
    /** See what event caused this callback. */
    switch (p_args->event)
    {
        case FLASH_EVENT_ERASE_COMPLETE:
        /**Handle event. */
        break;

        case FLASH_EVENT_WRITE_COMPLETE:
        /**Handle event. */
        break;

        case FLASH_EVENT_BLANK:
        /**Handle event. */
        break;

        case FLASH_EVENT_NOT_BLANK:
        /**Handle event. */
        break;

        case FLASH_EVENT_ERR_DF_ACCESS:
        /** Handle error. */
        break;

        case FLASH_EVENT_ERR_CF_ACCESS:
```

```
        /** Handle error. */
        break;

        case FLASH_EVENT_ERR_CMD_LOCKED:
        /** Handle error. */
        break;
    }
}
```

モジュールがユーザーアプリケーション内で直接使用されていない（たとえば、スタックの最上位レイヤーではない）場合は、そのコールバック関数が上記モジュールによって処理されます。UART インタフェースモジュールが必要なコンソールインタフェースモジュールが存在する場合は、コンソールモジュールが UART のコールバック関数を制御および使用します。この場合、ユーザーはアプリケーションコード内で UART モジュール用のコールバック関数を作成する必要がありません。

2.2.4.4 SSP インタフェース API 構造体

すべてのインタフェースには、サポートされているすべてのインタフェース関数の関数ポインタを含む API 構造体が含まれています。コメントを削除した、デジタル / アナログ変換 (DAC) 用の構造体の例を以下に示します。

```
typedef struct st_dac_api
{
    ssp_err_t (*open)(dac_ctrl_t *p_ctrl, dac_cfg_t const *const p_cfg);
    ssp_err_t (*close)(dac_ctrl_t *p_ctrl);
    ssp_err_t (*write)(dac_ctrl_t *p_ctrl, dac_size_t *p_value);
    ssp_err_t (*start)(dac_ctrl_t *p_ctrl);
    ssp_err_t (*stop)(dac_ctrl_t *p_ctrl);
    ssp_err_t (*versionGet)(ssp_version_t *p_version);

} dac_api_t;
```

API構造体は、モジュールを同じインタフェースのインスタンスである他のモジュールに対して簡単にスワップインまたはスワップアウトできるようにするために使用されます。上記 DAC インタフェースを使用したアプリケーションの例を見てみましょう。

Synergy MCU は DAC ペリフェラルを内蔵しています。DAC インタフェースで DAC API 構造体が使用されていない場合は、アプリケーションから直接モジュールを呼び出すことができます。下の例では、アプリケーションは r_dac モジュールで提供されている R_DAC_Write 関数を呼び出しています。

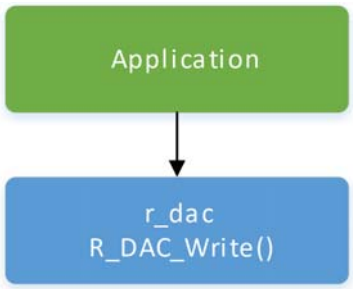


図 6: DAC 書き込みの例

ここで、MCU 上で使用可能な DAC チャネルだけでは足りず、r_dac_external. という名前の新しい外部 DAC モジュールを追加する場合を考えます。外部 DAC は通信に I2C を使用します。アプリケーションは 2 つのモジュールを区別する必要があるため、複雑さが増してアプリケーションに対する依存度が高まります。

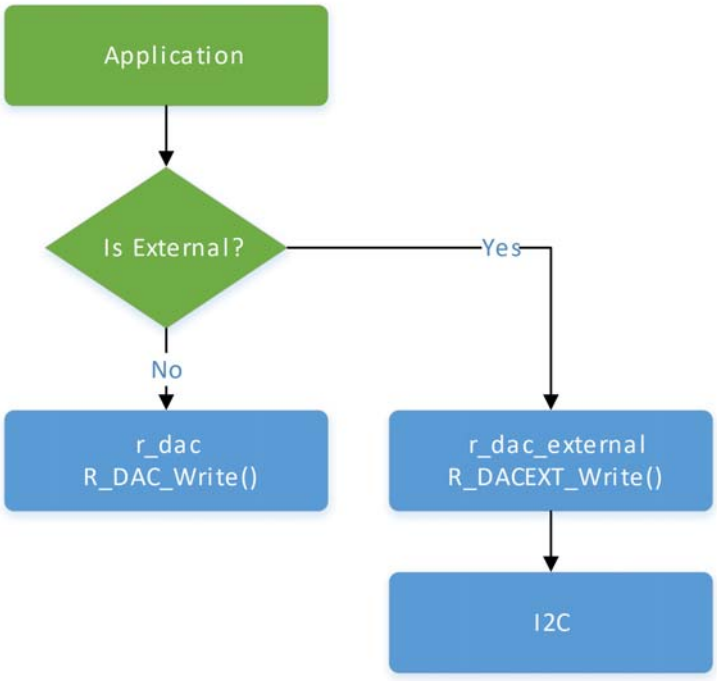


図 7: 2 つの書き込みモジュールを使用した DAC 書き込み

インタフェースと API 構造体を使用すると、抽象化された DAC を使用できます。つまり、外部ロジックが不要で、アプリケーションは特定のハードコードされたモジュールに依存しません。代わりに、アプリケーションは、任意の数のモジュールで実装可能な DAC インタフェース API に依存します。

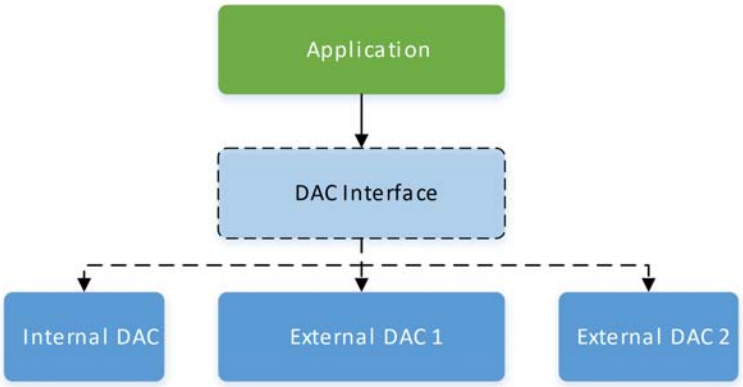


図 8: DAC インタフェース

API 構造体の内部関数は共通の名前に従っています。ほとんどのモジュールが open() 関数と close 関数のペアを使用します。open() 関数は他の関数の前に呼び出す必要があります。唯一の例外は、ユーザーが指定した情報に依存しない versionGet() 関数です。

共通して使用される他の関数は、read()、write()、get()、および set() です。関数名は名詞の後に動詞が続くように設計されています。名前の例を以下に示します。

- read()、write()、writeRead()
- statusGet()
- calendarAlarmSet()、calendarAlarmGet()
- accessWindowSet()、accessWindowClear()

2.2.4.5 SSP インタフェースバージョン情報

すべてのインタフェースが versionGet() 関数を提供します。この関数は ssp_version_t という型の構造体を返します。この構造体は、インタフェース（API）用と現在使用されている基礎となるインスタンス用の 2 つのバージョンで構成されます。

```
/** Common version structure */
typedef union st_ssp_version
{
/** Version id */
uint32_t version_id;
/** Code version parameters */
struct
{
    uint8_t code_version_major; ///< Code major version
    uint8_t code_version_minor; ///< Code minor version
    uint8_t api_version_major; ///< API major version
    uint8_t api_version_minor; ///< API minor version
```

```
};
} ssp_version_t;
```

API バージョンは変更されないことが理想ですが、稀に変更される場合があります。API を変更するには、さかのぼってコードを変更する必要があります。コードバージョン、つまり、現在のインスタンスのバージョンは頻繁に更新される可能性があります。バグの修正、機能強化、および追加機能のすべては、コードバージョンの増加につながる可能性があります。ユーザーコードがインスタンスから提供される拡張機能を使用している場合、コードバージョンの変更では、ユーザーコードの変更のみが必要です。

2.2.4.6 SSP インスタンス

インタフェースは指定された機能を指示するのに対して、インスタンスは実際にそれらの機能を実装します。インスタンスはそれぞれ、特定のインタフェースに関連付けられます。インスタンスは、インタフェースの列挙、データ構造体、および API プロトタイプを使用します。これにより、インタフェースを使用するアプリケーションは必要に応じてインスタンスをスワップアウトすることができます。

Synergy MCU では、一部のペリフェラルがインタフェースとインスタンスの 1 対 1 マッピングを使用しますが、それ以外のペリフェラルは 1 対多マッピングを使用します。下の例では、IIC と SPI がそれぞれ 1 つのインタフェースにしかマッピングされていないのに対して、SCI は 3 つのインタフェースを実装しています。

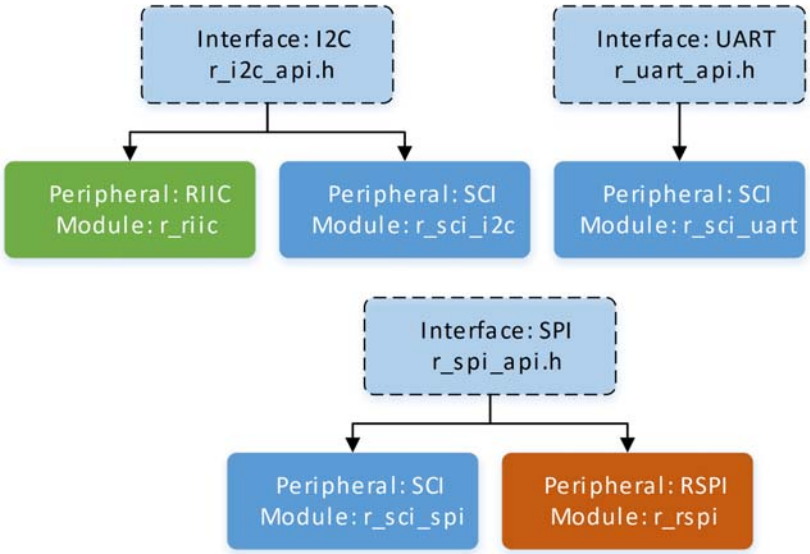


図 9: インスタンス

2.2.4.7 SSP インスタンスの API 構造体

各インスタンスには、インタフェースの API を実装するその関数を含む固定のグローバル構造体が含まれています。この構造体の名前は、g_<interface>_on_<instance>として標準化されています。例として、g_spi_on_rspi、

g_transfer_on_dtc、g_adc_on_adc などが挙げられます。この構造体は、インスタンスヘッダーファイル（それぞれ、r_rspi.h、r_dtc.h、および r_adc.h) 内の extern 経由で使用できます。

2.2.5 ビルド時設定

すべてのモジュールに、ビルド時設定ヘッダーファイルが割り当てられています。ほとんどの構成オプションは実行時に提供されます。稀にしか使用されない、または、モジュールのすべてのインスタンスに適用される一部のオプションはビルド時に移動される場合があります。ビルド時構成オプションを使用するメリットは、未使用の機能を削除することでコードサイズを小さくできる可能性があることです。性能を強化することもできます。

すべての SSP モジュールに、それぞれのパラメータチェックを有効化または無効化するビルド時オプションが 1 つ以上付属しています。SSP モジュールは、可能な限り、関数引数の妥当性をチェックします。ユーザーは、テストの完了時にこの機能を無効化して、コードスペースを節約し、実行速度を上げることができます。

2.2.6 インタフェース拡張機能

インスタンスでは、インタフェースから提供されるよりも多くの情報が必要になることがあります。この状況は次の 2 つの場合に発生する可能性があります。

- あるインスタンスがインタフェースのほとんどのインスタンスに共通しない特別な機能を提供している。
- あるインタフェースをどうしても汎用的にしなければならない。インタフェースが汎用的になるほど、使用可能なインスタンスの数が増えます。この典型例がブロックメディアインタフェースです。

```
/** Interface Configuration */
typedef struct st_sf_block_media_cfg
{
    uint32_t block_size; ///< Block size in bytes
    void const * p_extend; ///< Instance dependent configuration
} sf_block_media_cfg_t;
```

ブロックメディアインタフェースの構成構造体は意図的に小さく作られています。これにより、ほぼ無限のインスタンスが可能になります。例として、SD カード、SPI フラッシュ、SDRAM、USB などが挙げられます。インスタンスごとに別々の構成情報が必要です。これは、*p_extend* パラメータ経由で情報を提供することによって実現されます。この *p_extend* 内で提供される構成データはインスタンス間で同じではありませんが、その後の API 呼び出しは同じです。これは、1 か所を変更するだけで済むことを意味します。

必ずしも、インタフェース拡張機能を使用する必要はありません。一部のインスタンスについては、機能がすべてインタフェースで提供されるため、拡張機能がありません。大抵の場合、*p_extend* メンバーを NULL に設定できます。NULL が指定され、インスタンスが拡張機能を提供している場合、インスタンスはこれを、デフォルトのオプションが使用されるものと見なします。インタフェース拡張機能が提供されているかどうかと、その使用が必須か任意かについては、各インスタンスのドキュメントで確認できます。

2.2.7 SSP 事前定義レイヤー

SSP には、2 つの事前定義レイヤー（ドライバーレイヤーとフレームワークレイヤー）が付属しています。これらのレイヤーは、モジュールが別々のフォルダー内に存在し、プレフィックスが異なるため、簡単に識別できます。ドライバーレイヤーモジュールは *ssp/src/driver* フォルダに配置され、フレームワークレイヤーモジュールは *ssp/src/framework* フォルダに配置されます。ドライバーレイヤー内のモジュールは *r_* プレフィックスで始まりますが、フレームワークレイヤーモジュールは *sf_* プレフィックスで始まります。

両レイヤー間の機能的な主な違いとして、ドライバーレイヤーモジュールには RTOS に対応したペリフェラルドライバーでなければならないという制限がありますが、RTOS オブジェクトの使用や RTOS API コールを行うことはありません。これは、RTOS のある（またはない）アプリケーションでドライバーレイヤーモジュールを使用できることを意味します。

フレームワークレイヤーモジュールはセマフォ、ミューテックス、イベントフラグなどの RTOS オブジェクトを自由に使用できます。フレームワークモジュールは、必要に応じて独自の RTOS オブジェクトを作成することもできます。ハードウェアにアクセスする必要があるフレームワークレイヤーモジュールは、通常、ドライバーレイヤーインタフェースを通じてそれを行います。複数のペリフェラルを同時に使用しなければならず、複数の個別インタフェースを使用することが実際的でない特殊な場合は、例外を認めることができます。

2.2.8 SSP ファイルの構造

SSP のファイル構造の概要を以下に示します。

```
ssp
+---inc
|   +---bsp
|   +---driver
|   |   +---api
|   |   \---instances
|   \---framework
|       +---api
|       \---instances
---src
    +---bsp
    +---driver
    |   \---r_module
    \---framework
        \---sf_module
ssp_cfg
+---bsp
+---driver
\---framework
```

ベースとなる *ssp* フォルダの直下では、フォルダが source フォルダと include フォルダに分岐しています。include フォルダーは、include パスの参照とセットアップを容易にするために source から分離されています。*ssp/inc* フォルダと *ssp/src* フォルダの下に、同じフォルダのセット（*bsp*、*driver*、*framework*）があります。

BSP と違って、SSP の 2 つの事前定義レイヤー（ドライバーとフレームワーク）が存在します。BSP と違って、SSP の 2 つの事前定義レイヤー（ドライバーとフレームワーク）が存在します。ドライバーレイヤーモジュールは *ssp/src/driver* フォルダーに位置していますが、フレームワークレイヤーモジュールは *ssp/src/framework* フォルダーに位置しています。include ツリーの下では、ドライバーレイヤーフォルダーとフレームワークレイヤーフォルダーにそれぞれ 2 つずつのフォルダー(*api* と *instances*) が含まれています。*api* フォルダーには、そのレイヤー用のインタフェースヘッダーファイルが含まれています。instances* フォルダーには、そのレイヤー用のインスタンスヘッダーファイルが含まれています。両方のレイヤーが内部的に平坦であるため、プロジェクトに必要な include パスの数は制限されます。

ssp_cfg フォルダーには、モジュールごとの構成ヘッダーファイルが保存されます。そのレイアウトは、*ssp* フォルダーと同じで、BSP、Driver、および Framework レイヤーが別々の平坦なディレクトリで構成されます。これらのヘッダーファイルで提供される情報については、「ビルド時設定」の項を参照してください。

2.2.9 SSP 接続レイヤー

SSP モジュールは再利用可能かつスタック可能に設計されています。モジュールは他のモジュールには依存しませんが、他のインタフェースには依存することを覚えておいてください。ニーズに最適なインスタンスを使用したインタフェースを自由に構築できます。

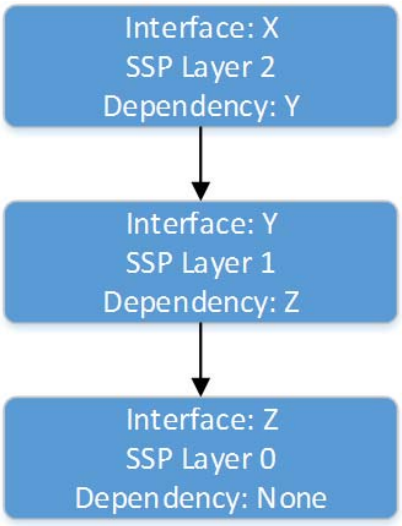


図 10: 接続レイヤー

上の図では、インタフェース Y がインタフェース X から依存されていると同時に、インタフェース Z に依存しています。インタフェース X はインタフェース Y に依存しているのみです。また、インタフェース X はインタフェース Z を認識していません。レイヤーを簡単にスワップアウトできるようにするには、このようにする必要があります。これを下の図に示します。

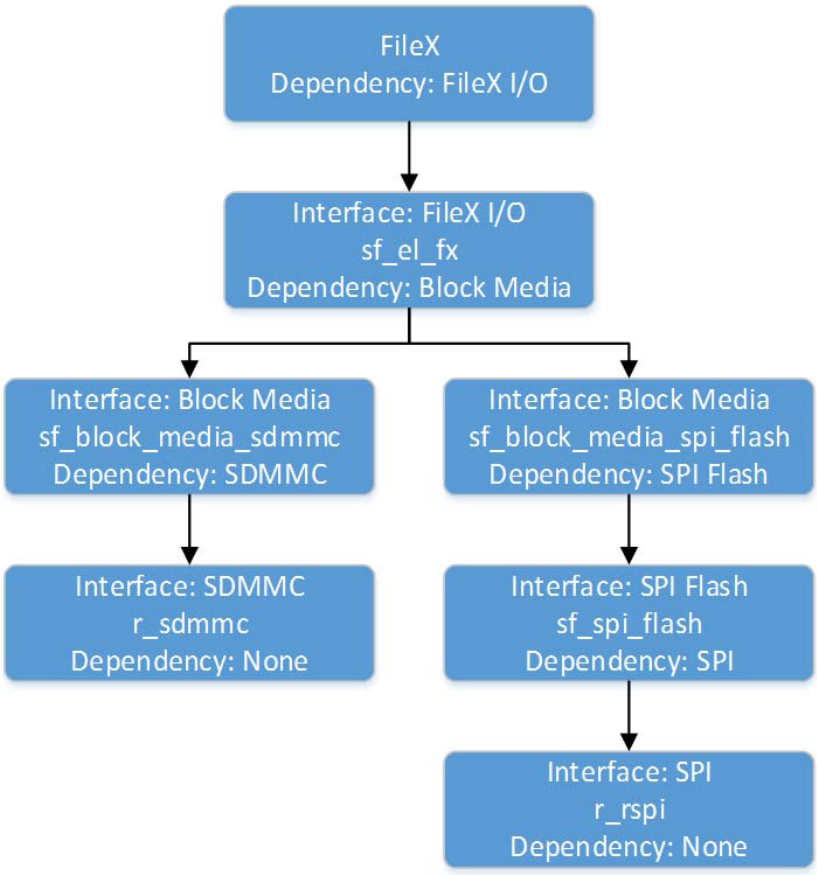


図 11: FileX インタフェースを使用した接続レイヤー

この例では、2 つのストレージメディア（SDMMC と SPI フラッシュ）上の Express Logic, Inc. FileX ファイルシステムを使用しています。SPI フラッシュインタフェースは SPI フラッシュプロトコルを処理しますが、実際の SPI バス通信には SPI インタフェースが必要です。SDMMC インタフェースはプロトコルとバス通信を処理するため、何にも依存しません。

2.2.10 実際の SSP アーキテクチャ

SSP スタック内の各レイヤーには、その依存対象の API 関数を呼び出す役割があります。このことは、ユーザーが自身のやり取りするレイヤーでしか API 関数を呼び出さないと言い換えることもできます。上の FileX の例では、ユーザーはアプリケーションコード内で FileX 関数を呼び出すだけです。内部的に、この後 FileX は FileX I/O を呼び出し、次いで FileX I/O はブロックメディアインタフェースモジュールを呼び出します。ブロックメディアインタフェースは複数のドライバーを呼び出すことができます。上位レイヤーモジュールでは少なくとも、依存しているインタフェースの *open()* 関数を呼び出します。

モジュールを使用してアプリケーションを作成するには、以下のようにします。

1) 呼び出す open() 関数を決定します。依存関係はインタフェースに基づくので、モジュールは呼び出すインスタンスを何らかの方法で識別できなければなりません。
2) 設定パラメータを決定します。モジュールは、伝達する構成情報を認識する必要もあります。モジュールが特定の構成パラメータを設定しなければならない場合もあります。その場合は、モジュールがそれらの構成構造体メンバー自体を設定してから、残りの構造体を伝達します。残りの設定構造体メンバーはモジュール外部で指定する必要があります。
3) モジュールインスタンス固有であり、上位レイヤーモジュールで割り当て可能な制御構造体を提供します。

つまり、モジュールインスタンスとやり取りするには次のポインタが必要です。

- インスタンスの API 構造体へのポインタ
- モジュールインスタンスの構成構造体へのポインタ
- モジュールインスタンスの制御構造体へのポインタ

これは、任意のモジュールを使用するのに十分な情報です。API 構造体はインスタンスに固有である唯一の構造体であり、*モジュールインスタンス*固有ではありません。これは、API 構造体が同じインスタンスの複数の使用で変化しないためです。SPI が SCI チャネル 0 および 2 で使用されている場合は、両方のモジュールインスタンスで同じ API 構造体が使用されますが、構成および制御構造体は異なります。

モジュールインスタンスをより使いやすくするため、これらのすべての部分は各インタフェースで見つかったインスタンス構造体内にカプセル化されます。これらの構造体は、<interface>_instance_t. という標準化された名前を持っています。WDT インタフェースからの例を下に示します。

```
/** This structure encompasses everything that is needed to use an instance of t
his interface. */
typedef struct st_wdt_instance
{
    wdt_ctrl_t * p_ctrl; ///< Pointer to the control structure forthis instance
    wdt_cfg_t const * p_cfg; ///< Pointer to the configuration structure for thi
s instance
    wdt_api_t const * p_api; ///< Pointer to the API structure for this instance

} wdt_instance_t;
```

インタフェースに対する依存関係を持つ上部レイヤーモジュールは、インスタンス構造体を使用して、そのインタフェースのインスタンスとやり取りするのに必要なすべての情報を保持できます。上の WDT の例に続いて、スレッド監視フレームワークインタフェース構成構造体を下に示します。このスレッド監視インタフェースは WDT インタフェースに依存しています。

```
/** Configuration for Thread Monitor framework */
typedef struct st_sf_thread_monitor_cfg
{
    wdt_instance_t * p_lower_lvl_wdt; ///< Pointer to lower level watchdog insta
nce
    bool profiling_mode_enabled; ///< Enables or disables profiling mode
    UINT priority; ///< Priority of thread monitor thread
```

```
    void const * p_extend; ///< Instance dependent configuration

} sf_thread_monitor_cfg_t;
```

スレッド監視モジュールは、WDT インタフェースとやり取りするのに必要なすべての情報を *p_lower_lvl_wdt* 構造体メンバー内に持っています。

場合により、モジュール依存関係はインタフェースではなくインスタンスで定義されます。たとえば、ブロックメディアインタフェースは、SDMCC、SPI フラッシュ、または他の多くのインスタンスで実装できます（も参照）。実装方法は広範囲にわたるため、特定のインタフェースのインスタンス構造体をブロックメディアインタフェースの構成構造体で直接使用することはできません。ブロックメディアインタフェースの構成構造体を再び下に示します。

```
/** Interface Configuration */
typedef struct st_sf_block_media_cfg
{
    uint32_t block_size; ///< Block size in bytes
    void const * p_extend; ///< Instance dependent configuration

} sf_block_media_cfg_t;
```

インスタンス構造体ポインタが指定されていることに注意してください。この理由は、前述のようにブロックメディアインタフェースが汎用的すぎて特定のインタフェースに対する依存関係を強制できないためです。

モジュールがブロックメディアなどの汎用インタフェースのインスタンスで、他のモジュールに依存している場合は、インタフェースの *p_extend* 構成メンバー経由で参照される拡張機能構造体内に下位レイヤーポインタを配置します。これは、インタフェースの拡張を強制せずにモジュールスタッキングを可能にして、多数のオプション構成メンバーを使用できるようにするために必要です。

```
typedef struct st_block_media_on_sdmmc_cfg
{
    sdmmc_instance_t const * const p_lower_lvl_sdmmc; ///< Pointer to SDMMC inst
ance structure

} sf_block_media_on_sdmmc_cfg_t;
```

2.2.11 SSP モジュールの使用

ここでは、SSP モジュールの使用方法に関する一般的な情報を提供します。

2.2.11.1 インタフェースの選択

最初に必要な機能用のインタフェースを選択します。たとえば、UART 通信には UART インタフェースを使用します。

2.2.11.2 インタフェースの適切なインスタンスの検索

インタフェースを選択したら、それに対応するインスタンスを選択します。インタフェースの既知のインスタンスのリストは、インタフェースのドキュメンテーションコメント内に記載されています。選択したインスタンスのヘッダーファイルを、そのインスタンスを使用するアプリケーションのソースファイルにインクルードします。

2.2.11.3 制御構造体と構成構造体の割り当て

e^2 studio 統合ソリューション開発環境には、インタフェースおよびインスタンス構成構造体のパラメータを設定するためのグラフィカルユーザーインタフェースが備わっています。また、GUI で構成されたそれらの構造体は、アプリケーションコードにインクルード可能なアプリケーション固有のヘッダーファイルに自動的に含められます。

ISDE で構成が処理される方法については、『ISDE ユーザーズガイド』の e^2 studio ISDE ユーザーガイドを参照してください（プロジェクトの設定）。

構成および制御構造体の型は、それぞれ <interface>_ctrl_t および <interface>_cfg_t, という標準名に従います。ISDE では、ISDE が作成するアプリケーション固有のヘッダーファイル内に、これら両方の構造体用のストレージが割り当てられます。ISDE の [Properties] ビューを使用し、必要に応じて構成構造体のメンバーの値を設定します。多くのメンバーが、使用可能なオプションで列挙を参照可能な型指定列挙です。

インタフェースにコールバック関数が付属している場合は、最初にソースコードでその関数を宣言して定義する必要があります。戻り値は常に型 *void* で、関数へのパラメータは <interface>_callback_args_t. という名前の型指定構造体です。関数が定義されたら、その名前を構成構造体の *p_callback* メンバーに割り当てます。コールバックでコンテキスト情報が必要な場合は、*p_context* メンバーへのポインタを指定します。統合ソリューション開発環境で選択したモジュールの [Properties] ウィンドウを通じて、コールバック関数名を割り当てることができます。

インタフェース拡張機能が提供されているかどうかを確認するには、インスタンスドキュメントを参照してください。提供されている場合は、*<interface>_*on_*<instance>_cfg_t*. という名前のインスタンスヘッダーファイルに含まれています。インタフェースの構成構造体と同様に、複数のメンバーで構成される場合があります。これらのメンバーは、統合ソリューション開発環境を使用して設定することができます。

2.2.11.4 インタフェースのインスタンス構造体を使用したやりとり

インスタンス構造体を作成した後は、必要に応じてインスタンスとやり取りできます。下に示すのは、SCI で実装されている UART インタフェースのインスタンス構造体を構築するコードです。e^2 studio を使用する場合は、次のコードが自動的に生成されることに注意してください。

```
/** Include the header file of the Instance. */
/** This will in turn include the r_uart_api.h Interface */
#include “r_sci_uart.h”

/** Allocate Instance specific control structure. This will be used in place of
uart_ctrl_t. */
sci_uart_instance_ctrl_t my_uart_ctrl;

/** Setup extended UART configuration on SCI. */
uart_on_sci_cfg_t my_uart_extended_cfg =
```

```
{

/** Set extended configuration members... */

};

/** Configure standard UART Interface. */
uart_cfg_t my_uart_cfg =

{
    .data_bits = UART_DATA_BITS_8,
    /** Continue configuring other members... */
    .p_extend = &my_uart_extended_cfg
};

/** Setup instance structure */

uart_instance_t my_uart = {
    .p_ctrl = &my_uart_ctrl,
    /** Extended configuration is brought through in p_extend */
    .p_cfg = &my_uart_cfg,
    .p_api = &g_uart_on_sci ///< Defined in r_sci_uart.h
};
```

これでインスタンス構造体は準備完了です。UART インタフェースとやり取りできます。e^2 studio では、インスタンス構造体の名前は、統合ソリューション開発環境の [Properties] ウィンドウでモジュールインスタンスを構成するときに指定した*名前*です。

```
ssp_err_t err;
/** Initialize UART */
err = my_uart.p_api->open(my_uart.p_ctrl, my_uart.p_cfg);

/** Check return for errors. */
if (SSP_SUCCESS != err)
{

    /** Handle error. */

}

/** Use other Interface functions. */
err = my_uart.p_api->write(my_uart.p_ctrl, ...);
err = my_uart.p_api->read(my_uart.p_ctrl, ...);
```

2.2.12 コーディングスタイル

2.2.12.1 C99 の使用

SSP では、ISO/IEC 9899:1999 (C99) C プログラミング言語規格が使用されます。使用する C99 で導入された特定の機能には、標準の整数型 (stdint.h)、ブール型 (stdbool.h)、指定初期化子、および宣言とコードを組み合わせる機能が含まれます。

2.2.12.2 API パラメータ内での const の使用

const 修飾子は必要に応じて API パラメータと一緒に使用されます。実例を以下に示します。

```
ssp_err_t (* open)(flash_ctrl_t * const p_ctrl,
                   flash_cfg_t const * const p_cfg);
```

絶対確実というわけではありませんが、これにより SSP コード内で追加のチェックが実行されるため、変更すべきではない引数に対する変更を防ぐうえで役立ちます。

2.2.12.3 Weak シンボル

Weak シンボルは SSP 内で使用される場合と SSP と一緒に使用される場合があります。このシンボルを使用すると、ユーザーがオプション関数を定義していない場合にもプロジェクトをビルドできます。

2.3 BSP アーキテクチャ

2.3.1 BSP アーキテクチャ

このセクションでは、BSP（ボードサポートパッケージ）について説明します。API リファレンスについては、Board Support Package を参照してください。BSP はボード固有であり、結果的に MCU 固有です。

2.3.2 BSP の機能

BSP には、リセット以降の MCU からユーザーアプリケーション（main() 関数など）に到達できるようにする責任があります。ユーザーアプリケーションに到達する前に、BSP は、スタック、ヒープ、クロック、割り込み、C ランタイム環境をセットアップします。また、BSP は、ポート I/O ピンを設定およびセットアップし、ボード固有の初期化も行います。

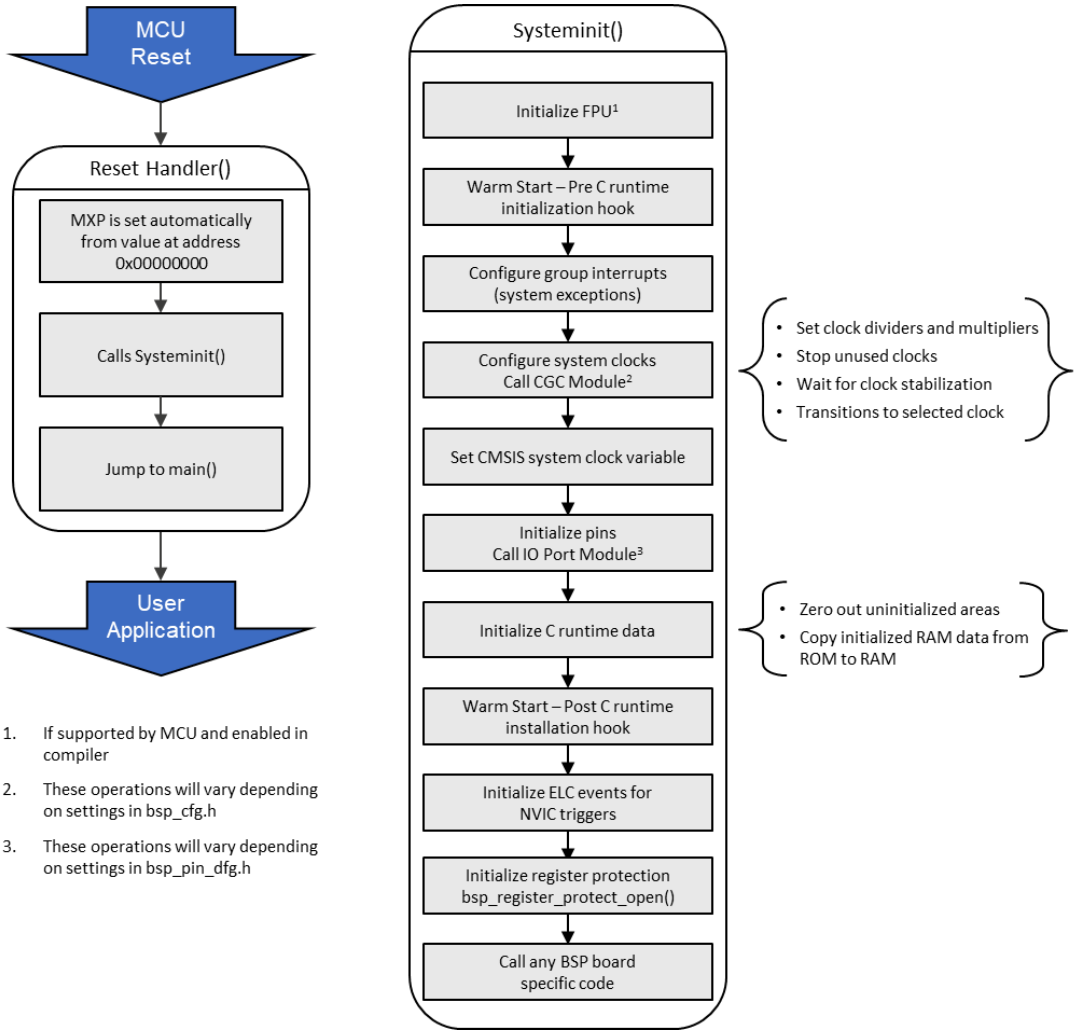


図 12: BSP のフロー

2.3.3 BSP 関連の用語

Term	意味
Filename_xxxx.c	'xxxx' は MCU の種類を表します。たとえば、S7G2 MCU を参照する場合は Filename_S7G2.c になります。
BSP	Board Support Package の略語。BSP には、一般に特定のボードに関連するソースファイルがあります。

Term	意味
Callback Function	この用語はイベント発生時に呼び出される関数を意味します。ユーザーは、NMI システム例外の発生を知りたい可能性があります。ユーザーに通知するため、グループ割り込み（すべてが NMI に結び付けられている例外のグループ）に対してコールバック関数を設定できます。NMI が発生すると、BSP は指定されたコールバック関数にジャンプして、ユーザーがエラーを処理できます。割り込みコールバック関数は長くならないように慎重に扱う必要があります。これは、この関数が呼び出されたときに、MCU はまだ割り込みの内部にあり、保留中の割り込みが後回しにされるためです。

2.3.4 BSP ディレクトリー構成

BSP は、MCU 情報、ボード固有情報、CMSIS 情報が格納されるフォルダーに編成されます。

Synergy は CMSIS に準拠しており、CMSIS-Core に基づいています。そのためには、下記の CMSIS の要件と命名規則に従う必要があります。

- プロセッサペリフェラルに対する標準化された定義
  - NVIC（ネスト型ベクトル割り込みコントローラ）
  - Systick（システムティックタイマ）
  - MPU（メモリ保護ユニット）
- プロセッサ機能にアクセスするための標準化されたアクセス関数
  - NVIC_SetPriority()
  - NVIC_EnableIRQ
- システム例外ハンドラーのための標準化された関数名
  - Reset_Handler()
  - SysTick_Handler()
- システム初期化のための標準化された関数。
  - SystemInit() – system_S7G2.c で S7G2 MCU 用に定義
- クロック速度情報のための標準化されたソフトウェア変数
  - SystemCoreClock

BSP ディレクトリー構成は以下のとおりです。

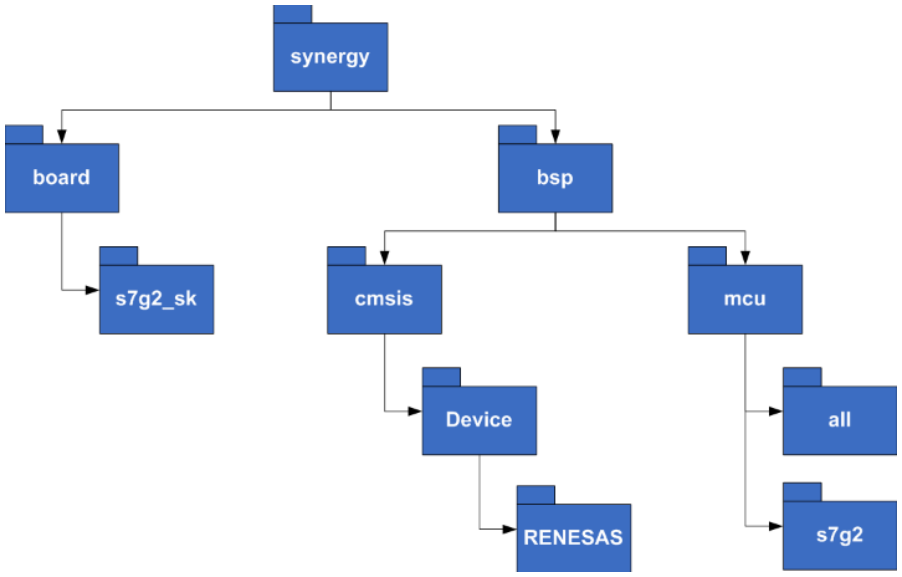


図 13: BSP ディレクトリー構成

2.3.5 BSP の設定

BSP は非常にデータ駆動型であり、ほとんどの機能は、設定ファイルの内容に基づいて設定されます。設定ファイルは、ユーザーによって指定された設定を表しており、[Generate Project Content] ボタンをクリックしたときに、ISDE によって生成されます。

2.3.6 BSP コンフィギュレーション設定

次の表では、BSP のそれぞれの変更可能な設定について説明します。これらの設定の多くは MCU 固有であり、サポートされるそれぞれの MCU で使用可能な設定に違いがあります。

表 : BSP 設定オプション

BSP Property	説明
Part number	MCU 部品番号
ram_size_bytes	この MCU パッケージで使用可能な RAM
rom_size_bytes	この MCU パッケージで使用可能な ROM
data_flash_size_bytes	この MCU パッケージで使用可能なデータフラッシュ

BSP Property	説明
package_style	パッケージのスタイル （たとえば BGA）
package_pins	この MCU パッケージのピン数
series	MCU パーツシリーズ
Main stack size (bytes)	メインスタックのサイズ。0 よりも大きい必要があります。
Process stack size (bytes)	プロセススタックのサイズ。このスタックの使用はオプションです。 0 の場合、 PSP の使用が無効になります。
Heap size (bytes)	ピープのサイズ（バイト単位）。0 の場合、ヒープが無効になります。
OFS0 register settings: IWDT Start Mode, IWDT Timeout Period, IWDT Dedicated Clock Frequency Divisor, IWDT Window End Position, IWDT Window Start Position, IWDT Reset Interrupt Request Select, IWDT Stop Control, WDT Start Mode Select, WDT Timeout Period, WDT Clock Frequency Division Ratio, WDT Window End Position, WDT Window Start Position, WDT Reset Interrupt Request, WDT Stop Control and the program flash area of the flash memory.	詳細は MCU ユーザーマニュアルを参照してください。
OFS1 register settings: Voltage Detection 0 Circuit Start, Voltage Detection 0 Level, HOCO Oscillation Enable. S3 MPU has MPU configuration settings.	詳細は MCU ユーザーマニュアルを参照してください。
MPU - Enable or disable PC Region 0	アクセスウィンドウ保護のための開始ブロックアドレス
MPU - PC0 Start, MPU - PC0 End, MPU - Enable or disable PC Region 1, MPU - PC1 Start, MPU - PC1 End, MPU - Enable or disable Memory Region 0, MPU - Memory Region 0 Start, MPU - Memory Region 0 End, MPU - Enable or disable Memory Region 1, MPU - Memory Region 1 Start, MPU - Memory Region 1 End, MPU - Enable or disable Memory Region 2, MPU - Memory Region 2 Start, MPU - Memory Region 2 End, MPU - Enable or disable Memory Region 3, MPU - Memory Region 3 Start, MPU - Memory Region 3 End	セキュア MPU ROM レジスタ設定。詳細は MCU ユーザーマニュアルを参照してください。
ID Code Mode, ID Code (32 Hex Characters)	ブートモードとデバッガーアクセス保護のための ID コードを設定します。
MCU Vcc (mV)	一部のモジュール（LVD など）では、MCU に供給される電圧を知る必要があります。この情報はここから取得されます。

BSP Property	説明
Parameter checking	パラメータチェックのグローバル設定を有効にするか無効にするかを定義します。ローカルモジュールはこの値をデフォルトで取得しますが、ローカルにオーバーライドすることができます。
Assert Failures	アサーション失敗が発生したときに行われる処理を定義します。
Error Log	エラーを ssp_error_log に記録するかどうかを定義します。

2.3.7 BSP の設定ファイル

設定ファイルは、BSP により、ROM レジスタ、クロック、割り込み、ELC イベント、および初期ピンを設定するために使用されます。これらの設定ファイルは、ssp_cfg\bsp にあります。

2.3.7.1 bsp_cfg.h

この構成ファイルは、BSP システム設定の値を含みます。これは、ISDE の BSP プロパティタブで変更可能な設定です。これには、ROM レジスタ設定、スタックおよびヒープサイズ、パラメータチェック、エラーロギングの制御が含まれています。

一部のレジスタは ROM 内にあるため、コンパイル時に設定する必要があります。これには、いくつかのオプション設定メモリ (OFS) レジスタと、特定のメモリ保護レジスタが含まれます。

オプション設定メモリは、リセット後の MCU の状態を決定します。たとえば、IWDT の設定と有効化、電圧検出の有効化、HOCO 発振の有効化を行うことができます。これらのレジスタが設定されている場合、操作は、MCU のリセットベクトルがフェッチされ、例外が開始される前に完了します。

一部の Synergy MCU にはメモリ保護ユニット (MPU) が搭載されています。MPU は、プログラム可能なデバイスであり、各種メモリ領域について、メモリアクセス権限（たとえば、特権アクセス専用またはフルアクセス）と、メモリ属性（バッファリング可能、キャッシング可能など）を定義するために使用できます。MPU は、最大で 8 個のプログラム可能メモリ領域をサポートでき、それぞれが独自のプログラム可能開始アドレス、サイズ、設定を持ちます。

ISDE は、指定された MPU 設定の値を設定することで、これらのメモリ領域を設定します。これらのレジスタは慎重に設定する必要があります。設定が正しくないと、必要なメモリ領域にアクセスできなくなったり、MCU 全体にアクセスできなくなる可能性があります。

2.3.8 BSP のピン設定

アプリケーションで使用するピンは、ISDE のピンコンフィギュレーターで設定できます。ピンの設定を参照してください。

2.3.9 BSP のクロック設定

すべてのシステムクロックは、BSP の初期化時に、bsp_clock_cfg.h の設定に基づいて設定されます。これらの設定は、ISDE の [Clocks] タブ設定で指定したクロック設定情報が基になります。

- 比較的低速（たとえば 32 kHz) のクロック上で開始される可能性があることから、クロック構成は、起動処理を高速化するため、C ランタイム環境を初期化する前に実行されます。
- BSP は、選択したクロックが安定するために必要な遅延を実装します。

2.3.9.1 bsp_clock_cfg.h

この設定ファイルは、システムクロック設定の値を表します。これは、ISDE の [Clocks] タブで変更可能な設定です。

クロックの設定を参照してください。

2.3.10 システム割り込み

Synergy MCU は、Cortex-M Arm アーキテクチャに基づいているため、ネスト型ベクトル割り込みコントローラ (NVIC) が例外と割り込み設定、優先順位付け、割り込みマスクを処理します。Arm アーキテクチャでは、NVIC が例外を処理します。一部の例外はシステム例外と呼ばれます。システム例外は、ベクトルテーブルの先頭に静的に配置され、ベクトル番号 1 から 15 を占めます。ベクトル 0 は、メインスタックポインタ (MSP) 用に予約されています。残りの 15 個のシステム例外を以下に示します。

- リセット
- NMI
- Cortex-M4 ハード障害ハンドラー
- Cortex-M4 MPU 障害ハンドラー
- Cortex-M4 バス障害ハンドラー
- Cortex-M4 使用率障害ハンドラー
- 予約
- 予約
- 予約
- 予約
- Cortex-M4 SVCall ハンドラー
- Cortex-M4 デバッグモニターハンドラー
- 予約
- Cortex-M4 PendSV ハンドラー
- Cortex-M4 SysTick ハンドラー

NMI とハード障害例外は、リセット以降に有効になっており、優先度は固定です。その他の例外の優先度は設定可能であり、一部は無効にできます。

2.3.11 グループ割り込み

グループ割り込みは、マスク不可能な割り込み (NMI) をトリガできる 12 のソースを表すために使用される用語です。NMI が発生した場合、NMI ハンドラーは NMISR（ステータスレジスタ）を調べ、割り込みのソースを判定します。NMI 割り込みはすべての割り込みより優先され、CPU 割り込みとしてのみ使用でき、データトランスファコントローラ (DTC) またはダイレクトメモリアクセスコントローラ (DMAC) のアクティブ化を行うことはできません。

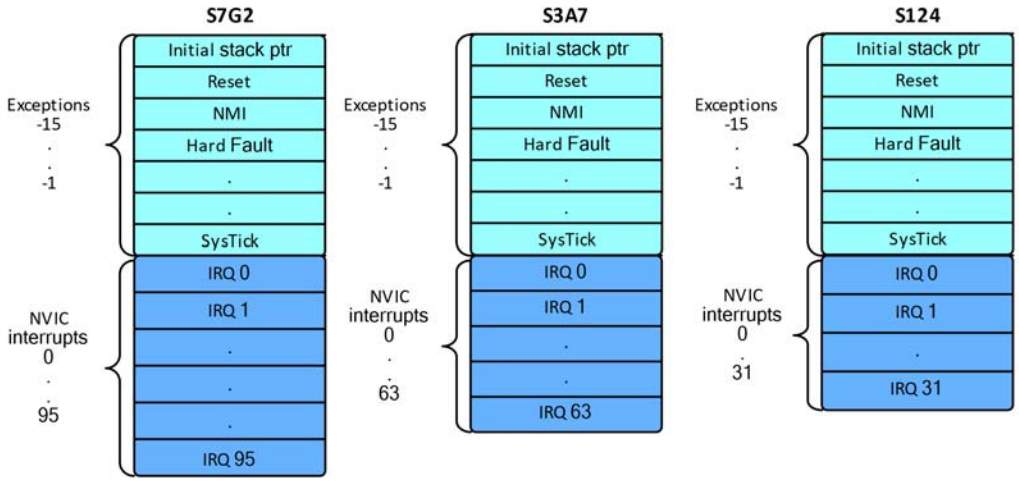
グループ割り込みのソースとしては、以下のものが考えられます。

- IWDT アンダーフロー / 更新エラー
- WDT アンダーフロー / 更新エラー
- 電圧監視 1 割り込み
- 電圧監視 2 割り込み
- VBATT 監視割り込み
- 発振停止検出
- NMI ピン
- RAM パリティエラー
- RAM ECC エラー
- MPU バススレーブエラー
- MPU バスマスターエラー
- MPU スタックエラー

ユーザーは、BSP API 関数 R_BSP_GroupIrqWrite を使用してコールバックを登録することで、1 つ以上のグループ割り込みの通知を有効にすることができます。NMI 割り込みが発生すると、NMI ハンドラーは、割り込み原因に対してコールバックが登録されているかどうかを確認し、登録されている場合はそのコールバック関数を呼び出します。

前述のように、ベクトルテーブルの最初の 16 個のスロットはすでにシステム例外に割り当てられています。スロット 16 以降はユーザーが設定可能な割り込みです。外部割り込みか、ペリフェラルで生成される割り込みを設定できます。

NVIC 割り込みテーブルのサイズは、以下のように Synergy MCU の種類によって異なります。



NVIC 割り込みベクトルテーブルの空きスロットの数は少なく見えるかもしれませんが、BSP では、割り込みを生成できるイベントが最大で 512 個定義されます。BSP は、イベントマッピングを使用することで、ユーザーが有効にしたイベントを NVIC 割り込みにマッピングします。S7G2 MCU では、これらのイベントのうち 96 個だけを一度にアクティブにできますが、ユーザーは、アクティブなイベントを生成するイベントを柔軟に選択できます。

下図は S7G2 の割り込みベクトルテーブルを示しています。

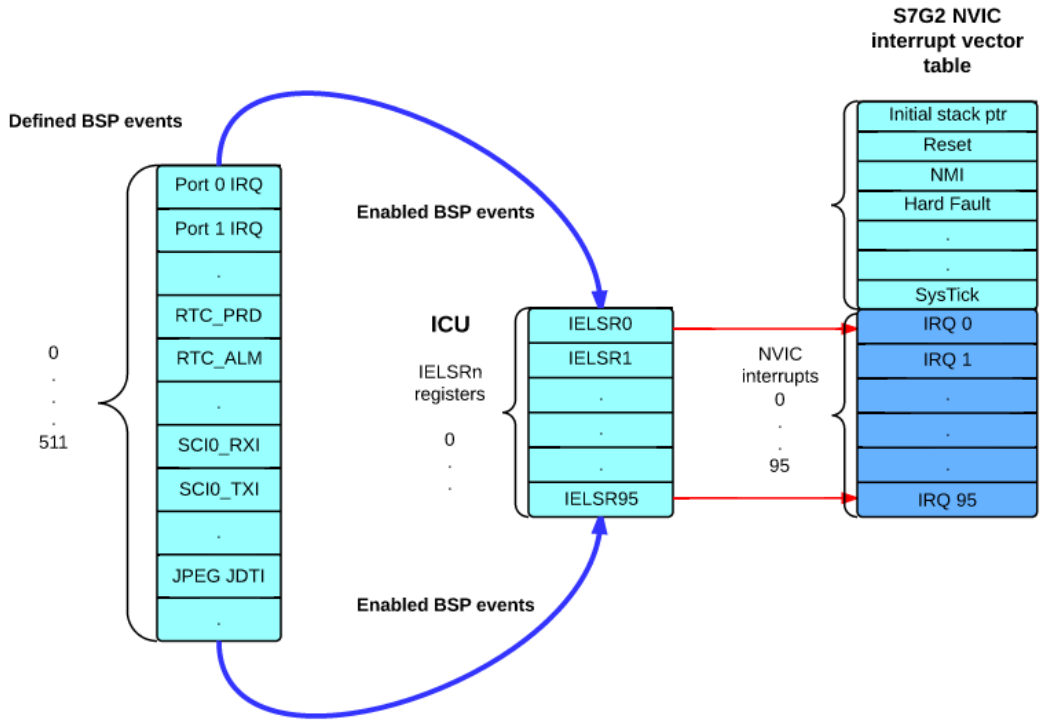


図 14: NVIC 割り込みベクターテーブル

割り込み要因として関心があるイベントのみをユーザーが選択できるようにすることで、高速でイベント固有の割り込みサービスルーチンを提供できます。

たとえば、他のマイクロコントローラにおいて、標準の NVIC 割り込みベクトルテーブルには、SCI0（シリアル通信インタフェース）用の単一のベクトルエントリが格納される可能性があります。そのための割り込みサービスルーチンは、割り込みの「本当の」ソースのステータスレジスタを確認する必要があります。Synergy の実装では、関心がある SCI0 イベントごとに 1 つのベクトルエントリがあります。標準の NVIC テーブルと Synergy S7G2 NVIC テーブルの違いを下に示します。

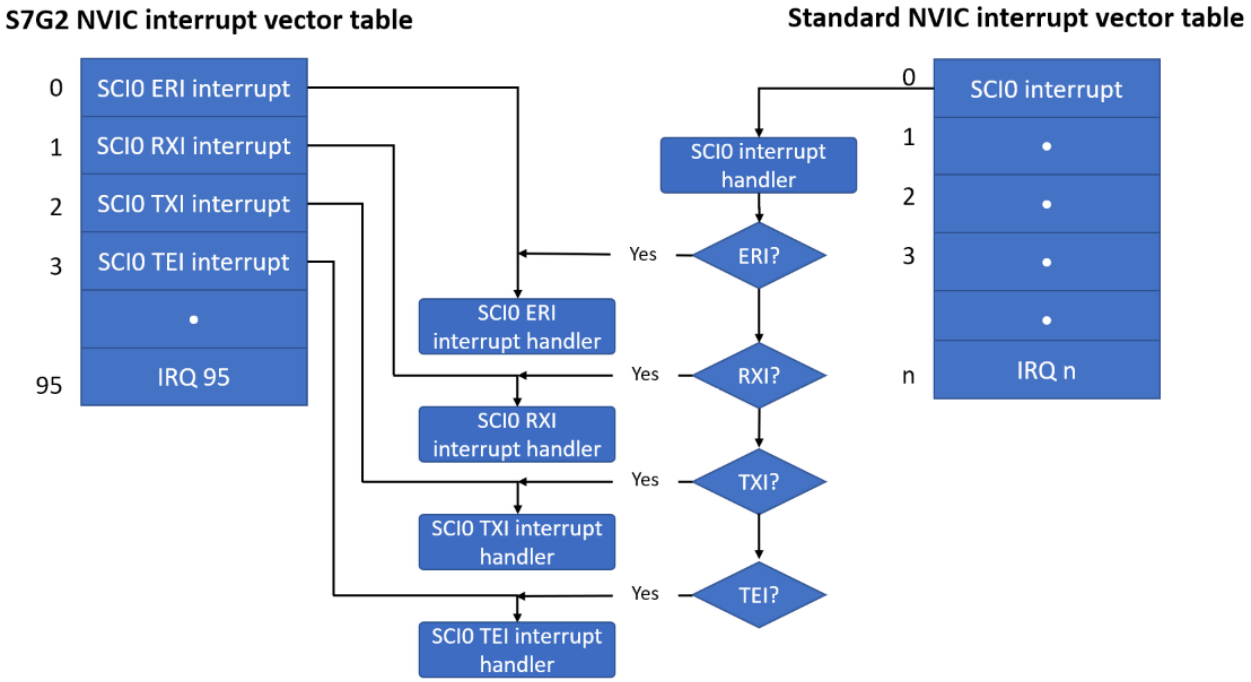


図 15: NVIC 割り込みベクトルテーブルの例

割り込みの構成は統合ソリューション開発環境によって処理されます。モジュールにより使用される割り込みを選択すると、ベクトルテーブルのエントリを割り当てるのに必要なコードが生成され、適切な ICU ELC イベントにリンクされます。

割り込みが発生した場合に、ごく初期に行う必要がある処理の 1 つは、BSP によって割り当てられた NVIC 割り込みスロットに対応する割り込み番号と共に R_BSP_IrqStatusClear をコールすることです。

R_BSP_IrqStatusClear は、特定の割り込みの割り込みステータスフラグ (IR) をクリアします。特定の割り込みの割り込みステータスフラグ (IR) をクリアします。割り込みがトリガされると、IR ビットがセットされます。

ここに示す例で、優先度が割り当てられたエントリ（たとえば BSP_IRQ_CFG_ICU_IRQ0) では、対応する weak ハンドラーのアドレスが、次に使用可能なベクトルスロットに格納されます。可能性のあるすべて・フ割り込みソースがこのようにして反復処理されます。定義されている割り込みがベクトルテーブルに設定されます。

2.3.11.1 bsp_irq_cfg.h

現在では使用しておらず、将来的に削除される予定のレガシーファイルです。割り込みの構成は現在、全面的に統合ソリューション開発環境によって処理されています。

2.3.11.2 ベクトルテーブルエントリ

HardFault_Handler, などのシステム例外は、弱い参照として定義されます。このため、ユーザー独自のハンドラを定義し、特定の例外についてデフォルトハンドラをオーバーライドすることができます。

ベクトルテーブル内の他のエントリはすべて、ISDE で生成されます。その際、ベクトルとそれに対応するベクトル情報構造体を定義するマクロによって、ROM テーブルリンカセクションにエントリが生成されます（ベクトルには .vector.*、ベクトル情報には .vector_info.*）。

CMSIS の system_xxxx.c では、ウォームスタートコールバック関数 R_BSP_WarmStart() の weak 定義（と関数本体）もあることに注意してください。この関数は、weak 宣言と同じファイルに定義されているため、「デフォルトの」実装として呼び出されます。本体をユーザーアプリケーションにコピーし、必要に応じて変更することで、関数をオーバーライドできます。リンカーはこれを「strong」参照として認識して使用します。

2.3.11.3 ウォームスタートコールバック

BSP がボードをリセット状態から起動している最中に、ユーザーがコールバックを要求できる場所が 2 つあります。これらは、「Pre C」および「Post C」ウォームスタートコールバックとして定義されます。

前述のように、この関数は R_BSP_WarmStart() としてすでに weak 定義されているため、アプリケーションコードで関数を再定義する（または、CMSIS の system_xxxx.c から既存の本文をコピーする）だけで、コールバックを作成できます。R_BSP_Warmstart() は、実行中のウォームスタートコールバックの種類を示すイベントパラメータを受け取ります。

```
/** Different warm start entry locations in the BSP. */

typedef enum e_bsp_warm_start_event

{

    BSP_WARM_START_PRE_C = 0, // Called almost immediately after reset.

    /* No C runtime environment, clocks, or IRQs. */

    BSP_WARM_START_POST_C // Called after clocks and C runtime environment have
been set up.

} bsp_warm_start_event_t;
```

この関数は、有効 / 無効にされず、BSP のスタートアップの一部として、両方のイベントに対して必ず呼び出されます。そのため、ユーザーがオーバーライドしている場合にはコールされない関数本体が必要です。関数の本体は system_xxxx にあります。この関数を使用するには、この関数をユーザーのコードにコピーし、要件を満たすように変更します。

2.3.11.4 Pre C ウォームスタートコールバック

このコールバックはリセット後ほぼすぐに呼び出されます。この時点では、C ランタイム環境、クロック、IRQ がセットアップされていません。

「Pre C」ウォームスタートコールバックにユーザーが関心を持つ理由

以下にいくつかの例を挙げます。

- スタートアップ処理の一部としてのセーフティコード（たとえば破壊的なメモリテスト）の実行。
- クラッシュダンプ調査の一環としてのグローバルメモリの検査。

- すでに実行している RTC の再初期化の防止。

2.3.11.5 Post C ウォームスタートコールバック

このコールバックは、クロックと C ランタイム環境がセットアップされた後で呼び出されます。

「Post C」ウォームスタートコールバックにユーザーが関心を持つ理由

以下にいくつかの例を挙げます。

- クロックがセットアップされている必要のあるテストの実行。
- ADC の診断。
- ROM/ 外部メモリシステムのチェック。

2.3.12 カスタム BSP ボードサポート

いずれかの Synergy MCU に基づいて独自のボードを開発している場合はどうしますか。

以前のバージョンでは、外部コマンドラインユーティリティの Custom BSP Creator ツールがあり、これを使用して、e^2 studio 内からカスタム BSP を作成することができました。カスタム BSP の作成は現在、統合ソリューション開発環境内から行うことができます。これについては「統合ソリューション開発環境使用上の注意」(3 章) で説明します。

2.3.13 BSP API 関数

BSP にはパブリック関数があり、BSP を使用するすべてのプロジェクトで使用できます。この関数を使用すると、BSP でサポートされる MCU とボード全体で共通する機能にアクセスできます。

- R_BSP_SoftwareLockInit: BSP は、アトミックロックを実装するための API 関数を提供しています。これらのロックは、セマフォやミューテックスなどのコードの重要な領域を守るために使用できます。この関数は、単に定義されたロック構造体を BSP_LOCK_UNLOCKED に初期化します。
- R_BSP_SoftwareLock: 渡されたロックの取得を試みます。排他的ロード命令と排他的ストア命令は、排他的読み取り / 変更 / 書き込みを、入力ロックに対して実行するために使用されます。渡されたロックの取得を試みます。
  - 排他的ロード命令と排他的ストア命令は、排他的読み取り / 変更 / 書き込みを、入力ロックに対して実行するために使用されます。
  - この処理は以下のようになります。排他的ロード (LDREXB) を使用してロックの値を読み取ります。
  - ロックを使用できる場合は、ロック値を変更して確保します。
  - 返されたステータスビットをテストし、書き込みが実行されたかどうかを判断します。
- R_BSP_SoftwareUnlock: 既存のソフトウェアロックを解放します。
- R_BSP_HardwareLock: ハードウェアロックはソフトウェアロックに似ています。: 既存のソフトウェアロックを解放します。: ハードウェアロックはソフトウェアロックに似ています。実際に、BSP のソフトウェアロック関数は、ハードウェアロック関数によって呼び出されます。たとえば、フラッシュ API

の open() 関数が呼び出された場合、フラッシュハードウェアのロックを取得して、フラッシュ API の close が呼び出されるまでロックを保持します。

- R_BSP_HardwareUnlock: 既存のハードウェアロックを解放します。上のフラッシュの例で、フラッシュ API の close 関数がこの関数を呼び出すことが考えられます。
- R_BSP_GroupIrqWrite: サポートされるいずれかのグループ割り込みに対し、コールバック関数を登録します。サポートされるいずれかのグループ割り込みに対し、コールバック関数を登録します。前述のように、割り込みは 12 個あり、すべて NMI 例外にマッピングされています。NMI が発生した場合、NMI_Handler が NMISR（ステータスレジスタ）を参照し、割り込みのソースを決定します。コールバック引数に NULL が渡された場合は、過去に登録されたコールバックが登録解除されます。
- R_BSP_IrqStatusClear: 特定の割り込みの割り込みステータスフラグ (IR) をクリアします。: 特定の割り込みの割り込みステータスフラグ (IR) をクリアします。割り込みがトリガされると、IR ビットがセットされます。
- R_BSP_SoftwareDelay: ブロッキングソフトウェア遅延を実装します。: ブロッキングソフトウェア遅延を実装します。遅延は、システムクロックレートに基づいて実装されます。
- R_BSP_VersionGet: BSP のバージョンを返します。
- R_BSP_LedsGet: ボード上の LED に関する情報を返します。
- R_BSP_ModuleStop : ストップビットを設定する必要のあるモジュールを指定します。
- R_BSP_ModuleStart : ストップビットをクリアする必要のあるモジュールを指定します。
- R_BSP_CacheOff(): ROM キャッシュをオフにして、以前の状態に戻します。
- R_BSP_CacheSet(): キャッシュを特定の状態（オンまたはオフ）に設定します。
- R_BSP_RegisterProtectEnable: レジスタ保護を有効にします。保護されたレジスタに書き込むことはできません。レジスタ保護は、保護レジスタ (PRCR) と MPC の書き込み保護レジスタ (PWPR) を使用することで可能になります。保護が可能なレジスタは、3 つのグループのいずれかにグループ化されます。
  - BSP_REG_PROTECT_CGC - クロック生成回路に関連するレジスタ。
  - BSP_REG_PROTECT_OM_LPC_BATT - 動作モード、低電力消費、バッテリーバックアップ機能に関連するレジスタ。
  - BSP_REG_PROTECT_LVD – LVD（低電圧検出）に関連するレジスタ。

BSP レジスタ保護機能は、参照カウンタを利用して、特定のレジスタを指定した後で別の関数を呼び出すアプリケーションで、そのレジスタ保護設定が誤って変更されないようにします。

- RegisterProtectDisable() が呼び出されるたびに、それぞれの参照カウンタがインクリメントされます。
- RegisterProtectEnable() が呼び出されるたびに、それぞれの参照カウンタがデクリメントされます。

どちらの関数も、参照カウンタがゼロの場合は、保護状態のみを変更します。

以下の例に示すように、参照カウンタがないと、MODULE1 が保護解除したレジスターを MODULE2 が再度保護し、MODULE1 が書き込めなくなります。

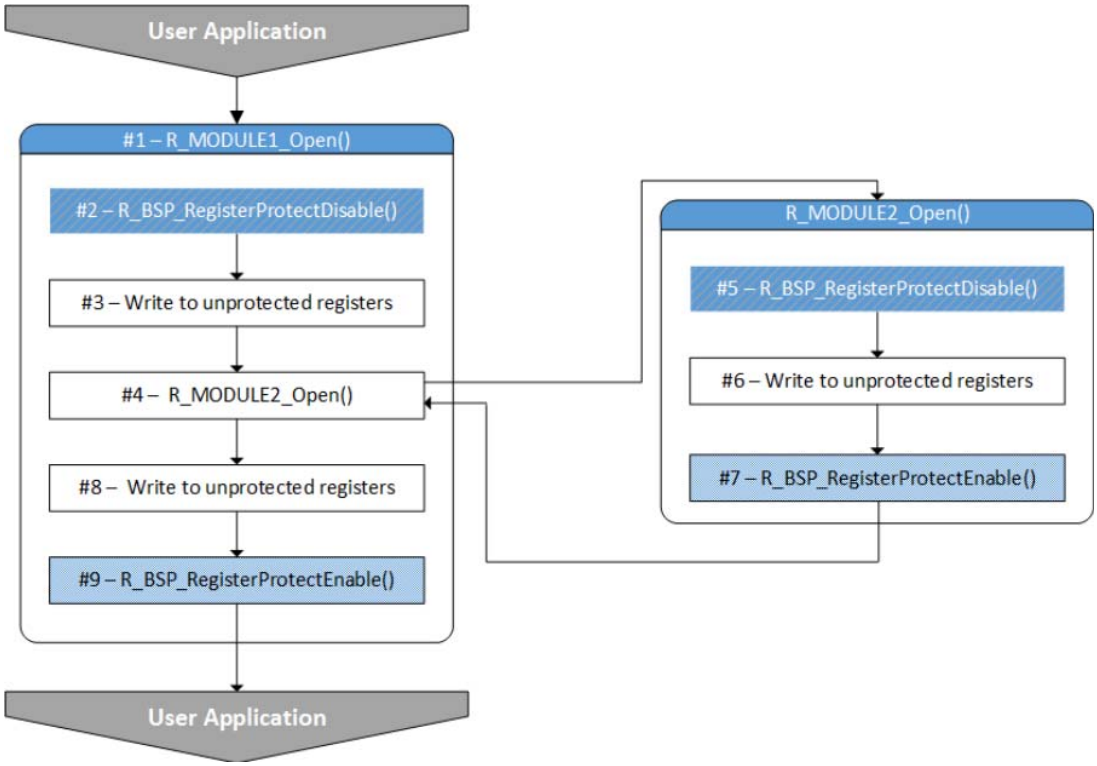


図 16: レジスタ保護

- R_BSP_RegisterProtectDisable: レジスタ保護を無効にします。保護されていないレジスタに書き込むことができます。レジスタ保護は、保護レジスタ (PRCR) と MPC の書き込み保護レジスタ (PWPR) を使用して無効にします。ここでも上記のレジスタのグループ化が適用されます。

第 3 章　開発の開始

Renesas Synergy Software Package (SSP) を使用して開発を開始するには、e2 studio をダウンロードして、対象となる Synergy 開発と評価ボードを入手し、この章にあるチュートリアルに従って実行します。e2 studio ISDE ユーザーガイドとチュートリアルには、簡単なアプリケーションから始められる詳細なインストラクションが含まれています。SSP を始めるには、これらのページを参照してください。

- e2 studio ISDE ユーザーガイド
- チュートリアル : Your First Synergy Project - Blinky
- チュートリアル : Using HAL Drivers - Programming the WDT
- IAR Embedded Workbench for Renesas Synergy
- Synergy Standalone Configurator（SSC）とは

3.1 e2 studio ISDE ユーザーガイド

3.1.1 e2 studio ISDE の使用

ここでは、e2 studio ISDE（統合ソリューション開発環境）を使用して、Renesas Synergy Software Package (SSP) を使用したアプリケーションを作成する方法について説明します。SSP のアーキテクチャは、e2 studio ISDE をどのように使用して Synergy アプリケーションを開発するかを直接決定します。このマニュアルに含まれる SSP アーキテクチャの詳細については、次のドキュメントを参照してください。

- SSP アーキテクチャ
- BSP アーキテクチャ

e2 studio で生成および構築された簡単なサンプルプロジェクトについては、次を参照してください :

- チュートリアル : Using Hal Drivers - Programming the WDT
- チュートリアル : Your First Synergy Project - Blinky

このマニュアルのすべてのユーザーガイドでは、e2 studio ISDE を使用してドライバーを構成し、アプリケーションを開発する方法について説明しています。以下を参照してください。

- HAL レイヤー : HAL レイヤーユーザーガイド
- フレームワークレイヤー : フレームワークレイヤーユーザーガイド

e2 studio の幅広いヘルプコンテンツも参照してください。[Help] > [Help Contents] をクリックしてアクセスできます。

3.1.2 e2 studio ISDE の概要

e2 studio ISDE（統合ソリューション開発環境）は、コードの開発、ビルド、デバッグを包含した開発ツールです。ISDE はオープンソース Eclipse IDE および関連する C/C++ Development Tooling(CDT) に基づいています。特に Synergy MCU では、Synergy プロジェクト用に、Synergy Software Package (SSP) を使用してコードを設定および自動生成するための、いくつかのグラフィカルユーザーインタフェース (GUI) ウィザードが ISDE に備わっています。ISDE にはスマートマニュアルも取り込まれており、ドライバーとデバイスのドキュメントが、コード中でツールヒントの形で利用できます。

図 17: e2 studio の開始画面

e2 studio 統合ソリューション開発環境と Synergy Project Editor のコンフィギュレータは、特定のアプリケーションに必要な SSP モジュールをできるだけ簡単かつ迅速に選択し、プロジェクトに追加して構成できるように開発されています。ISDE は、Synergy MCU アプリケーション用に SSP のすべての要素を設定するグラフィカルユーザーインタフェースを提供しています。ISDE は、HAL モジュールとフレームワークモジュールに加えて、RTOS スレッド、セマフォ、ミューテックス、イベントフラグ、キューを追加および設定できます。これにより、RTOS のサポートをアプリケーションに非常に簡単に追加できます。プロジェクトが生成されたら、必要に応じて任意のモジュールと設定値を再設定できます。ISDE は構成ビューの選択から完全かつ正確な構成コードを生成するため、アプリケーションコードの記述に専念することができます。

プロジェクトの一部となるSSPの要素は、e2 studio統合ソリューション開発環境に含まれているSynergy Project Editor の [Threads] タブに表示されます。SSP の構成構造とパラメータはすべて XML ファイルにマッピングされます。統合ソリューション開発環境ではこの XML ファイルにより、[Properties] ビューに構成可能なオプションの視覚的リストを提示できます。モジュールを設定するためのコードを生成するのに加えて、XML はモジュールの依存関係情報も提供します。

プロジェクトに SSP モジュールを追加すると、e2 studio 統合ソリューション開発環境はそのモジュールの依存関係を確認し、すべての必要なドライバーとフレームワークモジュールを追加して、適切なソフトウェア

スタックを作成します。ユーザーによる選択が必要な依存関係がある場合は、[Stacks] ペインでそのモジュールがハイライトされ、選択可能なオプションが統合ソリューション開発環境により表示されます。

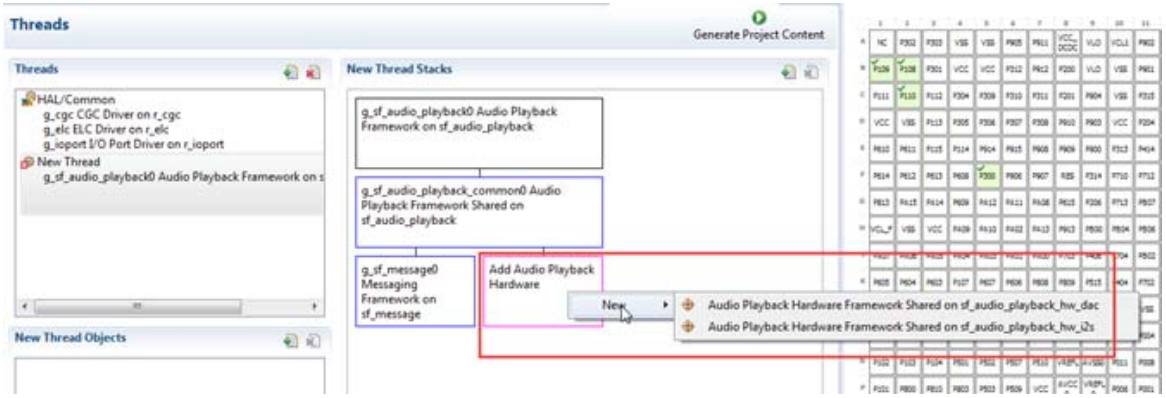


図 18: ISDE の依存関係チェック

エラーは、HAL/ 共通モジュールまたは新しいスレッドモジュールペインのドライバー名の横にフラグされます。また、[Problems] ビューでエラーを確認できます。

3.1.3 e2 studio ISDE の前提条件

3.1.3.1 Synergy キットの取得

SSP でアプリケーションを開発するには、いずれかの Renesas Synergy キットでスタートします。Renesas Synergy キットは e2 studio ISDE とシームレスに統合するよう設計されています。いくつかの種類のキットが提供されています。

- 開発キット（DK)
- スターターキット（SK）
- 製品サンプル（PE)
- ターゲットボード（TB）

すべての Synergy キットに関する注文情報、クイックスタートガイド、ユーザーマニュアル、その他のドキュメントが http://renesassynergy.com で入手できます。

3.1.3.2 PC 要件

e2 studio ISDE を使用するには、ご使用のパソコンが次の最小要件を満たしていることを確認してください。

- Windows 7 または Windows 10 と Intel i5 または i7、あるいは AMD A10-7850K または FX
- メモリ : 8 GB DDR3 または DDR4 DRAM（16 GB DDR4/2400 MHz RAM を推奨）
- 最低 250 GB の空きハードディスク容量

3.1.3.3 e2 studio および SSP のインストール

e2 studio ISDE および SSP の詳細なインストール手順とインストーラーは、Renesas Synergy Gallery ウェブサイト https://synergygallery.renesas.com. にあります。e2 studio のリリースノートをレビューして、e2 studio のバージョンが選択された SSP バージョンをサポートしていることを確認します。

関連項目 :SSP リリースの処理

3.1.3.4 ツールチェーンの選択

e2 studio ISDE は、GNU Arm コンパイラや IAR ツールチェーンなど、いくつかのツールチェーンとツールチェーンバージョンと連携できます。特定のバージョンの GNU Arm コンパイラが e2 studio インストーラに含まれており、その SSP バージョンで実行できることが検証されています。

Arm 用 IAR ツールチェーンと e2 studio を連携させるには、IAR Embedded Workbench for Renesas Synergy（IAR EW for Synergy）をインストールします（Synergy Gallery から入手できます。ただし、IAR によるライセンス供与が必要です）。IAR で Synergy プロジェクトを始める前に、必ず IAR Embedded Workbench for Arm Eclipse プラグインをインストールします（[Help] メニューの IAR Embedded Workbench プラグインマネージャーを使用）。

3.1.3.5 SSP ライセンス設定

デフォルトで、SSP ダウンロードには SSP の評価ライセンスが含まれます。Synergy Project の構成中にライセンスファイルを指定するようプロンプトされた場合は、インストール手順のライセンスに関する説明を参照してください。評価ライセンスファイルは以下のディレクトリにあります :<e2_studio_base_dir>/internal/projectgen/arm/Licenses/

評価ライセンスで、完全に機能する SSP バージョンを使用することができます。これは、SSP のすべてのモジュールをコンパイルして、アプリケーションにリンクできることを意味します。ただし、フレームワークレイヤーのモジュールのソースコードは暗号化されており、変更できません（Express Logic X-Ware コンポーネントを含む）。この場合でも、選択したモジュールのソースコードを表示させることができます。このタイプの暗号化コードは保護コードと呼ばれます。詳細は、ライセンスファイルを参照してください。

HAL レイヤーのモジュールのソースコードは保護されておらず、ssp/src/<module>/<module>.c ファイルをクリックして e2 studio ISDE から表示できます。

他のライセンスタイプでは、統合ソリューション開発環境のデバッグフェーズ中にセキュアデバッグビューで保護コードを表示できるか、選択されたソースに対して完全なリードまたはライトアクセスを行うことができます。

下の表では、利用可能なライセンスと機能を示しています。

License Type	利用可能	利用不可
Evaluation License	コンパイル / リンク	保護ソースの保存 / 編集。すべてのモジュールを見ることはできません。詳細については、ライセンスファイルを参照してください。

License Type	利用可能	利用不可
Development/Production License	Secure Viewer での保護ソースのコンパイル / リンクおよび表示	保護ソースの保存 / 編集
Source License	Secure Viewer での保護ソースのコンパイル / リンクおよび表示。選択したソース（ソースライセンスによる）は平文のテキストファイルとして編集および保存できます。	ソースライセンスに含まれない保護ソースファイルの保存 / 編集

3.1.3.6 IAR Embedded Workbench for Renesas Synergy Compiler を e2 studio に追加する

e2 studio で IAR Embedded Workbench for Renesas Synergy Compiler（IAR コンパイラ）を使用できます。IAR EW for Synergy を使用せずに IAR コンパイラを使用できるため、開発者にとってメリットとなります。インストール手順には、IAR EW for Synergy、e2 studio、および IAR プラグインの各種インストールのプロセスが含まれています。インストール手順についてはアプリケーションノートをご覧ください。
https://www.renesas.com/en-us/search/keyword-search.html#q=R11AN0272

また、このアプリケーションノートには、プロジェクトのマイグレーション方法についても説明しています。e2 studio を使用する際に確実にプロジェクトファイルを開くことができるように、IAR 7.x から IAR 8.x へのプロジェクトのマイグレーションについての説明も含まれています。

3.1.4 Synergy プロジェクトとは

e2 studio では、すべての SSP アプリケーションは Synergy プロジェクトに編成されます。Synergy プロジェクトの設定には以下の部分があります。

1) プロジェクトの作成
2) プロジェクトの設定

e2 studio ISDE には、Synergy プロジェクト専用の多数のプロジェクトウィザードおよび構成ウィンドウがあります。Synergy Project Generator を使って新しい Synergy プロジェクトを作成することも、Synergy Project Editor を使って既存のプロジェクトの構成を編集することもできます。

e2 studio を起動してワークスペースを選択すると、選択したワークスペースで以前に保存したすべてのプロジェクトが読み込まれ、[Project Explorer] ビューに表示されます。各プロジェクトには、configuration.xml という関連する構成ファイルがあり、プロジェクトのルートディレクトリに置かれています。

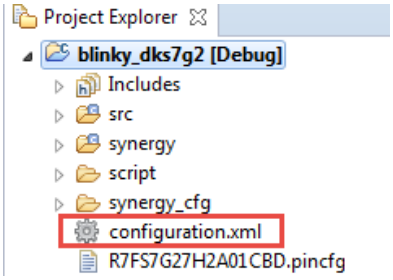


図 19: e2 studio のプロジェクト構成ファイル

configuration.xml ファイルをダブルクリックすると Synergy Project Editor が開き、このプロジェクトに関連する構成設定を表示または変更できます。プロジェクト構成を編集するには、Synergy Configuration パースペクティブが e2 studio ウィンドウの右上で選択されていることを確認してください。

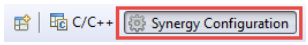


図 20: e2 studio Synergy Configuration パースペクティブ

注 : Synergy プロジェクト構成（つまり、configuration.xml ファイル）を保存すると、毎回、すべてのプロジェクト設定を含む詳細な Synergy プロジェクトレポートファイル (synergy_cfg.txt) が生成されます。ファイルの差分は、テキスト差分表示ツールを使用して簡単に確認することができます。生成されたファイルはプロジェクトのルートディレクトリに配置されます。Synergy Project Editor にはいくつかのタブがあります。個別のタブの構成ステップとオプションについては、続くセクションで取り上げます。

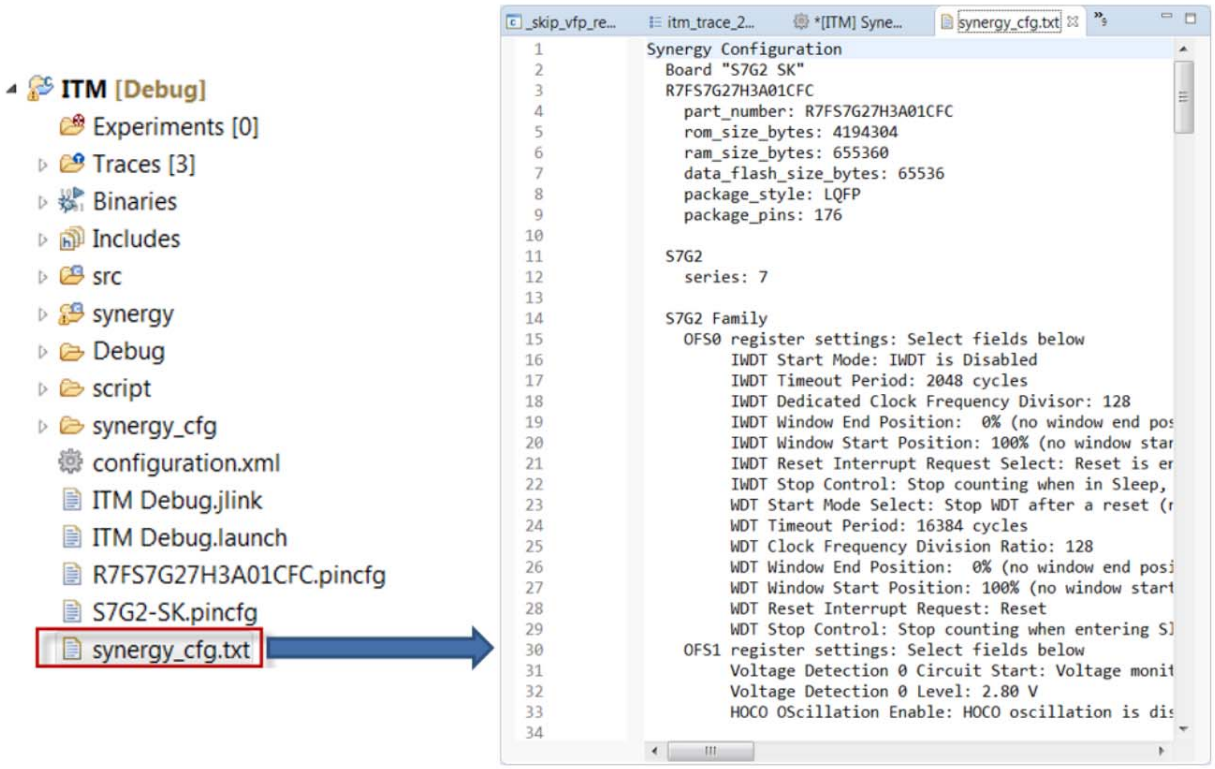


図 21: Synergy プロジェクトレポート

Synergy Project Editor にはいくつかのタブがあります。個別のタブの構成ステップとオプションについては、続くセクションで取り上げます。

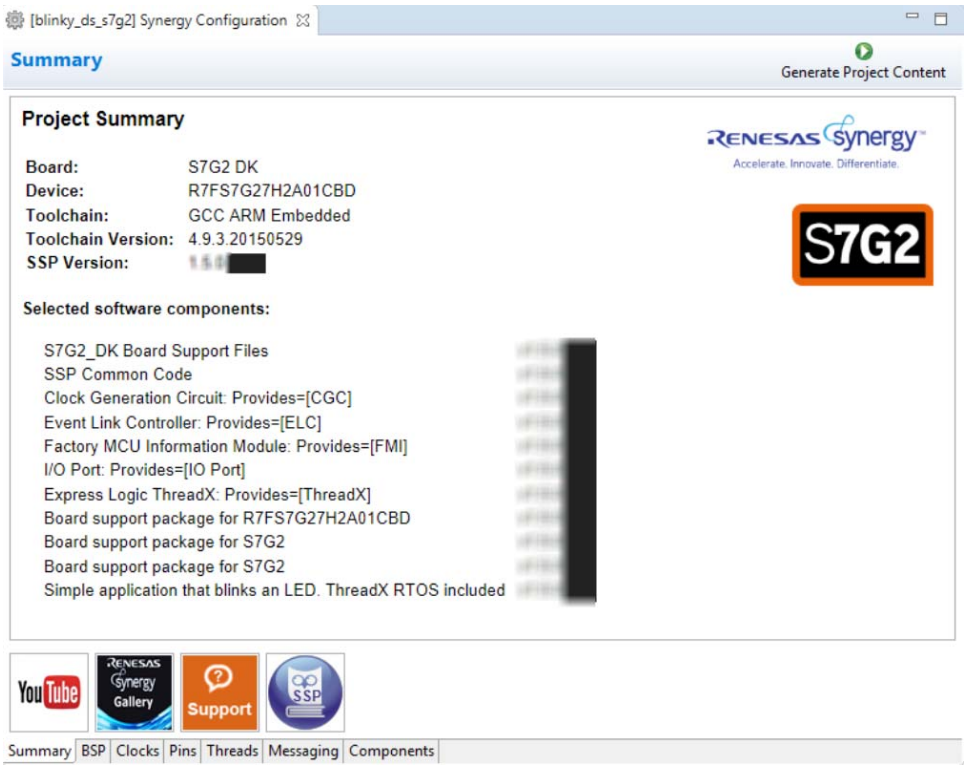


図 22: e2 studio Project Editor

3.1.5 Synergy プロジェクトの作成

このセクションでは、Synergy C または C++ プロジェクトを作成する詳細な手順について説明します。プロジェクトを作成すると、ハードウェア（クロック、ピン、割り込みなど）とアプリケーションの一部であるすべての SSP モジュールのパラメータを簡単に構成できます。

Synergy Project Generator で新しい Synergy C または C++ プロジェクトを作成するには、プロジェクト名を指定し、アプリケーション用にハードウェアを選択し、ライセンスの確認または追加を行い、ツールチェーンを選択し、さらにプロジェクトテンプレートの選択により事前構成されたクロック、ピン、MCU 関連の設定を選びます。

3.1.5.1 新規プロジェクトの作成

Synergy アプリケーションでは、次の方法で新しいプロジェクトを Synergy プロジェクトとして生成します。

1) 新規の Synergy C プロジェクトを作成する場合には、[File] > [New] > [Synergy C/CC++ Project] をクリック後、[Renesas Synergy C Executable Project] を選択します。新規の Synergy C++ プロジェクトを作成す

る場合には、[File] > [New] > [Synergy C/CC++ Project] をクリック後、[Renesas Synergy C++ Executable Project] を選択します。

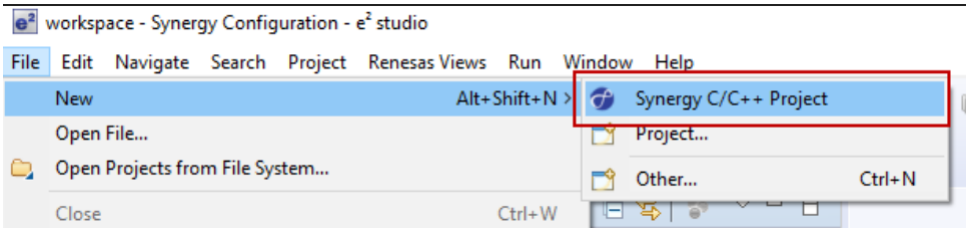


図 23: New Synergy Project

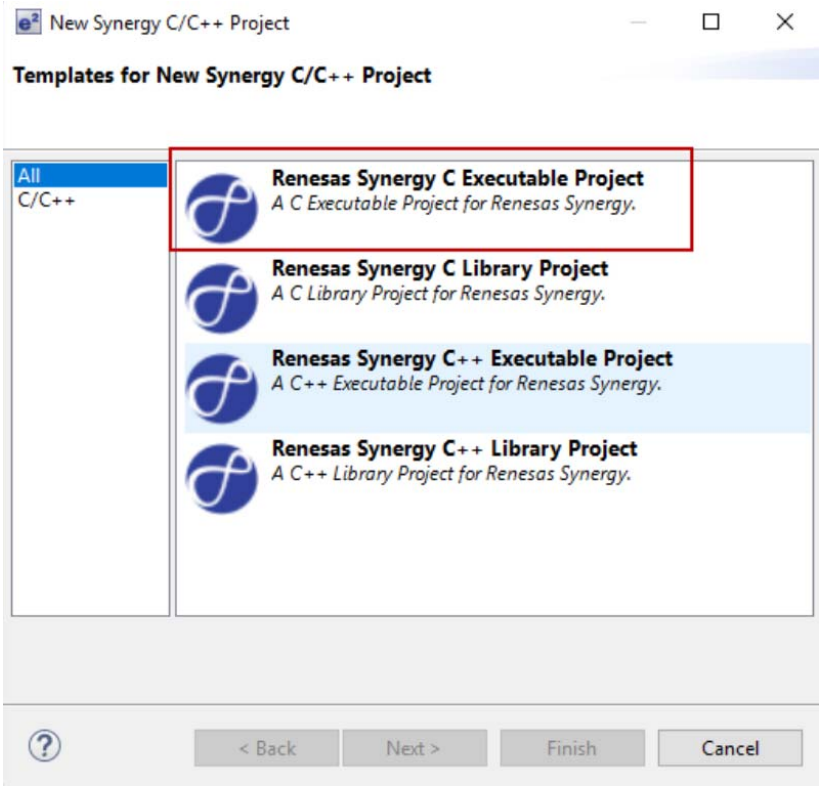


図 24: New Synergy Template

2) プロジェクト名と場所を選択します。

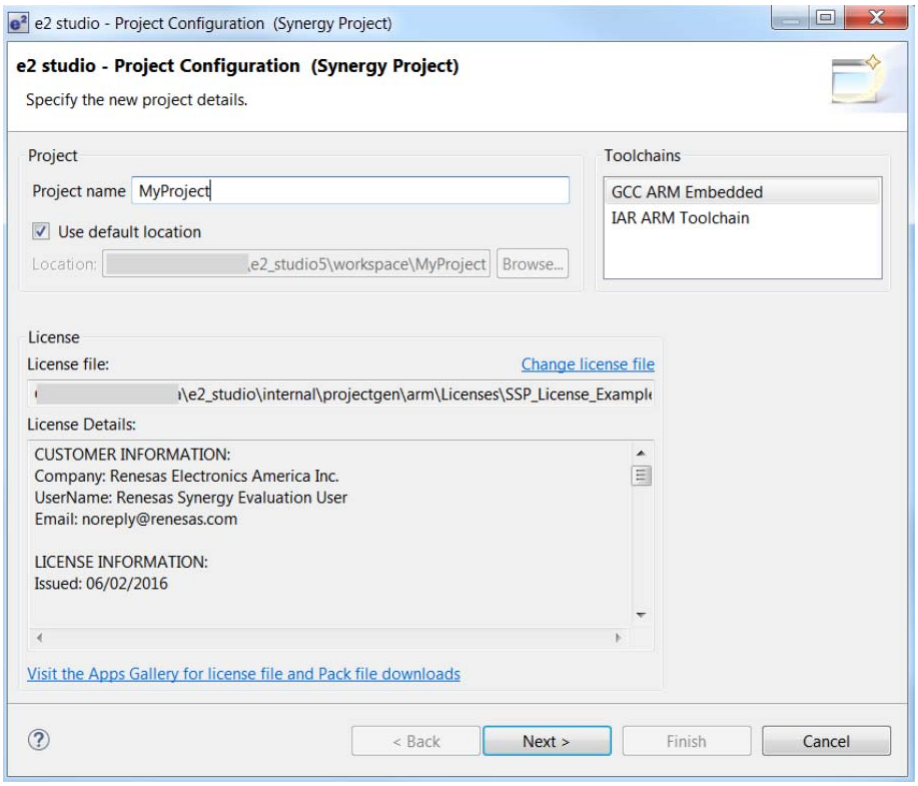


図 25: Synergy Project Generator（画面 1）

3) 最新の SSP のライセンスファイルがあることを確認します。SSP ダウンロードパックには評価ライセンスが含まれていますが、開発・生産ライセンスやソースライセンスなども利用可能です。ライセンスファイルの読み込み方法については、Synergy ウェブサイトにある『Installation Guide』を参照してください。e2 studio および SSP を初めてインストールした場合、ライセンスペインは空です。それ以外の場合、ライセンスペインにはインストールした最新のライセンスが表示されます。複数のライセンスがある場合は、いずれかを選択できます。ライセンスファイルは以下のディレクトリにあります：<e2_studio_base_dir>/internal/projectgen/arm/Licenses/

4) **[Next]** をクリックします。

3.1.5.2 ボードとツールチェーンの選択

次の [Project Configuration] ウィンドウで、ハードウェアとソフトウェアの環境を選択します。

1) プロジェクトで使いたい SSP のバージョンを選択します。

2) アプリケーションのボードを選択します。既存の Synergy キットを選択するか、独自の BSP 定義を持つ任意の Synergy デバイスの「カスタムユーザーボード」を選択することができます。

注 : 独自のカスタムボードパックを作成するには、カスタムボードパック（「カスタム BSP」）の作成の章を参照してください。

3) ツールチェーンのバージョンを選択します。これは GCC ツールチェーンでのみ利用できます。

4) デバッガを選択します。J-Link Arm デバッガーがあらかじめ選択されています。

5) [Next] をクリックします。

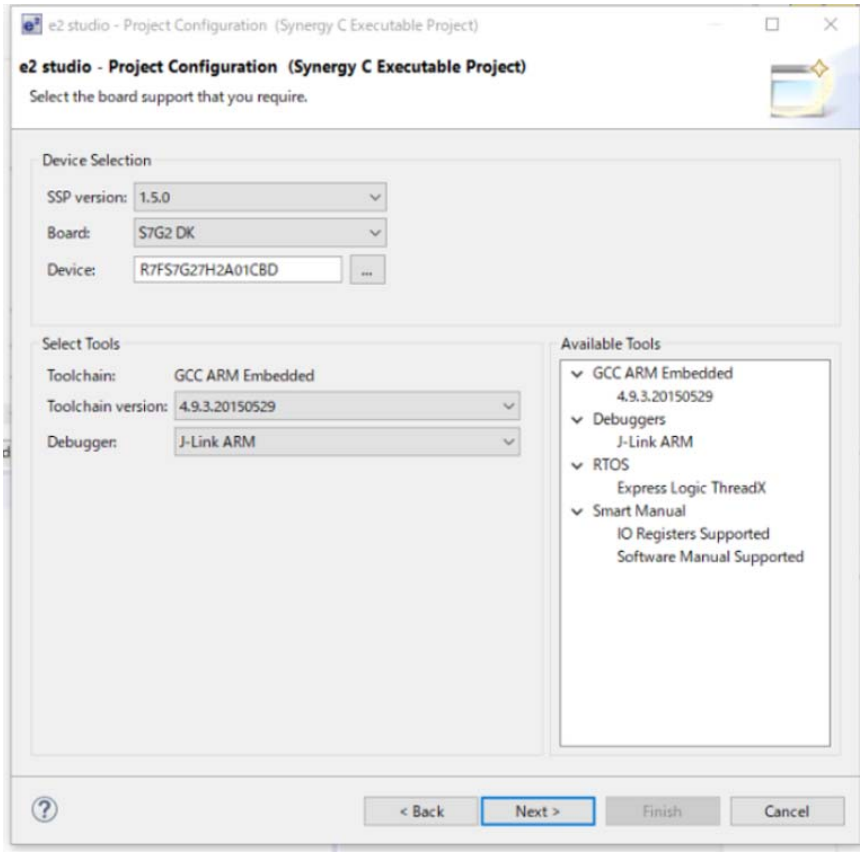


図 26: Synergy Project Generator（画面 2）

3.1.5.3 プロジェクトテンプレートの選択

続くウィンドウで、利用可能なテンプレートのリストからプロジェクトテンプレートを選択して、[Finish] をクリックします。

注 : 独自のアプリケーションを開発する場合、「BSP」など、使用するボードの基本テンプレートを選択します。プロジェクトのモジュールを構成する際にいつでも RTOS サポートを追加できます。

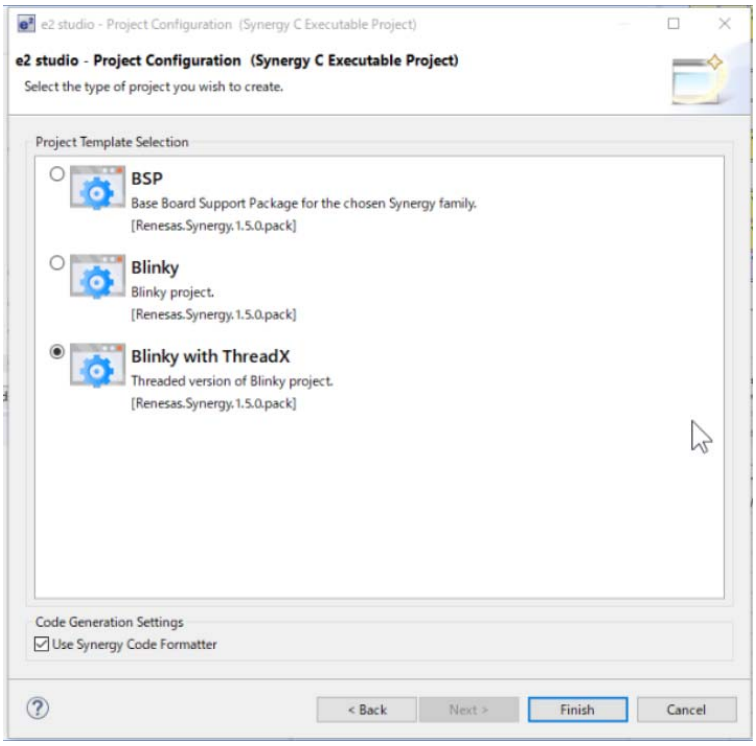


図 27: Synergy Project Generator（画面 3）

デフォルトでは、この画面には現在の SSP パックに含まれるテンプレートが表示されます。

プロジェクトが作成されると、ISDE は Synergy Project Editor で現在のプロジェクト構成の概要を表示します。

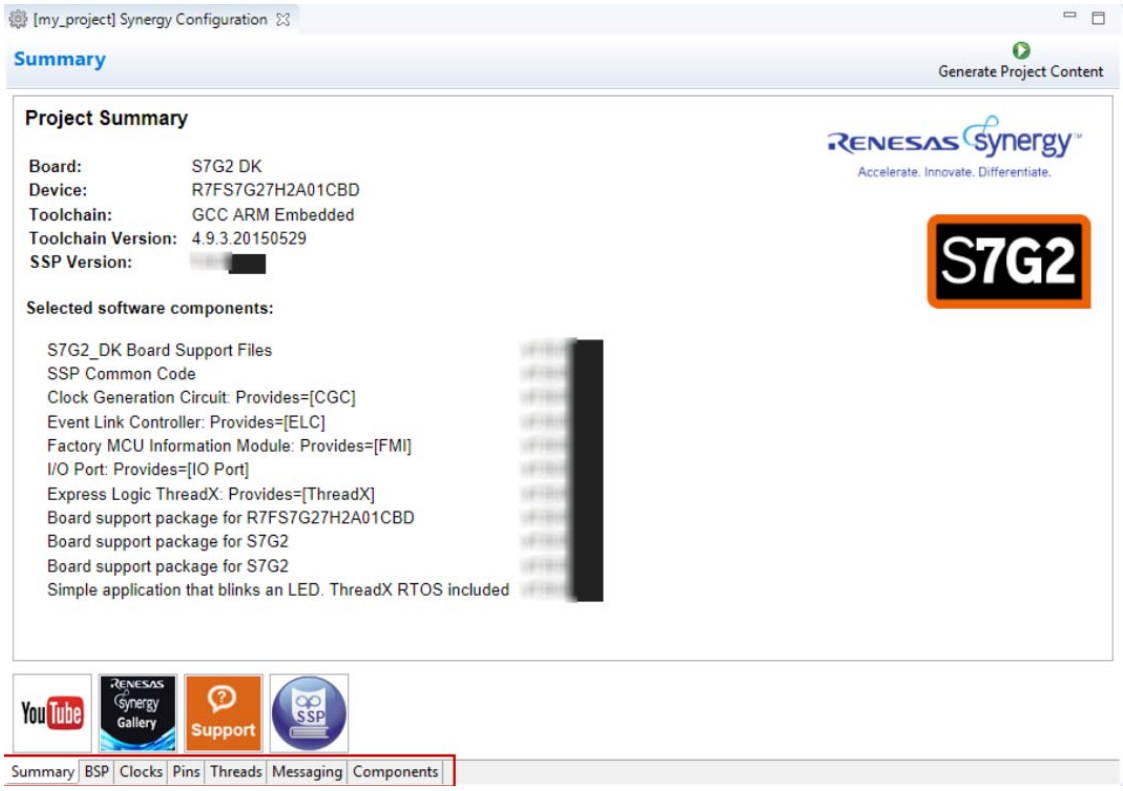


図 28: Synergy Project Editor と利用可能なエディタータブ

Synergy Project Editor ビューの下で、自分のプロジェクトの複数のアスペクトを構成できるタブがあります。

- [BSP] タブでは、初期プロジェクト選択からボード別のパラメータを変更できます。
- [Clocks] タブでは、プロジェクトの MCU クロック設定を構成できます。
- [Pins] タブでは、各ポートピンの電気特性と機能をペリフェラルレベルまたは個々のピンレベルで構成できます。
- [Threads] タブでは、RTOS およびそれ以外のアプリケーションの SSP モジュールとドライバーの追加と、モジュールとドライバーの構成を実行できます。このタブで選択したモジュールまたはドライバーごとに、[Properties] ビューでは構成パラメータ、割り込み優先順位、ピンの選択へのアクセスを提供します。
- [Messaging] タブでは、ThreadX ベースプロジェクトのメッセージフレームワークを構成できます。[Messaging] タブは、e2 studio バージョン 5.0 以降に含まれています。
- [Components] タブでは、選択したモジュールの概要を提供しています。ここでは、特定の SSP リリースのドライバーおよびアプリケーションサンプルコードを追加することもできます。しかし、基本的には [Threads] タブを使用して、SSP モジュールをプロジェクトに追加するようにしてください。

3.1.6 プロジェクトの設定

プロジェクトには 2 つのレベルの構成があります。

- [BSP]、[Clocks]、および [Pins] のタブは、リセット後にユーザーコードが実行される前の MCU の初期構成を決定します。プロジェクトの作成時にプロジェクトテンプレートを選択することにより、ISDE は選択したボードに適したデフォルト値を設定します。それらのデフォルト値は必要に応じて変更できます。
- [Threads] タブでは、プロジェクトに SSP モジュールを追加でき、アプリケーションの必要に応じてモジュールの構成パラメータを設定できます。メッセージングフレームワークは ThreadX ベースのアプリケーションに不可欠な部分であるため、[Messaging] タブではメッセージングが必要な各スレッドのメッセージングフレームワークを設定できます（e2 studio バージョン 5.0 以降の場合）。

3.1.6.1 ISDE での BSP の設定

[BSP] タブには、現在選択されているボード（選択されている場合）とデバイスが表示されます。[Properties] ビューは、下図の Synergy パースペクティブの左下に表示されています。

注 : [Properties] ビューが表示されていない場合、トップメニューバーで [Window] > [Show View] > [Properties] をクリックします。

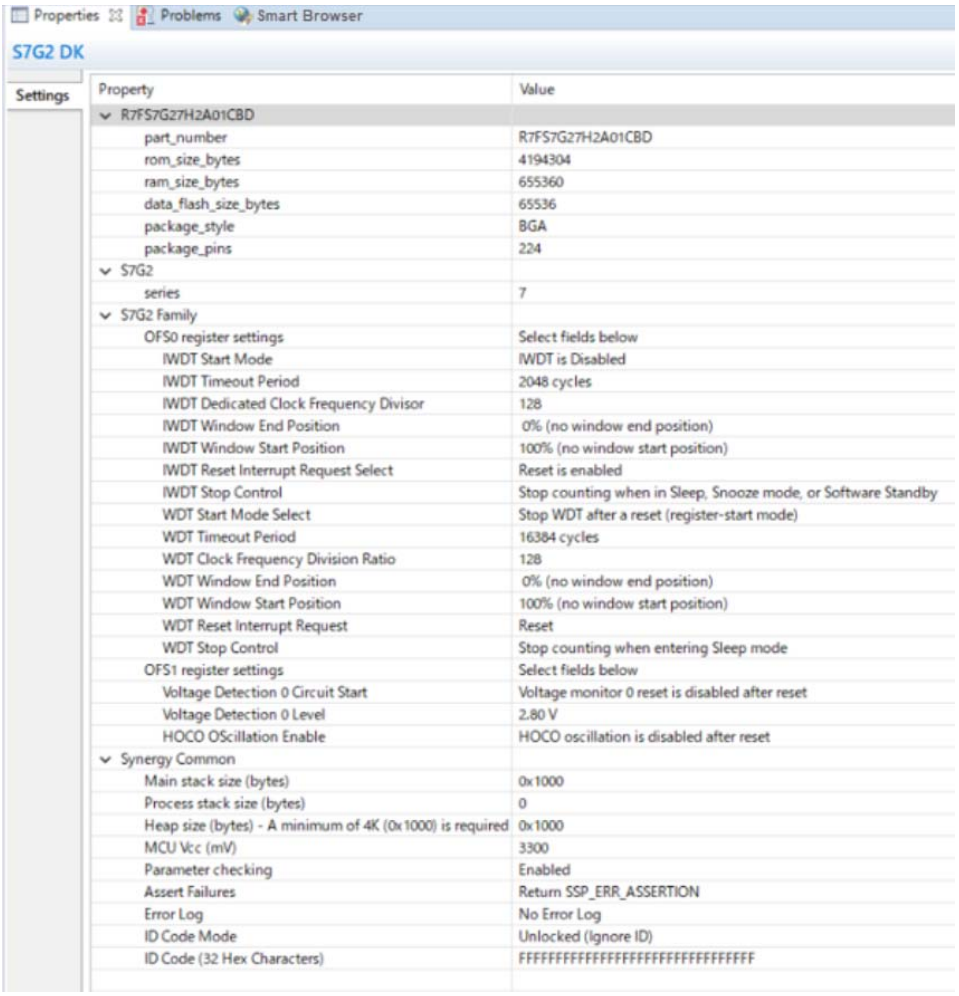


図 29: ISDE の [BSP] タブ

[BSP] タブでは、[Properties] ビューに BSP で使用できる構成可能なオプションが表示されます。これらのオプションは、必要に応じて変更できます。BSP は、MCU ハードウェア上の SSP レイヤーです。ISDE は、入力フィールドをチェックし、無効な入力を通知します。たとえば、スタックサイズには有効な数値のみを入力できます。

[Generate Project Content] ボタンを押すと、BSP の構成内容が次のファイルに書き込まれます。

synergy_cfg/ssp_cfg/bsp/bsp_cfg.h

このファイルが存在しない場合は作成されます。

注 : このファイルは、[GENERATE PROJECT CONTENT] ボタンを押すたびに上書きされるため、編集しないでください。

3.1.6.2 クロックの設定

[Clocks] タブには、MCU のクロックツリーがグラフィカルに表示され、各種のクロック除算値とソースを変更できます。クロック設定が無効な場合、問題のあるクロック値は赤く強調表示されます。この設定でもコードを生成できますが、正しい動作は保証されません。次の図で、USB クロック UCLK の除算値が変更され、クロック周波数が必要な 48 MHz ではなく 60 MHz になっています。このパラメータは赤く表示されます。

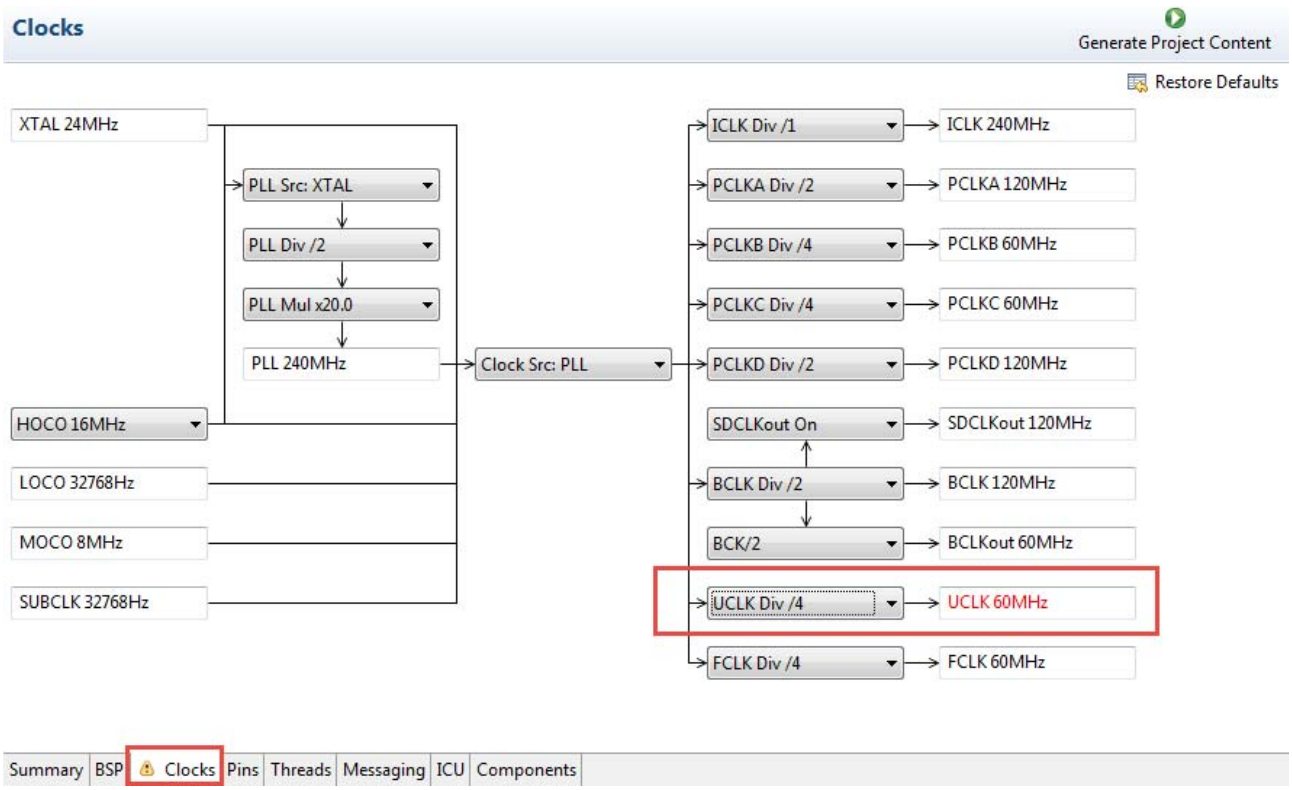


図 30: [ISDE Clocks] タブ

[Generate Project Content] ボタンを押すと、クロック構成の内容が次のファイルに書き込まれます。

synergy_cfg/ssp_cfg/bsp/bsp_clock_cfg.h

このファイルが存在しない場合は作成されます。

注 : このファイルは、[GENERATE PROJECT CONTENT] ボタンを押すたびに上書きされるため、編集しないでください。

3.1.6.3 ピンの設定

[Pins] タブでは、MCU のピンを柔軟に設定できます。多数のピンが多数の機能を提供できるため、ペリフェラルごとに設定できます。たとえば、SCI でシリアルチャネルを選択すると、そのモジュールとチャネルの送受信ピンの配置に関する複数のオプションが提供されます。設定されたピンは、[Package] ビューに緑色で表示されます。

注 :[Package] ビューウィンドウが ISDE で開いていない場合、トップメニューバーから [Window] > [Show View] > [Pin Configurator] > [Package] を選択して開きます。

[Pins] タブでは、ピンまたはペリフェラルごとにエラーをハイライトし、オプションを表示することで、ピンが高度に多重化された大規模なパッケージを簡単に構成できます。DK-S7G2 などの特定のボードでプロジェクトテンプレートを選択した場合、ボードに接続された一部のペリフェラルは事前に選択されています。

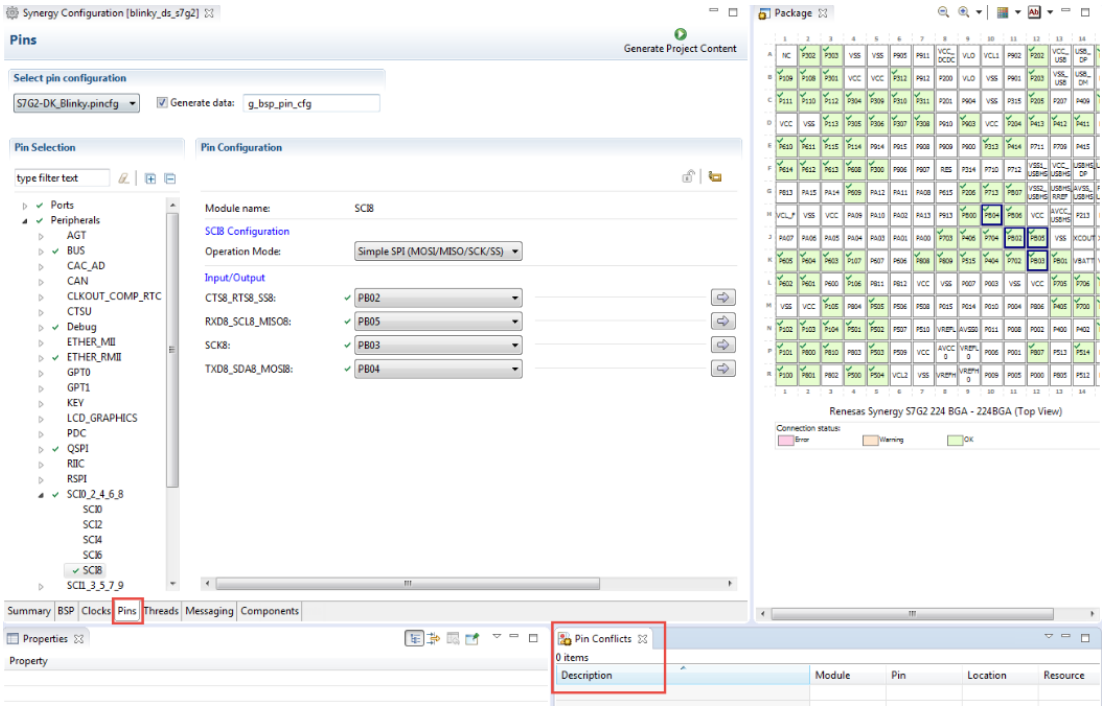


図 31: [ISDE Pins] タブ

ピンコンフィギュレータは、競合チェッカーを内蔵しています。同じピンが別のペリフェラルや I/O 機能に割り当てられた場合、ピンがパッケージビューで赤く表示され、メインの [Pins] タブの [Pin Selection] ペインと [Pin Configuration] ペインの赤い四角形の中に白い十字とともに表示されます。[Pin Conflicts] ビューにはすべての衝突が表示されるため、容易に識別および修正できます。

下の例では、ポートピン P100、P101、P102 は既に外部メモリペリフェラルに割り当てられ、使用されているので、これらのポートをシリアル通信インタフェース 0（SCI0）につなげようとすると、ダングリング接続エラーが発生します。このエラーを修正するには、別の SCI インタフェースを選択するか、タブの左側にある [Pin Selection] ペインで競合するペリフェラル（この場合は外部メモリペリフェラル）を無効にします。

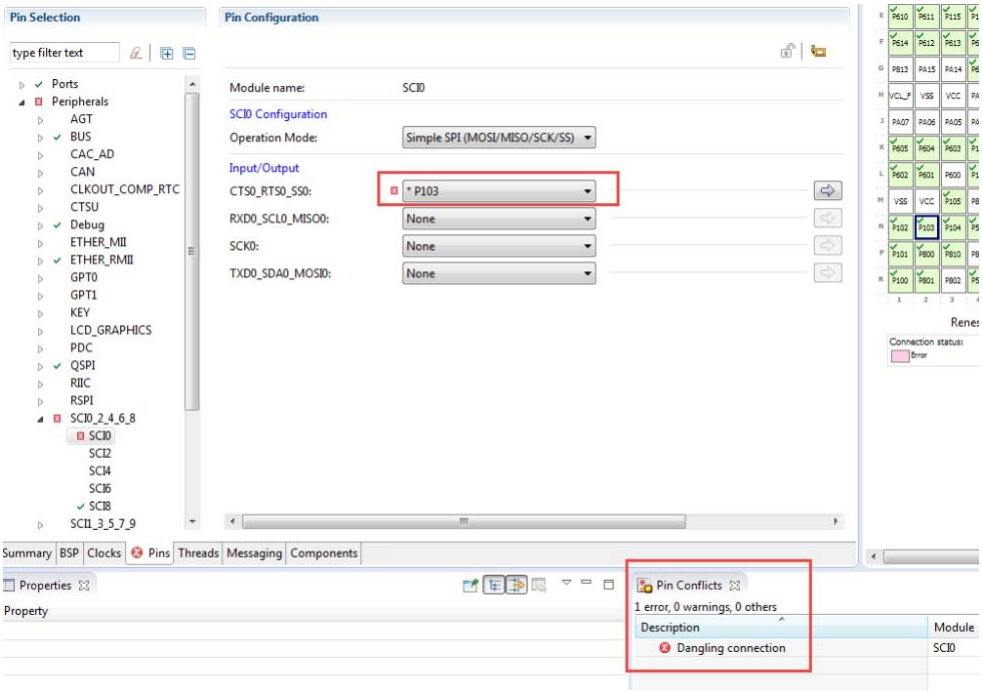


図 32: ISDE ピンコンフィギュレーター

ピンコンフィギュレータの [Package] ビューには、選択された各ピンの電気特性または機能特性が表示されます。

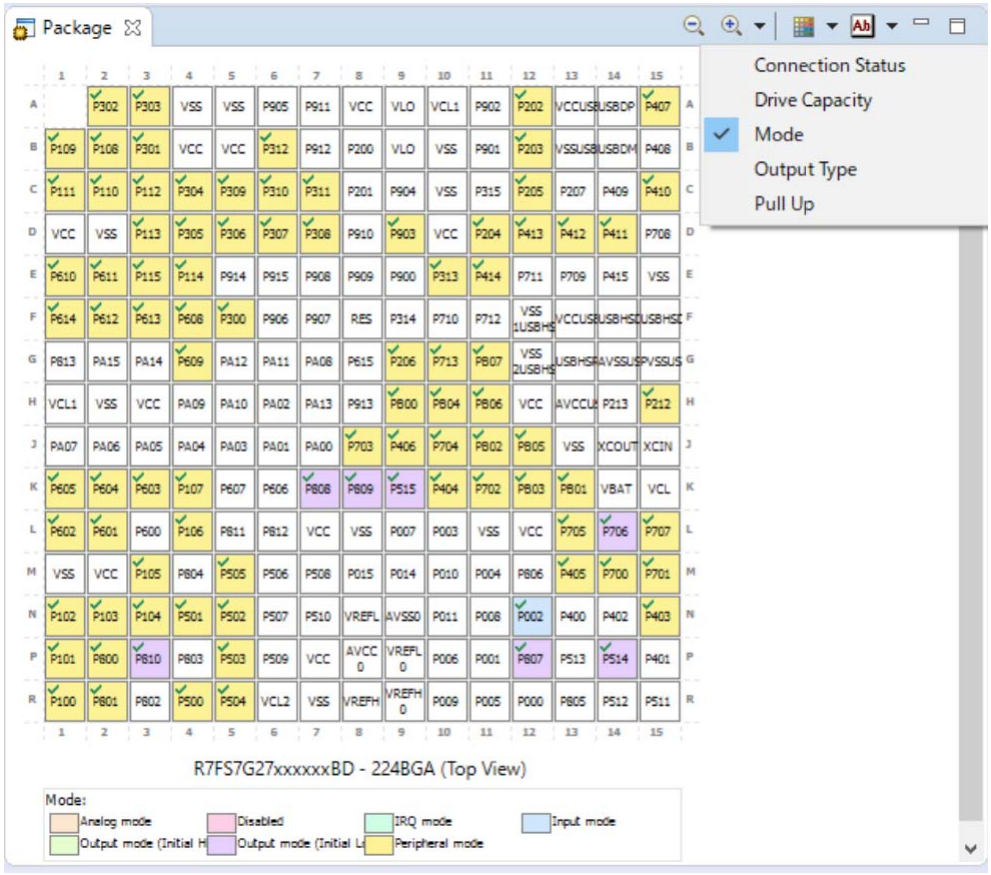


図 33: 統合ソリューション開発環境ピンコンフィギュレータの [Package] ビューに表示された各ピンの動作モード

[Generate Project Content] ボタンを押すと、ピン構成の内容が次のファイルに書き込まれます。

synergy_cfg/ssp_cfg/bsp/bsp_pin_cfg.h および

src/synergy_gen/pin_data.c_

これらのファイルが存在しない場合は作成されます。

注 : これらのファイルは、[GENERATE PROJECT CONTENT] ボタンを押すたびに上書きされるため、編集しないでください。

ピンに関する情報を簡単に共有できるよう、統合ソリューション開発環境はピン構成設定を csv 形式でエクスポートし、csv ファイルを synergy_cfg/ssp_cfg/bsp/<pincfg_file_name>.csv にコピーします。

3.1.7 スレッドとドライバーの追加

すべての ThreadX ベースの Synergy プロジェクトには、少なくとも 1 つの RTOS スレッドと、そのスレッドを実行する SSP モジュールのスタックが 1 つ含まれています。[Threads] タブは、スレッドに適切な SSP モジュールを追加して、スレッドのプロパティと各スレッドに関連するモジュールのプロパティを構成できるグラフィカルユーザーインタフェースです。スレッドを設定すると、その構成に応じて ISDE が自動的にコードを生成します。

任意のドライバー、さらに一般的にはスレッドに追加する任意のモジュールに対し、統合ソリューション開発環境は他のモジュールとのすべての依存関係を自動的に解決し、適切なスタックを作成します。このスタックは、選択されたモジュールとモジュールオプションに応じ、ISDE によって自動的に [Threads] ペインに入力表示されます。依存関係の要件を満たすモジュールが 2 つ以上ある場合、ドロップダウンメニューからモジュールを選択するよう表示されます。

たとえば、スレッドにオーディオ再生フレームワークを追加する際には、再生のフレームワークとして DAC または I2S を選択する必要があります。

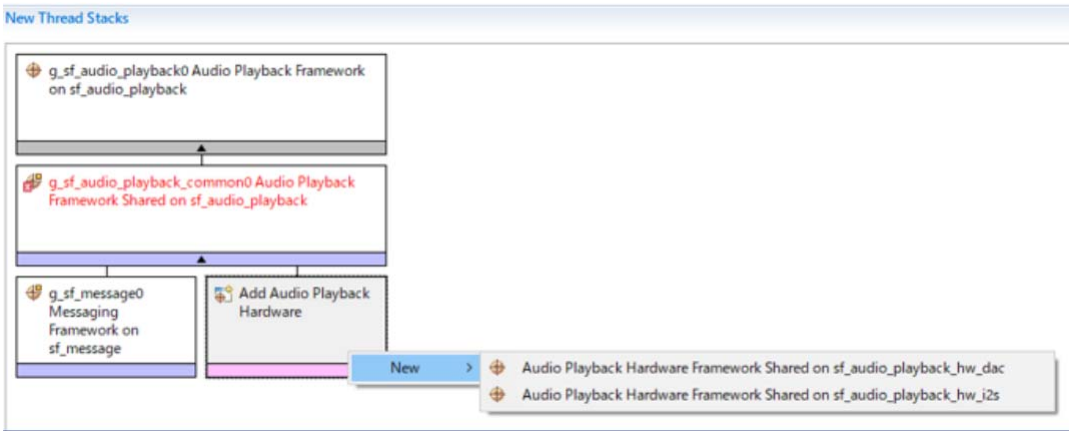


図 34: [Threads] タブでのモジュール依存関係の実現

[Threads] タブのデフォルトビューには、[HAL/Common] という共通スレッドが含まれています。このスレッドには、I/O 制御（IOPORT）、クロック発生回路（CGC）、ファクトリー MCU 情報（FMI）、イベントリンクコントローラ（ELC）のドライバーが付属しています。デフォルトのスタックは [HAL/Common Stacks] ペインに表示されています。HAL/ 共通スレッドに追加されるデフォルトのモジュールは、SSP がそれぞれに単一のインスタンスのみを必要とするという点で特殊で、これは ISDE によってすべてのユーザー定義スレッドにデフォルトで含まれます。

RTOS を使用しないアプリケーション、または RTOS 外で実行されるアプリケーションでは、HAL/ 共通スレッドがアプリケーションにドライバーを追加できるデフォルトの場所となります。

モジュールとスレッドの追加および設定方法についての詳細は、以下のセクションを参照してください。

- HAL ドライバーの追加と設定
- ドライバーのスレッドへの追加とドライバーの設定

モジュールを [HAL/Common] または新しいスレッドに追加したら、[Properties] ビューでドライバー設定オプションにアクセスできます。スレッドオブジェクトを追加したら、同様に [Properties] ビューでオブジェクト構成オプションにアクセスできます。

スレッドの設定方法については、スレッドの設定を参照してください。

注 : ドライバーやモジュールの選択肢や構成オプションは SSP パックで定義され、そのため SSP バージョン変更時に変更することができます。

注 : SSP のモジュールとドライバーの定義については、「*SSP 用語*」を参照してください。

3.1.7.1 HAL ドライバーの追加と設定

RTOS の外または RTOS なしで実行されるアプリケーションの場合、[Threads] タブで [HAL/Common] スレッドを使用してアプリケーションに他の HAL ドライバーを追加できます。ドライバーを追加するには、次の手順を実行します。

1) [Threads] ペインで HAL/Common アイコンをクリックします。[Modules] ペインが [HAL/Common Stacks] に変わります。

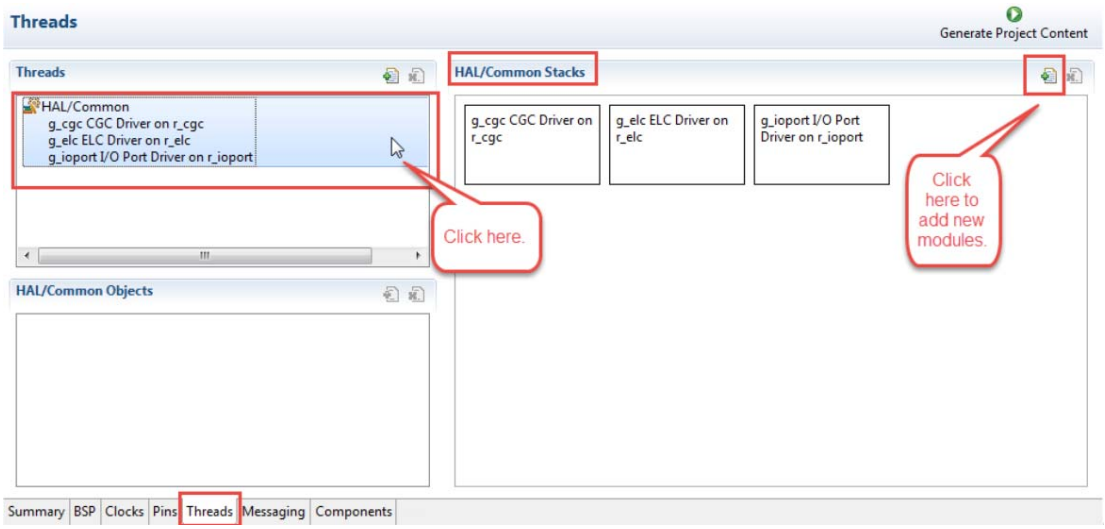


図 35: ISDE プロジェクトコンフィギュレーター - ドライバーの追加

2) [New Stack] をクリックすると、SSP で使用可能な HAL レベルドライバーのドロップダウンリストが表示されます。

3) [New] > [Driver] からドライバーを選択します。また、[New] > [Framework] > [Service] > [Power Profiles Framework on sf_power_profiles] から RTOS 独立アプリケーションのパワープロファイルを選択することができます。他のすべてのモジュールは、ThreadX が存在する場合にのみ、スレッドに追加できます。

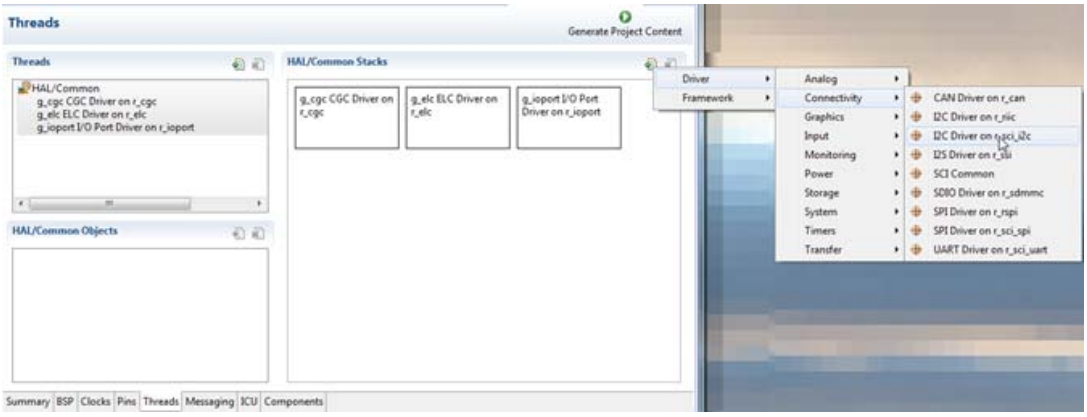


図 36: ドライバーの選択

ISDE は選択されたドライバーのスタックを作成し、ドライバーで有効化する必要のある追加のリソースがいつ必要かを知らせます。次の例では、[Properties] ビューで、CAN ドライバーに関する割り込みとコールバック関数名を構成できます。

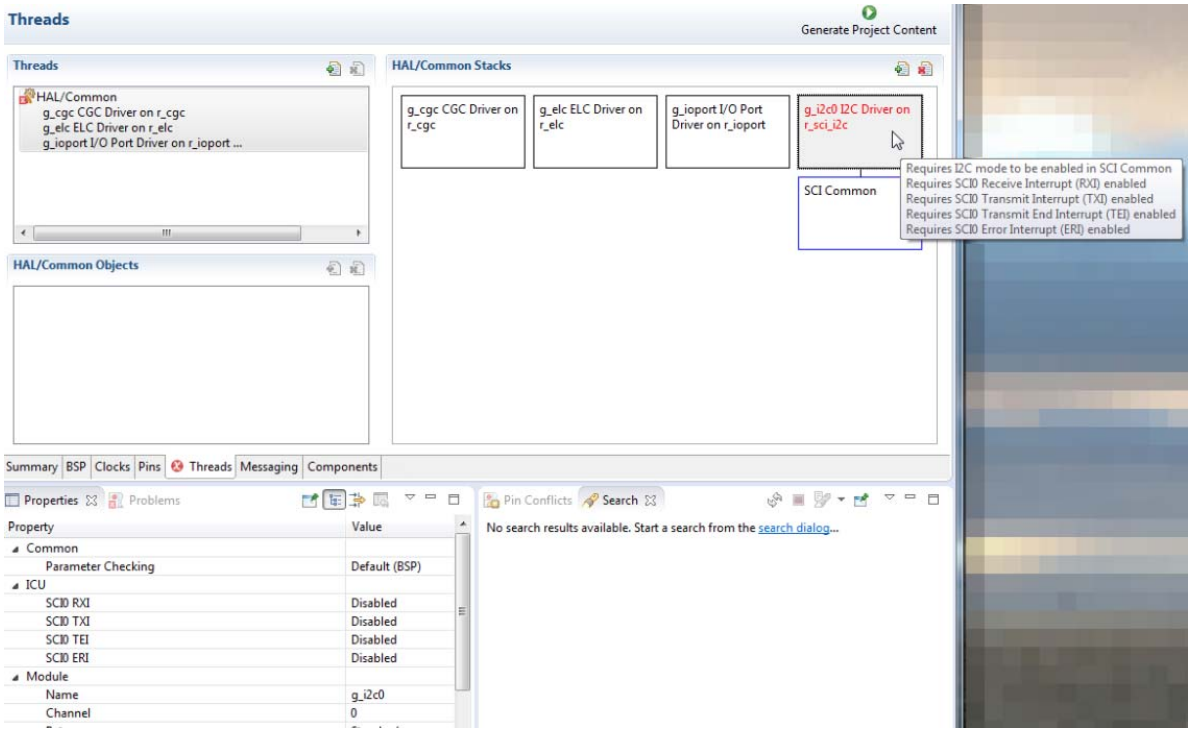


図 37: [Threads] タブでの依存関係の確認

4) ドライバーモジュールを [HAL/Common Modules] ペインで選択し、ドライバーのプロパティを [Properties] ビューで構成します。

[Generate Project Content] ボタンをクリックすると、統合ソリューション開発環境によって以下のファイルが追加されます。

- synergy/ssp ディレクトリ向けに選択したドライバーモジュールとそのファイル。
- main() 関数と構成構造体、およびアプリケーション用のヘッダーファイルが下記の表に表示されています。

File	内容	[プロジェクトコンテンツの生成] で上書きされるか
src/synergy_gen/main.c	main() 呼び出しとユーザーコードが含まれます。呼び出されたときには、BSP により MCU が初期化されています。	はい
src/synergy_gen/hal_data.c	HAL ドライバーのみのモジュールの設定構造体。	はい
src/synergy_gen/hal_data.h	HAL ドライバーのみのモジュールのヘッダーファイル。	はい
src/hal_entry.c	HAL ドライバーのみのコード用のユーザーエントリポイント。ここにコードを追加します。	いいえ

追加されているすべてのモジュール用の構成ヘッダーファイルは、このフォルダーに作成または上書きされます。

synergy_cfg/ssp_cfg/driver

3.1.7.2 ドライバーのスレッドへの追加とドライバーの設定

ThreadX RTOS で使用されるアプリケーションの場合、1 つ以上のスレッドを追加し、スレッドごとにそのスレッドで実行されるモジュールを追加できます。ドライバーまたはフレームワークのドロップダウンメニューからモジュールを選択できます。スレッドにモジュールを追加するには、次の手順を実行します。

1) [Threads] ペインで、[New Thread] をクリックし、スレッドを追加します。

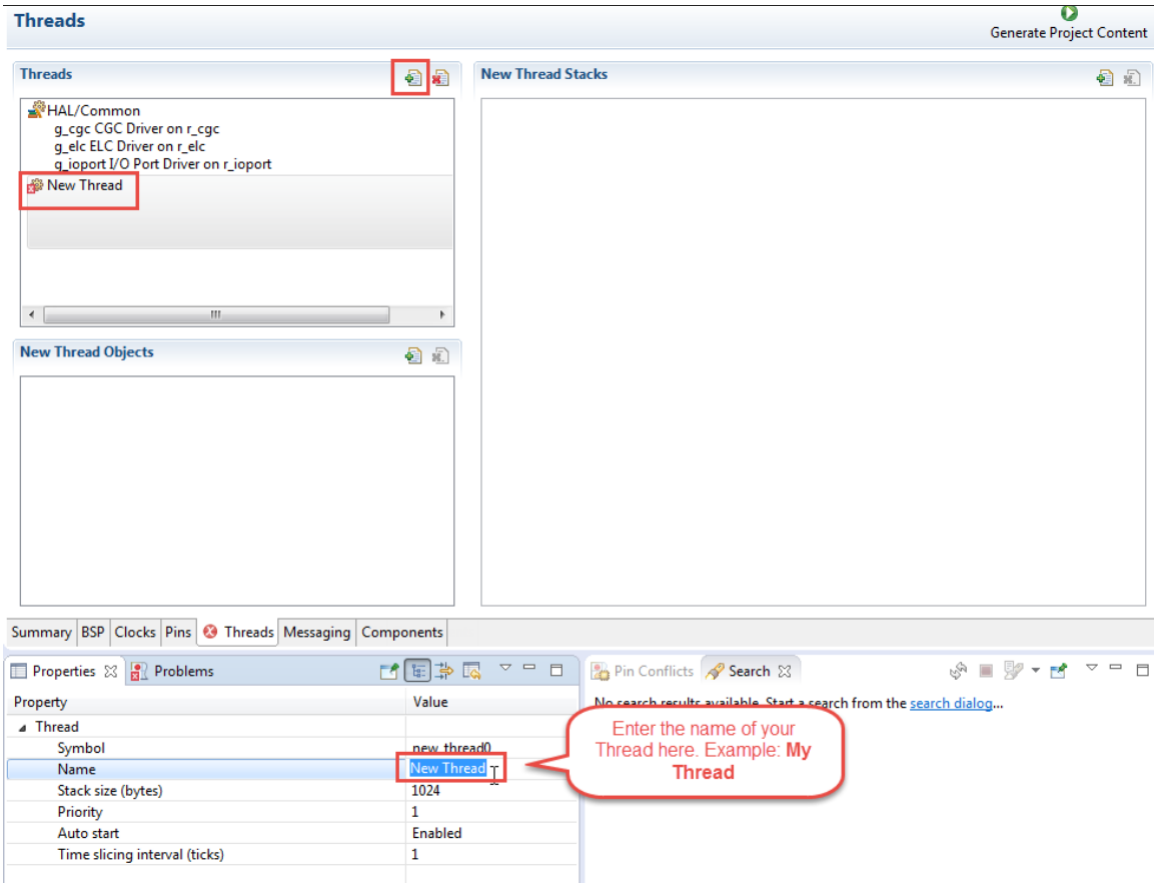


図 38: [Threads] タブでの新規 [RTOS Thread] の追加

2) [Properties] ビューで、[Name] と [Symbol] のエントリーをクリックして、新しいスレッドに固有の名前と記号を入力します。

注 : 統合ソリューション開発環境が、スレッドスタックペインの名前をユーザーがスレッド用に選択した名前に更新します。

3) スレッドスタックペインで [New] をクリックして、モジュールとドライバーのリストを表示します。フレームワークレベルのモジュールと HAL レベルのドライバーの両方をここで追加できます。

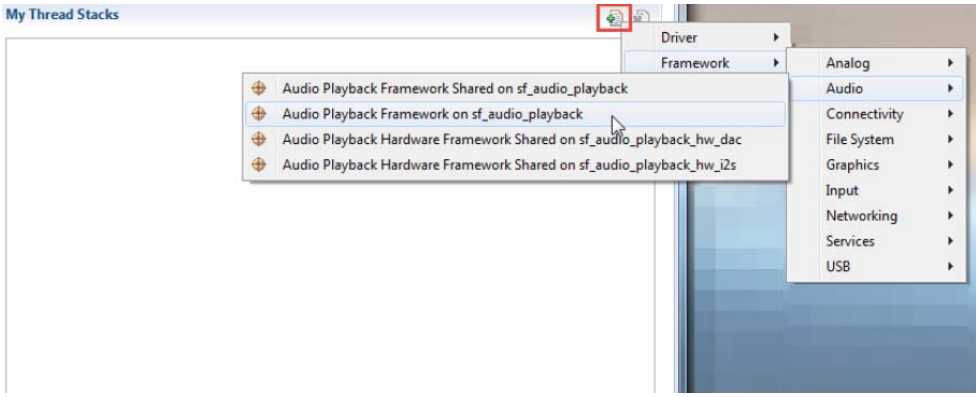


図 39: モジュールとドライバーをスレッドに追加

4) リストからモジュールまたはドライバーを選択します。
5) モジュールまたはドライバーが依存関係を示している場合、不足しているリソースを選択します。

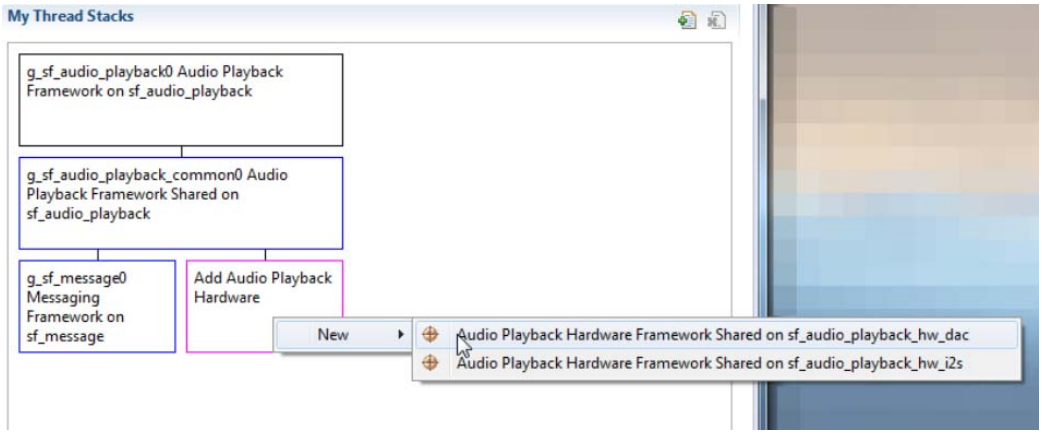


図 40: [Threads] タブでモジュールまたはドライバーの依存関係の特定

6) 追加したドライバーをクリックして、[Properties] ビューの構成パラメータを更新し、必要な依存関係を解決することにより、アプリケーションに必要なドライバーを構成します。選択したモジュールまたはドライバーを表示し、そのプロパティを編集するには、ドライバーを含むスレッドが [Threads] ペインでハイライトされていることを確認します。

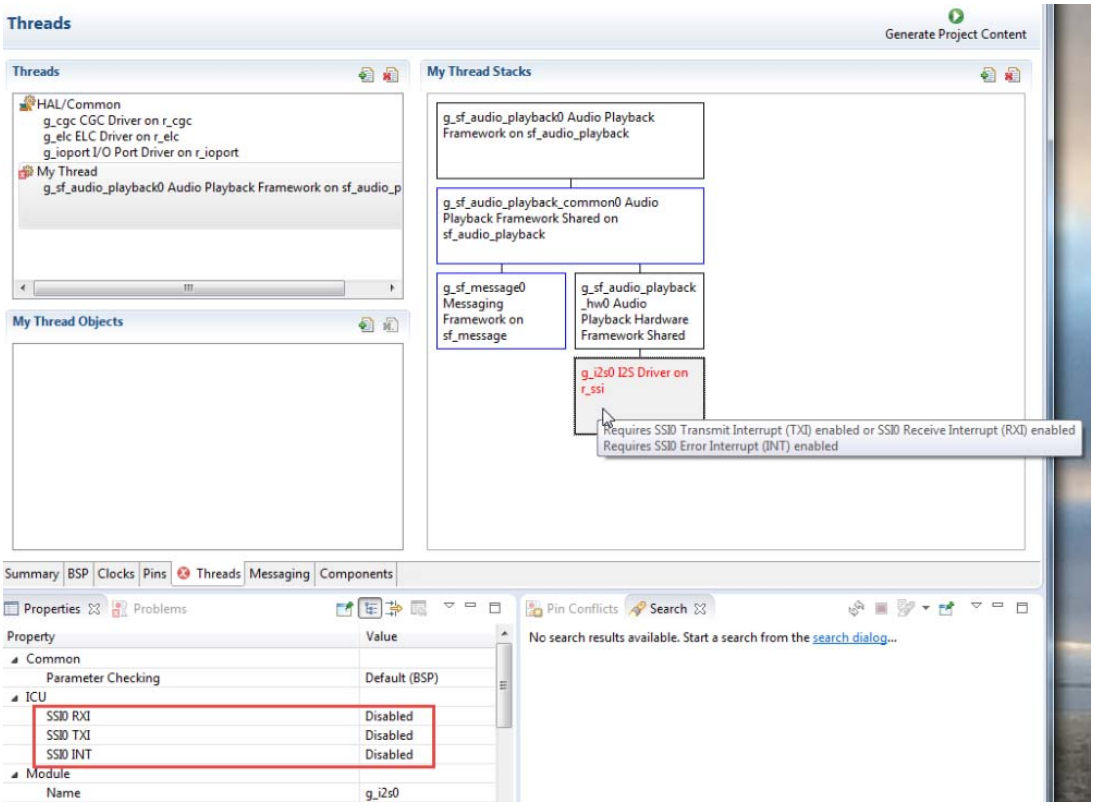


図 41: モジュールまたはドライバーのプロパティの設定

7) 必要に応じて、[Threads] ペインの [New] をクリックして、別のスレッドを追加します。

上記の例で [Generate Project Content] ボタンを押すと、ISDE によって以下の表に示されているファイルが作成されます。

File	内容	[プロジェクトコンテンツの生成] で上書きされるか
src/synergy_gen/main.c	main() 呼び出しとユーザーコードが含まれます。呼び出されたときには、BSP により MCU が初期化されています。	はい
src/synergy_gen/my_thread.c	生成されたスレッド「my_thread」と、このスレッドに追加されたモジュールの構成構造体。	はい

File	内容	[プロジェクトコンテンツの生成] で上書きされるか
src/synergy_gen/my_thread.h	スレッド「my_thread」用のヘッダーファイル	はい
src/synergy_gen/hal_data.c	HAL ドライバーのみのモジュールの設定構造体。	はい
src/synergy_gen/hal_data.h	HAL ドライバーのみのモジュールのヘッダーファイル。	はい
src/hal_entry.c	HAL ドライバーのみのコード用のユーザーエントリポイント。ここにコードを追加します。	いいえ
src/my_thread_entry.c	スレッド「my_thread」用のユーザーエントリポイント。ここにコードを追加します。	いいえ

追加されているすべてのモジュールとドライバー用の構成ヘッダーファイルは、次のフォルダーに作成または上書きされます。

synergy_cfg/ssp_cfg/driver

synergy_cfg/ssp_cfg/framework

3.1.7.3 スレッドの設定

アプリケーションで ThreadX RTOS が使用されている場合、[Threads] タブを使用して、ThreadX スレッド、セマフォ、ミューテックス、イベントフラグを簡単に作成できます。

各スレッドのコンポーネントは、([Threads] ペインで選択されていれば) [Properties] ビューで構成できます。

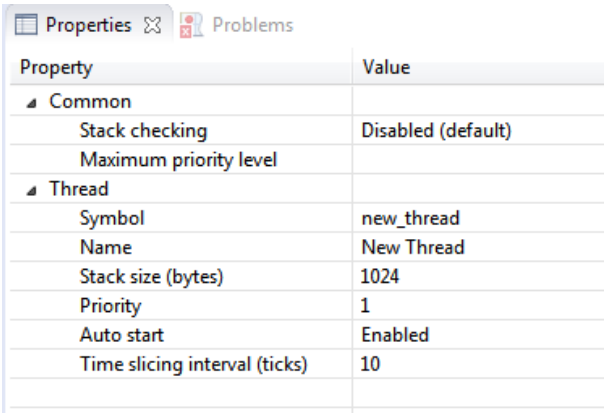


図 42: ISDE スレッドのプロパティ

[Properties] ビューには、すべてのスレッドに共通の設定([Common])と、この特定のスレッドの設定([Threads])が含まれています。

このスレッドインスタンスについて、スレッドの名前、プライオリティレベルやスタックサイズなどのプロパティを簡単に設定できます。ISDE は、プロパティフィールドのエントリーが有効であることを確認します。たとえば ISDE は、整数値を必要とする [Property] フィールドなどに 0 ～ 9 の数値のみが含まれていることを確認します。

ThreadX リソースをスレッドに追加するには、[Threads Objects] ペインでスレッドを選択して [New Object] をクリックします。ペインは選択したスレッドの名前、この場合は [Audio Thread Objects] となります。

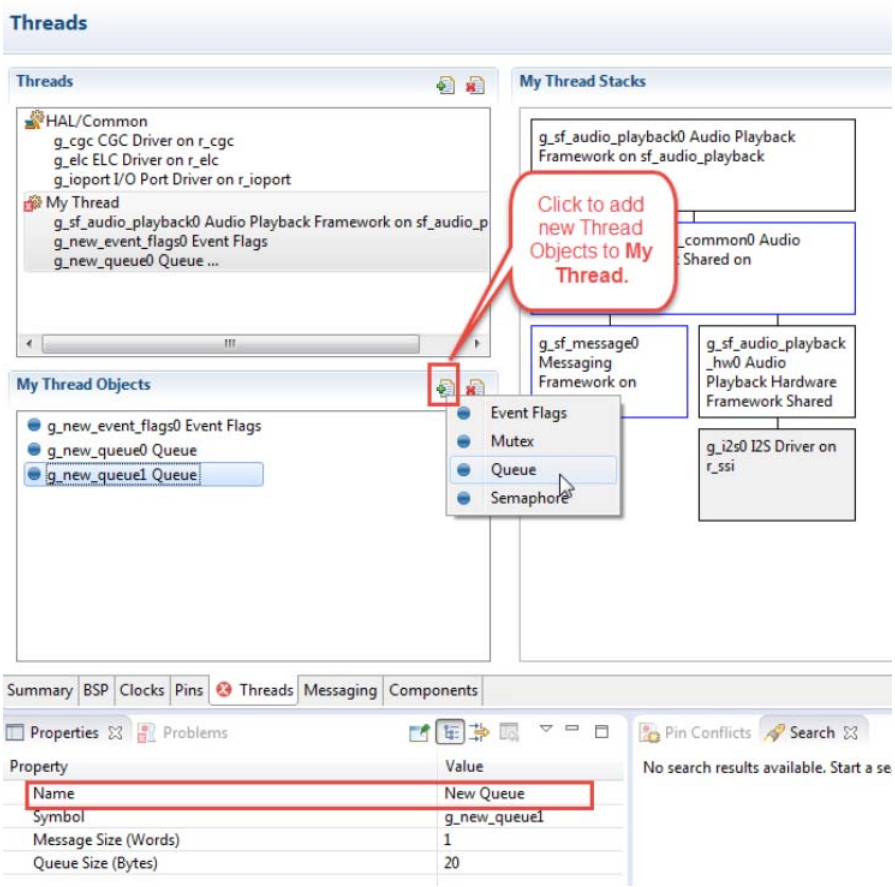


図 43: スレッドプロジェクトのプロパティの設定

それぞれのスレッドオブジェクトに固有の名前と記号が付されるよう、[Name] と [Symbol] のエントリーを [Properties] ビューで更新します。

3.1.7.4 割り込みの設定

[Threads] タブにある [Properties] ビューを使用して割り込み優先順位を設定すると、割り込みを有効にすることができます。[Threads] ペインで表示するスレッドを選択し、プロパティを編集します。

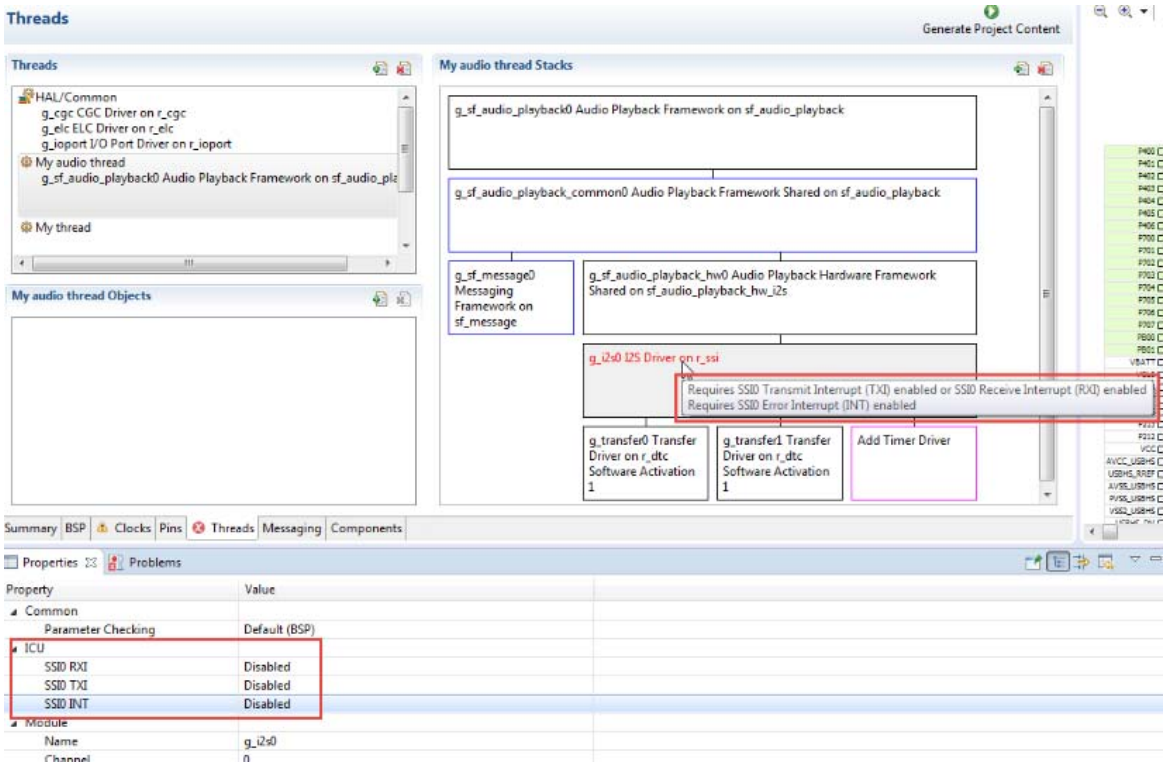


図 44: [Threads] タブで割り込みの設定

3.1.8 SSP メッセージングフレームワークの設定

メッセージングフレームワークはThreadXメッセージングキュー機能に拡張される、最も重要なSSPモジュールの 1 つです。これは、スレッドが相互にメッセージを交換して通信するメカニズムを提供します。メッセージングフレームワークは、事前設定またはユーザー設定されたイベントが発生した際にスレッドがメッセージを送信（パブリッシュ）または受信待ち（リッスン）できるようにします。スレッドは、特定のイベントクラスをサブスクライブしているすべてのスレッドが受信待ちおよび対応できるイベントクラスを付随させたメッセージをパブリッシュできます。特定のイベントクラスを受信待ちできるスレッドのリストは、そのイベントクラスのサブスクライバーリストと呼ばれます。

メッセージングフレームワークを使用するには、まず [Threads] タブでメッセージングフレームワークインスタンスを 1 つ追加する必要があります。メッセージングフレームワークは、HAL/ 共通スレッド以外のすべてのスレッドに追加できます。プロジェクト内のすべてのスレッドがこのインスタンスを使用して相互に通信できます。タッチパネルフレームワークやオーディオ再生フレームワークなどのモジュールではメッセージングフレームワークが必要で、以下のオーディオ再生フレームワークの例に示されているとおり、自動的に追加されます。プロジェクトにモジュールなどのスレッドが含まれている場合は、アプリケーションにスレッドを追加しても、メッセージングフレームワークの他のインスタンスを追加する必要はありません。すべてのスレッドが 1 つのメッセージングフレームワークインスタンスを共有できます。

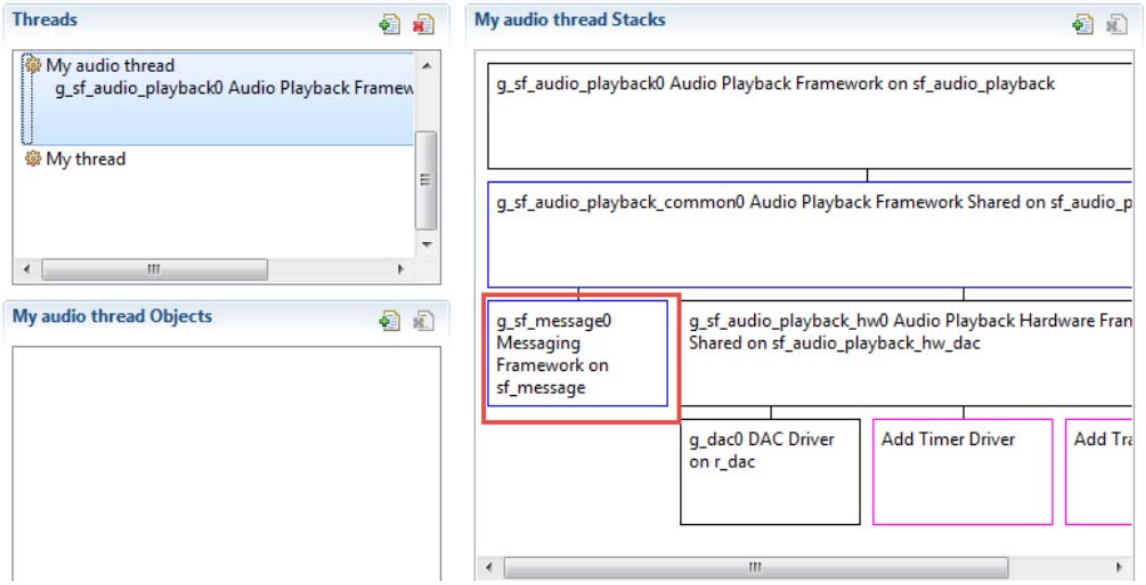


図 45: スレッドへのメッセージングモジュールの追加

[Threads] タブにメッセージングフレームワークインスタンスを追加したら、[Messaging] タブを使用して独自のイベントクラスとイベントを定義し、どのスレッドがどのイベントクラスを受信待ちするかを決定できます。SSP には、タッチパネルおよびオーディオフレームワークモジュール用に事前定義されたイベントクラスとイベントが含まれています。これらのモジュールを追加した場合、事前定義されたイベントクラスとイベントが [Messaging] タブにも表示されます。オーディオ再生フレームワーク用に事前定義されたイベントクラスとイベントは、以下のとおりです。

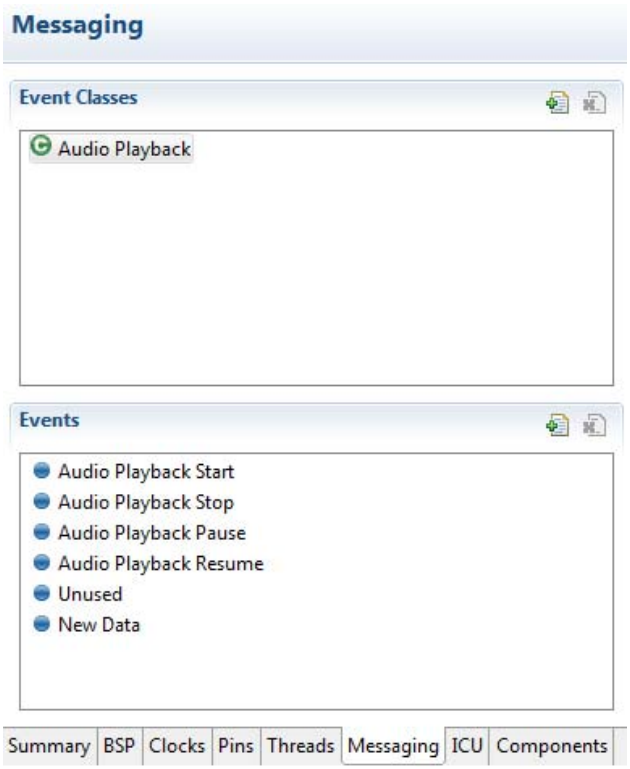


図 46: オーディオ再生フレームワーク用に事前定義されたイベントクラス

3.1.8.1 イベントクラスの追加

独自のユーザー定義イベントクラスをメッセージングシステムに追加するには、次の手順を実行します。

1) [Messaging] タブで、[Event Class] ペインを選択し、ボタンを追加します。
2) イベントクラスに一意の名前を入力します。
3) [OK] をクリックします。

   注 : ユーザー定義のイベントクラスは、イベントクラスアイコン右上の金色の四角形でマークされています。

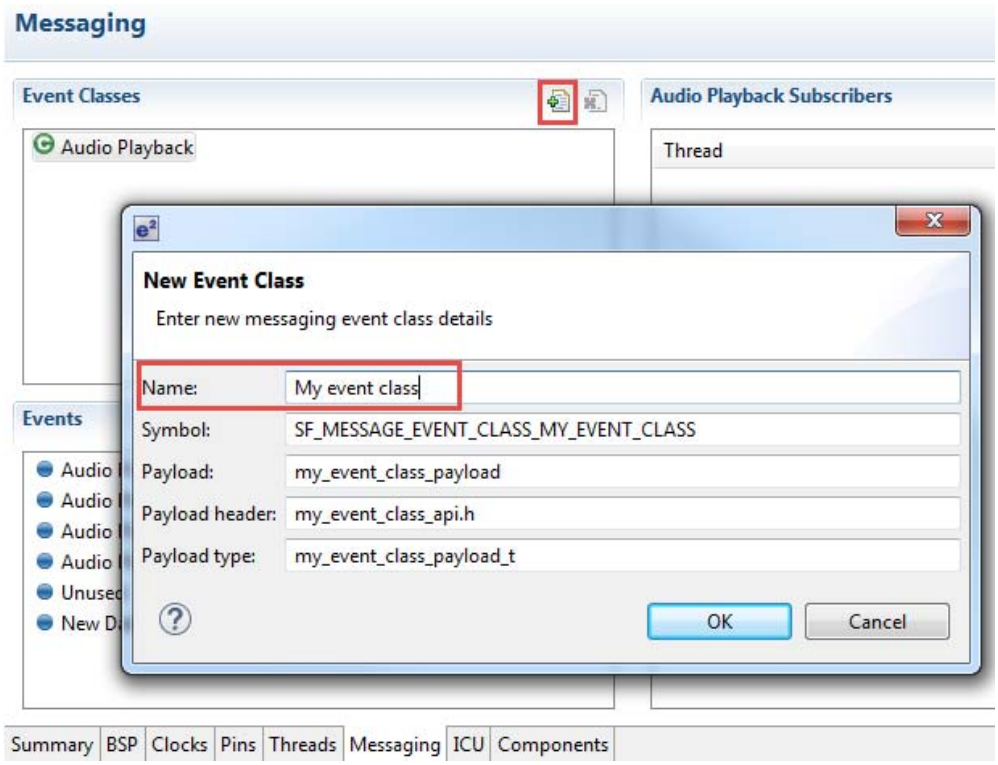


図 47: メッセージング – イベントクラスの追加

3.1.8.2 イベントの追加

独自のユーザー定義イベントをメッセージングシステムに追加するには、次の手順を実行します。

1) [Messaging] タブで、[Event] ペインを選択し、ボタンを追加します。
2) イベントクラスに一意の名前を入力します。
3) [OK] をクリックします。

   注 : ユーザー定義のイベントは、イベントアイコン右上の金色の四角形でマークされています。

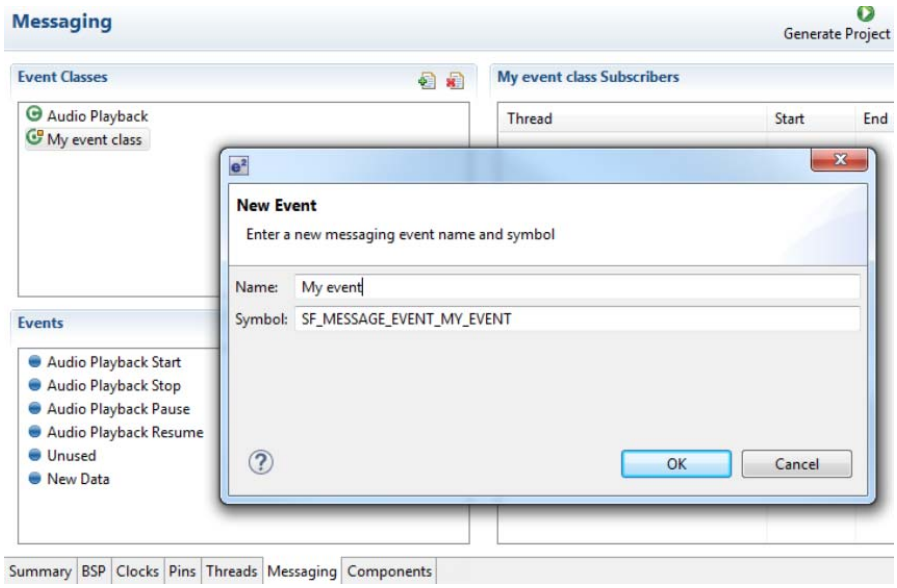


図 48: メッセージング – イベントの追加

3.1.8.3 メッセージングサブスクライバーリストの設定

サブスクライバーリストでは、メッセージパブリッシャーからのメッセージを受信待ちするスレッドを選択します。パブリッシュスレッドと受信待ちスレッド間の接続は、イベントクラスを通じて確立されます。そのため、プロジェクト内の各イベントクラスについてサブスクライバーリストを定義する必要があります。サブスクライバーリスト内のすべてのスレッドが、選択されたイベントクラスに属するメッセージを受信待ちして対応できます。

以下では、[Threads] タブで 2 つのスレッドが定義され、そのうちの 1 つがオーディオ再生フレームワークを使用することが想定されています。

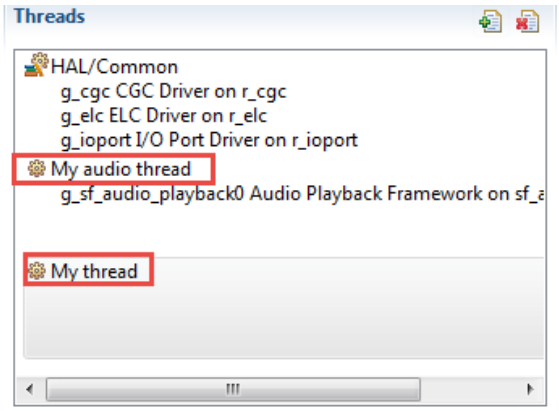


図 49: メッセージング – スレッドの例

イベントクラスのサブスクライバーリストを設定するには、次の手順を実行します。

1) [Messaging] タブの [Event Class] ペインでイベントクラスを選択します。[Subscriber List] ペインは選択したイベントクラスの名前になります。
2) 追加アイコンをクリックすると、[New Subscriber] ダイアログボックスが表示されます。
3) [Threads] ドロップダウンメニューから、サブスクライバーリストの追加するスレッドを選択します。
4) 開始点と終了点を選択して、インスタンス範囲を決定します。

   注：特定のイベントクラスのインスタンスが 1 つだけの場合は、開始値と終了値をデフォルト値（0）のままにします。イベントクラスのインスタンスが 2 つ以上ある場合にインスタンス範囲を選択する方法については、メッセージングフレームワークユーザーガイドを参照してください。複数のチャネルでのオーディオストリーミングなど、同じイベントクラスを複数回使用するアプリケーションでは、イベントクラスのインスタンスが複数あると有用です。
5) 選択したイベントクラスのサブスクライバーリストに追加するそれぞれのスレッドについて、手順 3 と 4 を繰り返します。

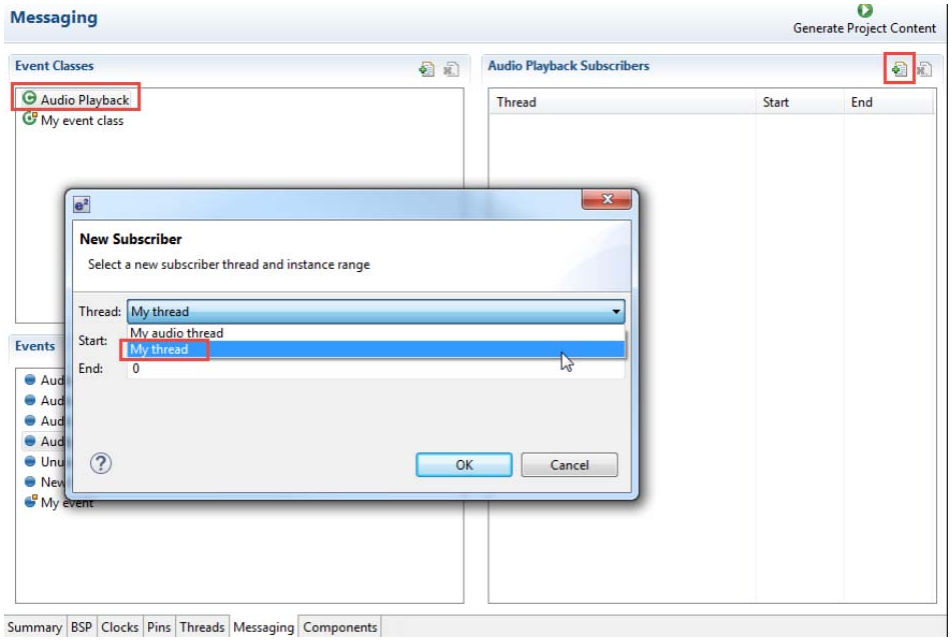


図 50: メッセージング – サブスクライバーリストの設定

3.1.8.4 メッセージングフレームワーク用ファイルの生成

[Generate Project] ボタンをクリックすると、統合ソリューション開発環境によって以下の構成済みメッセージフレームワーク用ファイルが生成されます。

File	内容	[プロジェクトコンテンツの生成] で上書きされるか
synergy_cfg/ssp_cfg/framework/sf_message_port.h	イベントクラスとイベントの列挙を含みます。	はい
synergy_cfg/ssp_cfg/framework/sf_message_payloads.h	イベントクラスペイロードへのポインタを含みます。	はい
synergy_cfg/ssp_cfg/framework/sf_message_payloads.h	メッセージングフレームワークのコンパイラオプション	はい

3.1.9 プロジェクトに含まれるコンポーネントの確認

[Components] タブには、アプリケーションに含まれる個々のコンポーネントのリストが表示されます。すべての Synergy プロジェクトに共通するコンポーネントは、あらかじめ選択されています（たとえば、[BSP]>[Board-specific BSP, and HAL Drivers]>[all]>r_cgc).[Threads] タブで選択されたモジュールに必要なコンポーネントは、すべて自動的に含まれます。コンポーネントを追加するには、必要なコンポーネントの横にあるボックスをオンにし、除外するにはボックスをオフにします。

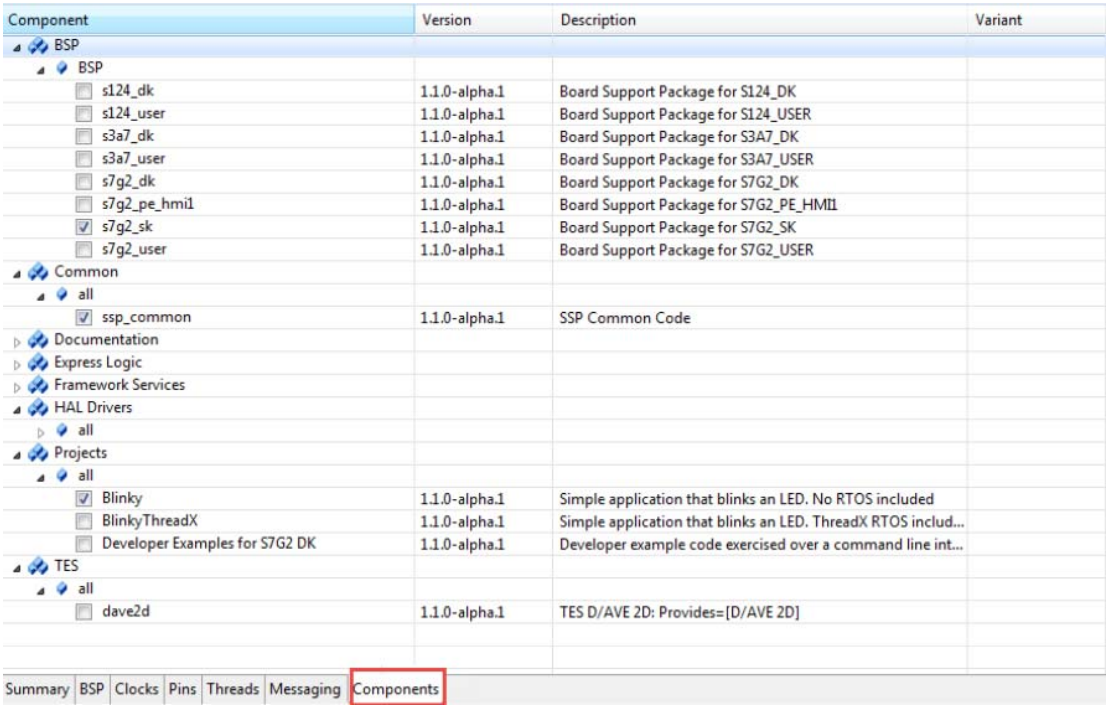


図 51: [Components] タブ

[Generate Project Content] ボタンを押すと、各コンポーネントの .c ファイルと .h ファイルがパックファイルから以下のフォルダにコピーされます。

- synergy/ssp/inc/bsp
- synergy/ssp/inc/driver
- synergy/ssp/inc/framework
- synergy/ssp/src/bsp
- synergy/ssp/src/driver
- synergy/ssp/src/framework

また統合ソリューション開発環境は、[Threads] タブの構成オプションが含まれた構成ファイルを synergy_cfg/ssp_cfg フォルダに作成します。

3.1.10 アプリケーションの作成

モジュールとドライバーを追加して、[Threads] タブで設定パラメータを設定した後、モジュールおよびドライバーを呼び出すアプリケーションコードを追加できます。

注 : 構成を確認するには、プロジェクトを一度ビルドして問題がないことを確かめてから、独自のアプリケーションコードを追加します。

3.1.10.1 RTOS 独立アプリケーション

アプリケーションを作成するには、以下の手順を実行します。

1) [Threads] タブですべてのドライバーとモジュールを追加し、不足している割り込みやドライバーのような、ISDE によってフラグされたすべての依存関係を解決します。
2) [Properties] ビューでドライバーを構成します。
3) [Project Configuration] ビューで [Generate Project Content] ボタンを押します。
4) [Project Explorer] ビューで、src/hal_entry.c ファイルをダブルクリックして、ソースファイルを編集します。

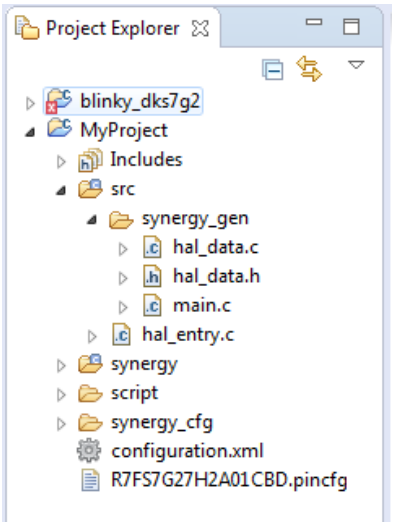


図 52: hal_entry.c

注 : アプリケーションで呼び出すドライバーに必要なすべての構成構造体は、src/synergy_gen/hal_data.c で初期化されます。

注 : ディレクトリ src/synergy_gen のファイルは変更しないでください。これらのファイルは [Generate Project Content] ボタンを押すたびに上書きされます。

5) アプリケーションコードをここに追加します。

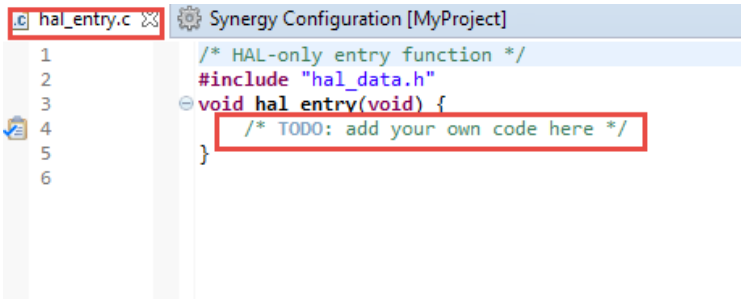


図 53: hal_entry.c にユーザーコードを追加

6) [Project]>[Build Project] をクリックして、プロジェクトをビルドします。発生したビルドエラーを解決します。

次のチュートリアルでは、上記の手順を実行してアプリケーションコードを追加する方法について説明しています : チュートリアル : Using HAL Drivers - Programming the WDT

WDT の例は HAL レベルアプリケーションで、RTOS を使用しません。各モジュールのユーザーガイドには、hal_entry.c に追加できる基本的なアプリケーションコードも含まれます。

3.1.10.2 ThreadX アプリケーション

ThreadX を使用した RTOS 対応アプリケーションコードを記述するには、次の手順を実行します。

1) [Threads] タブを使用してスレッドを追加します。
2) このスレッドの一意の名前を [Properties] ビューで指定します。
3) このスレッドのすべてのドライバーとリソースを構成し、不足している割り込みやドライバーのような、ISDE によってフラグされたすべての依存関係を解決します。
4) スレッドオブジェクトを構成します。
5) 各スレッドオブジェクトの一意の名前を [Properties] ビューで指定します。
6) 必要に応じてさらにスレッドを追加し、手順 1 から 5 を繰り返します。
7) [Synergy Project Editor] で [Generate Project Content] ボタンを押します。
8) [Project Explorer] ビューで、src/my_thread_1_entry.c ファイルをダブルクリックして、ソースファイルを編集します。

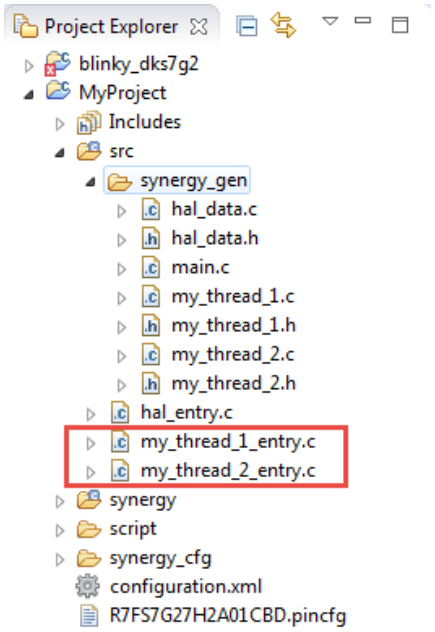


図 54: RTOS アプリケーション用 ISDE 生成ファイル

注 : アプリケーションで呼び出すドライバーに必要なすべての構成構造体は、synergy_gen/my_thread_1.c と my_thread_2.c で初期化されます。

注 : ディレクトリ src/synergy_gen のファイルは変更しないでください。これらのファイルは [Generate Project Content] ボタンを押すたびに上書きされます。

9) アプリケーションコードをここに追加します。

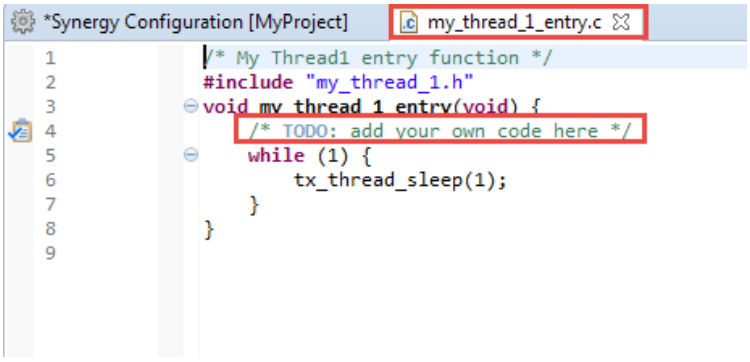


図 55: my_thread_1_entry.c へのユーザーコードの追加

10) 次のスレッドで手順 1 から 9 を繰り返します。

11) [Project]>[Build Project] をクリックして、プロジェクトをビルドします。発生したビルドエラーを解決します。

3.1.11 プロジェクトのデバッグ

プロジェクトの構築がエラーなしで完了すると、デバッガーを使ってアプリケーションをボードにダウンロードして、実行できます。

アプリケーションをデバッグするには、次の手順を実行します。

1) デバッグアイコンの横にあるドロップダウンメニューで **[Debug Configurations]** を選択します。

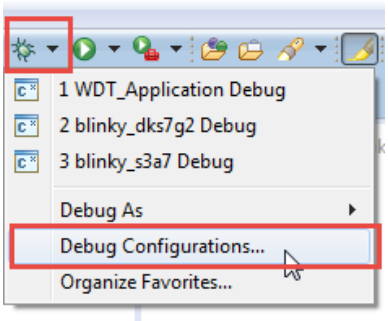


図 56: [Debug Configuration] ダイアログの呼び出し

2) [Debug Configuration] ビューで [MyProject Debug] としてリストされているプロジェクトをクリックします。

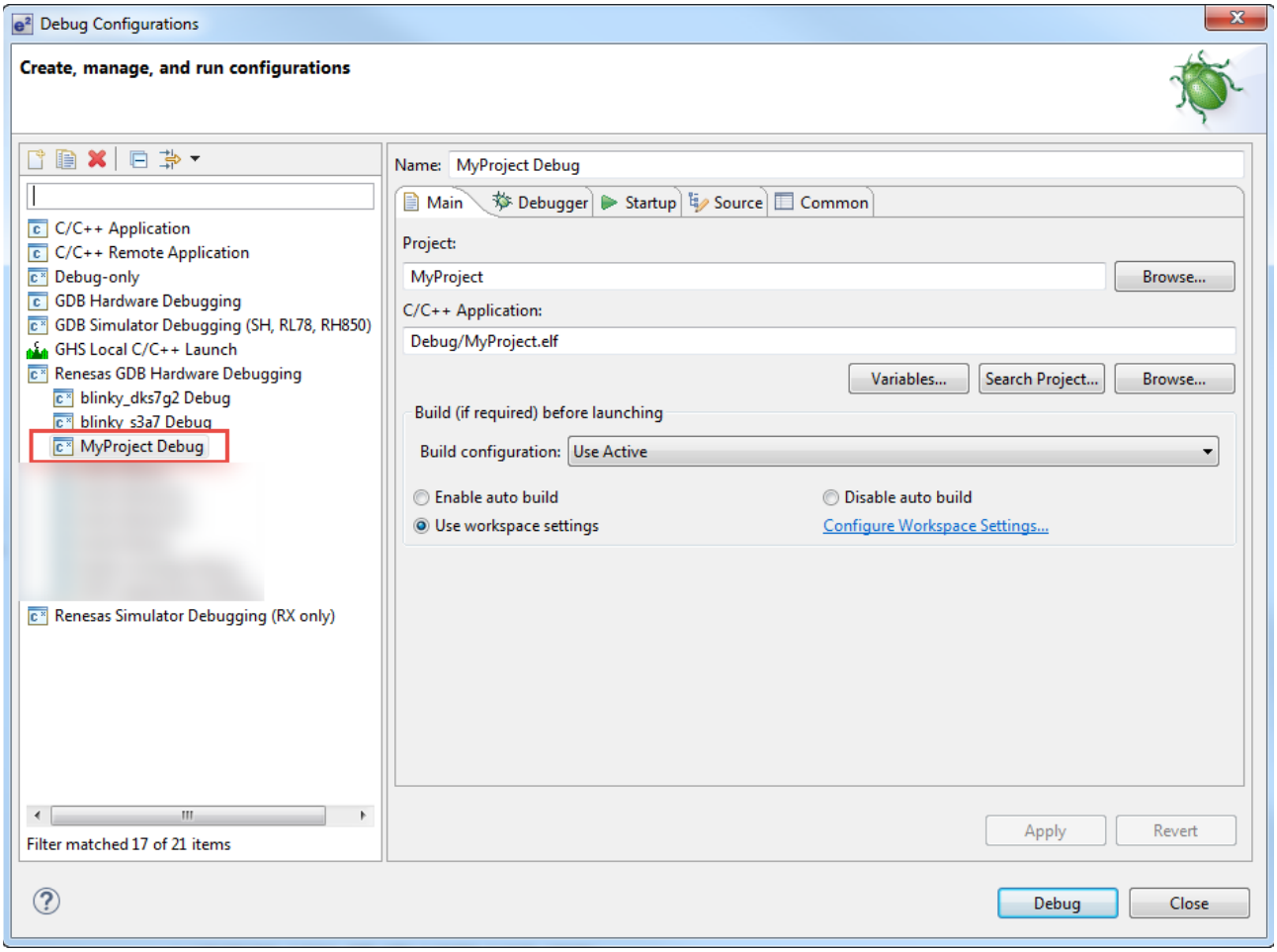


図 57: [Debug Configuration]

3) スタンドアロンの SEGGER J-Link デバッガまたは SEGGER J-Link On-Board（すべての Synergy DK および SK に含まれる）を使ってボードをパソコンに接続し、[Debug] をクリックします。

注 : J-Link の使用方法とボードをパソコンに接続する方法については、Synergy キットに含まれるクイックスタートガイドを参照してください。

3.1.12 Synergy プロジェクトでの TraceX の使用

注 : Synergy プロジェクトで TraceX を使用する前に、Synergy Gallery から TraceX ソフトウェアをダウンロードしてインストールする必要があります。

TraceX はホストベースの分析ツールで、リアルタイムのシステムイベントをグラフィカルに表示できます。TraceX はターゲットデバイスのデータを収集して、検査と分析のためにデータを表示します。Synergy デバイスの TraceX バージョンが Synergy Gallery ウェブサイトからダウンロードでき、Synergy ライセンスと共に使用できます。

TraceX を使用するには、次の手順を実行します。

1) [Threads] タブを使用して、スレッドのプロパティウィンドウの TraceX を有効にします。TraceX バッファーのデフォルト名は g_tx_trace_buffer のままにします。

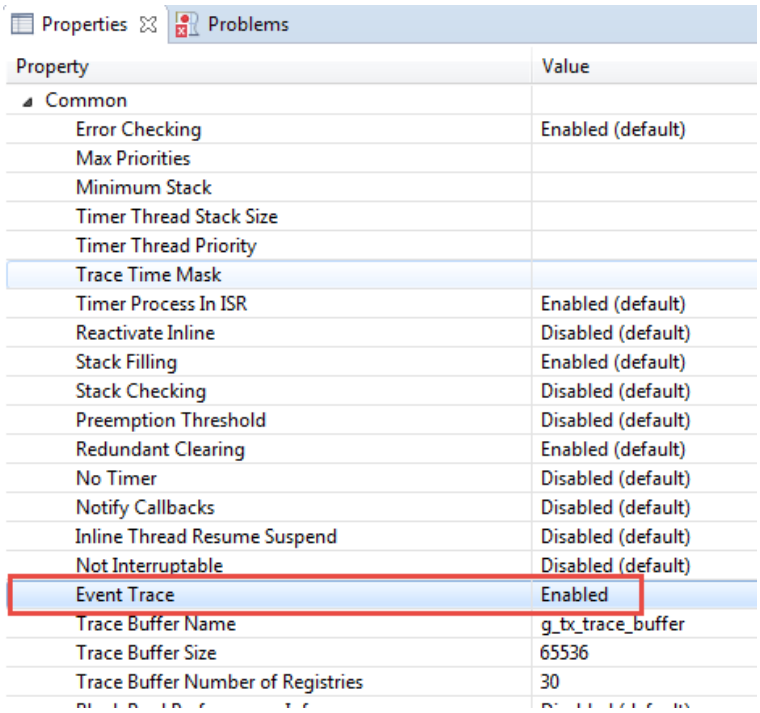


図 58: TraceX の設定

2) [Window]>[Preferences]>[C/C++]>[Renesas]>[TraceX] でTraceXアプリケーションへのパスを設定します。

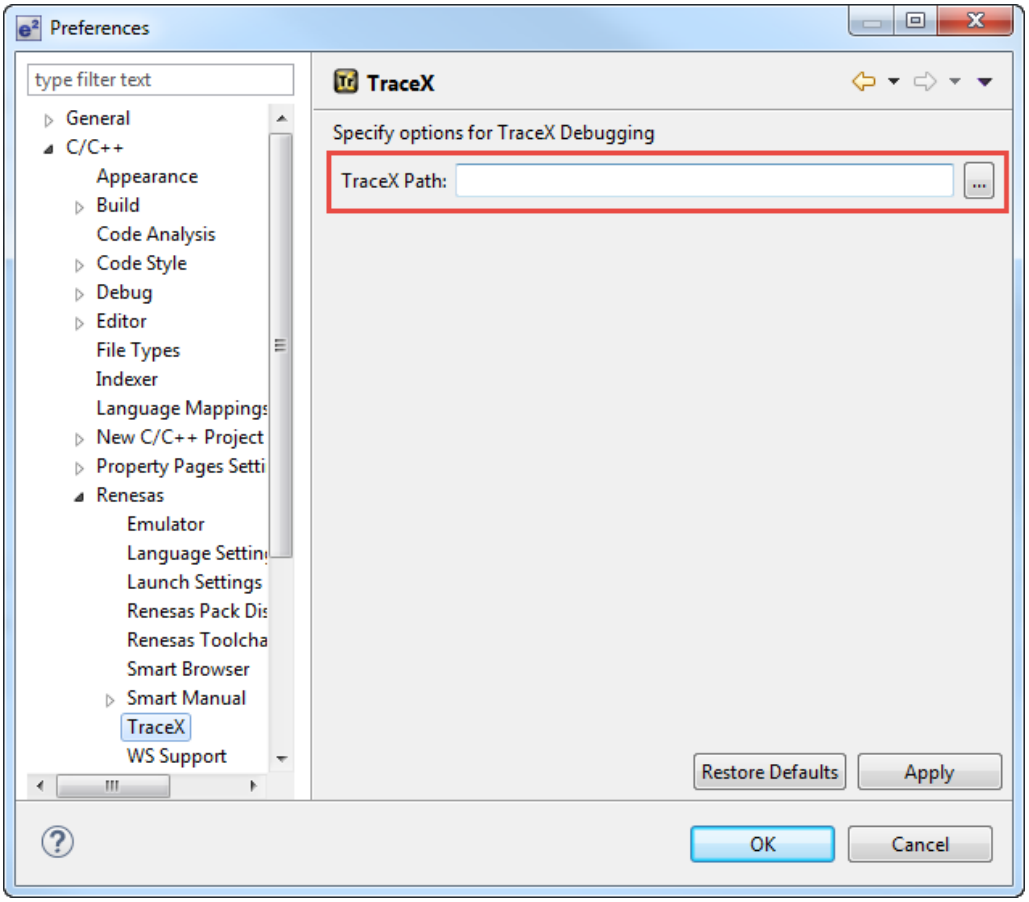


図 59: ISDE TraceX パス

3) [Run]>[TraceX] で [Launch TraceX Debugging] を選択します。

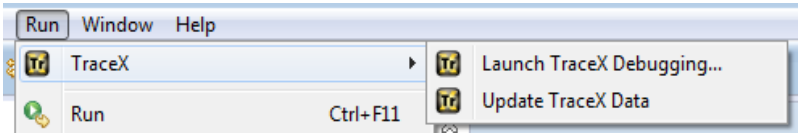


図 60: ISDE TraceX 起動

4) [TraceX Debugging] ウィンドウで、[Buffer Start Address] に &g_tx_trace_buffer. を設定します。

5) [TraceX Debugging] ウィンドウで、[Buffer Size (bytes)] を手順 1 のプロパティウィンドウで選択したバッファサイズに設定します。デフォルトは 65536 です。

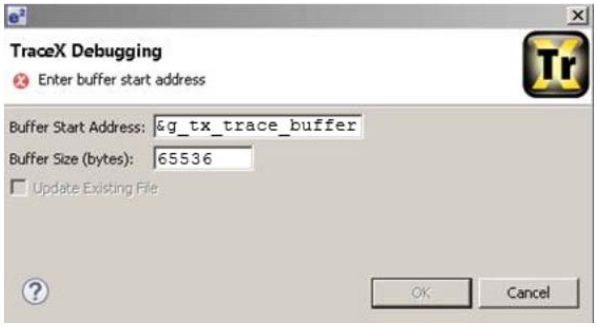


図 61: ISDE TraceX デバッグ

6) [OK] をクリックします。

Express Logic 社のウェブサイトに TraceX の詳しい使用方法が記載されています。

3.1.13 ツールチェーン設定の変更

使用するツールチェーンを変更することが必要な場合があります。(たとえば、コンパイラの最適化レベルを変更したり、リンカーにライブラリを追加する場合などです。) そのような変更を行うには、ISDE で、プロジェクトが選択された状態でメニューから [Project] > [Renesas Tool Settings] を選択します。次のスクリーンショットは、GNU Arm ツールチェーンの設定ダイアログを示しています。このダイアログの外観は、使用するツールチェーンによって若干異なります。

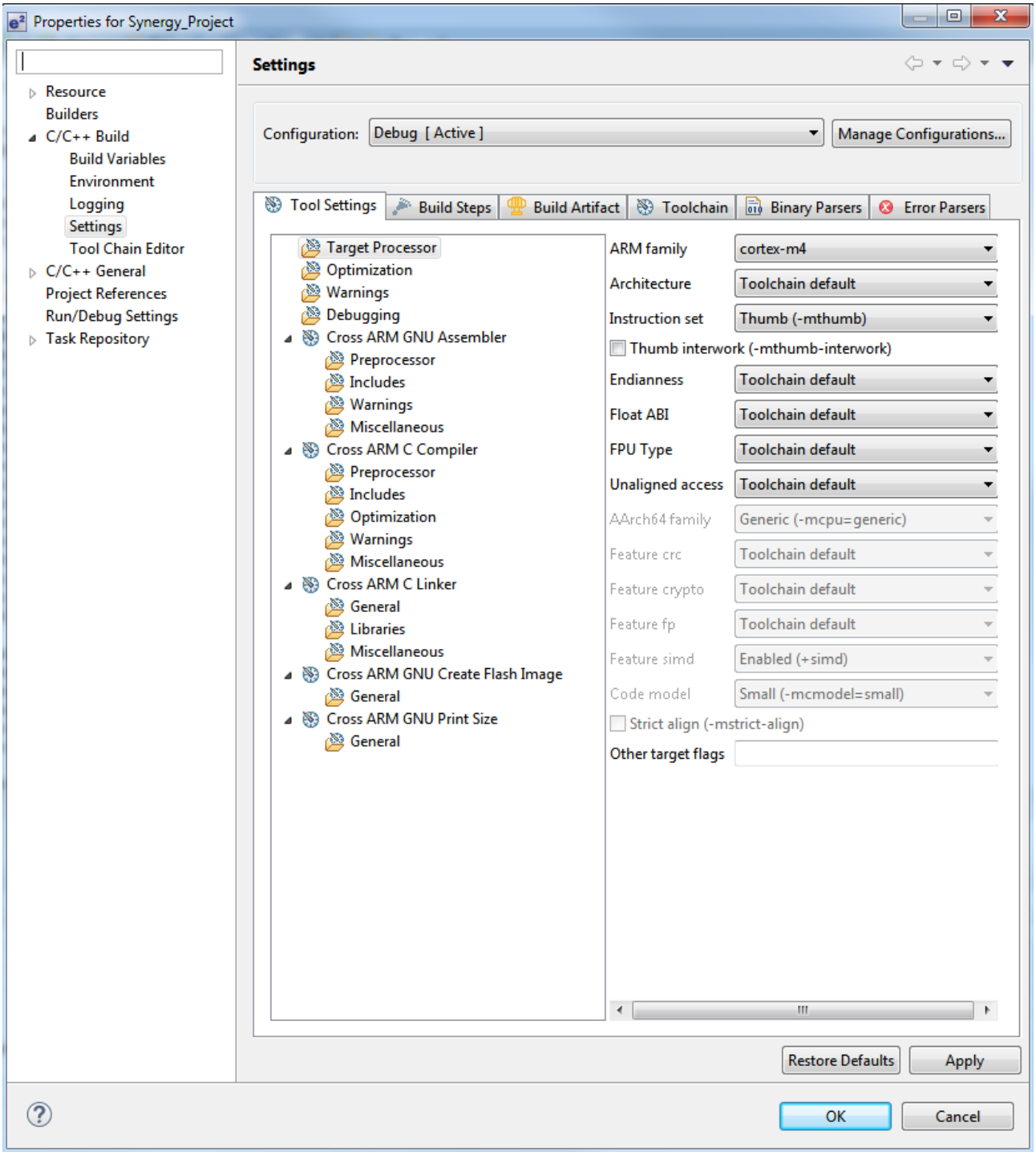


図 62: ISDE プロジェクトツールチェーン設定

設定のスコープはプロジェクトのスコープで、変更対象のプロジェクトのみで有効です。

リンカーの、各種メモリセクションの場所を制御する設定は、使用するデバイスに固有のスクリプトファイルに格納されます。作成されたスクリプトファイルはプロジェクトに含まれ、スクリプトフォルダーに格納されます（たとえば、/script/S7G2.ld)。

3.1.14 カスタムボードパック（「カスタム BSP」）の作成

ほとんどの Synergy ユーザーは、標準の Renesas Synergy ボードのいずれか（たとえば、DK、SK）で開発を開始します。しかし、（全員ではないにしても）そのほとんどが、いずれは Synergy デバイスを基に独自のカスタムボードを開発するようになります。そのときには、自身のカスタムボードに基づいて新しい Synergy プロジェクトを作成できるよう、独自のカスタムボードパックを作成したいと思うことも出てくるでしょう。ここではその方法を説明します。

カスタムボードパックの作成に必要な手順は、次のとおりです。

1) 新しい Synergy プロジェクトを作成します（カスタムボードパックを作成するボードに最も近いボードを基にします)。
2) 自身のカスタムボードに準じて、BSP、クロック、ピンの設定をカスタマイズします。
3) いくつかの主要ファイルを編集して、ボード名をカスタマイズします。
4) プロジェクトをビルドします（ビルドするカスタムボードに基づいたプロジェクトであることを確認します)。
5) User Pack Exporter を使用して、カスタムボードパックを作成します。

この時点で、新しいカスタムボードパックのテストとして、自身のカスタムボードを基に新しいプロジェクトを作成してみます。

詳しい手順

特定バージョンの SSP（SSP v1.2.0-b.1 など）のカスタムボードパックを作成するには、以下の手順に従います。

注：本稿執筆時点では、IAR Embedded Workbench® for Renesas Synergy™ にこの機能は存在しません。

1) e2 studio v5.2.1（またはそれ以降）で、BSP プロジェクトテンプレートを使用して、ユーザーがインストールした SSP バージョン（SSP v1.2.0-b.1 など）のための新しい Synergy C プロジェクトを作成します。ボードには、利用可能なボードのうち、新しいカスタムボードに最も近いものを選択します。
2) プロジェクトを作成したら、プロジェクト内の既存のピン構成ファイルをコピーして、'MyCustomBoard.pincfg' などの任意の名前に変更し、カスタムピン構成ファイルの名前とします。
3) 次に、以下の手順に従い、[Pins] タブのピンコンフィギュレータを、新しいカスタムピン構成ファイルに切り替えます。
   a) 現在アクティブな .pincfg ファイルの [Generate Data] ボックスをオフにします。
   b) プルダウンメニューでカスタム pincfg ファイルに切り替え、[Generate Data] ボックスをオンにして、データ構造体名「'g_bsp_pin_cfg'.」を入力します。
4) BSP、クロック、ピンのそれぞれのタブで、カスタムボードに合わせて設定をカスタマイズします。
5) [Pins]、[Clocks]、[BSP] の各タブでのカスタマイズを完了した後、[Generate Project Content] をクリックしてそれぞれのファイルをアップデートします。[Save All] をクリックします。

6) ボードのフォルダ名 'synergy/board/<xxx>' を編集し、カスタムボード名に合わせます（'synergy/board/my_custom_board'. など）。

7) ファイル 'synergy_cfg/ssp_cfg/bsp/bsp_board_cfg.h' を編集して、次のように 4 行目をボードパスに置き換えます。

```
#include "../../../synergy/board/<xxx>/bsp.h"
```

上記を以下のように変更します。

```
#include "../../../synergy/board/my_custom_board/bsp.h"
```

[Save All] をクリックします。

8) 次の手順で、ファイル bsp_board_cfg.h のプロパティを「リード専用」に変更し、プロジェクトをビルドするときに上書きされないようにします。

a) bsp_board_cfg.h を右クリックして、[Properties] を選択します。

b) [Resource] をクリックします。

c) [Read-only] ボックスをオンにします。

d) [Apply]、[OK] の順にクリックします。

9) プロジェクトをビルドします。（この手順で、新しく作成したボードパックに基づくプロジェクトが正常にビルドされることを確認します。）

10) 新しいカスタムボードパックのエクスポート処理を開始するには、プロジェクトを右クリックして、**[Export Synergy User Pack]** を選択します。

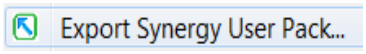


図 63: [Export Synergy User Pack] ウィザードの起動

[Export Synergy User Pack] ウィザードが起動します。

11) [Create a board pack] を選択します。

12) 新しいパックの場所を設定します。

- e2 studio を使用する場合は、場所として <e2_studio_base_dir>/internal/projectgen/arm/Packs/ を指定します。
- IAR Embedded Workbench for Synergy を使用する場合は、場所として <SSC_base_dir>/internal/projectgen/arm/Packs/ を指定します。

13) パックの詳細を指定するには、[PackVendor]、[Pack Name]、[Pack Version] の情報を入力します。パックのバージョンは、このパックとともに使用する SSP バージョンに一致させる必要があります。その結果、次のような画面になります。

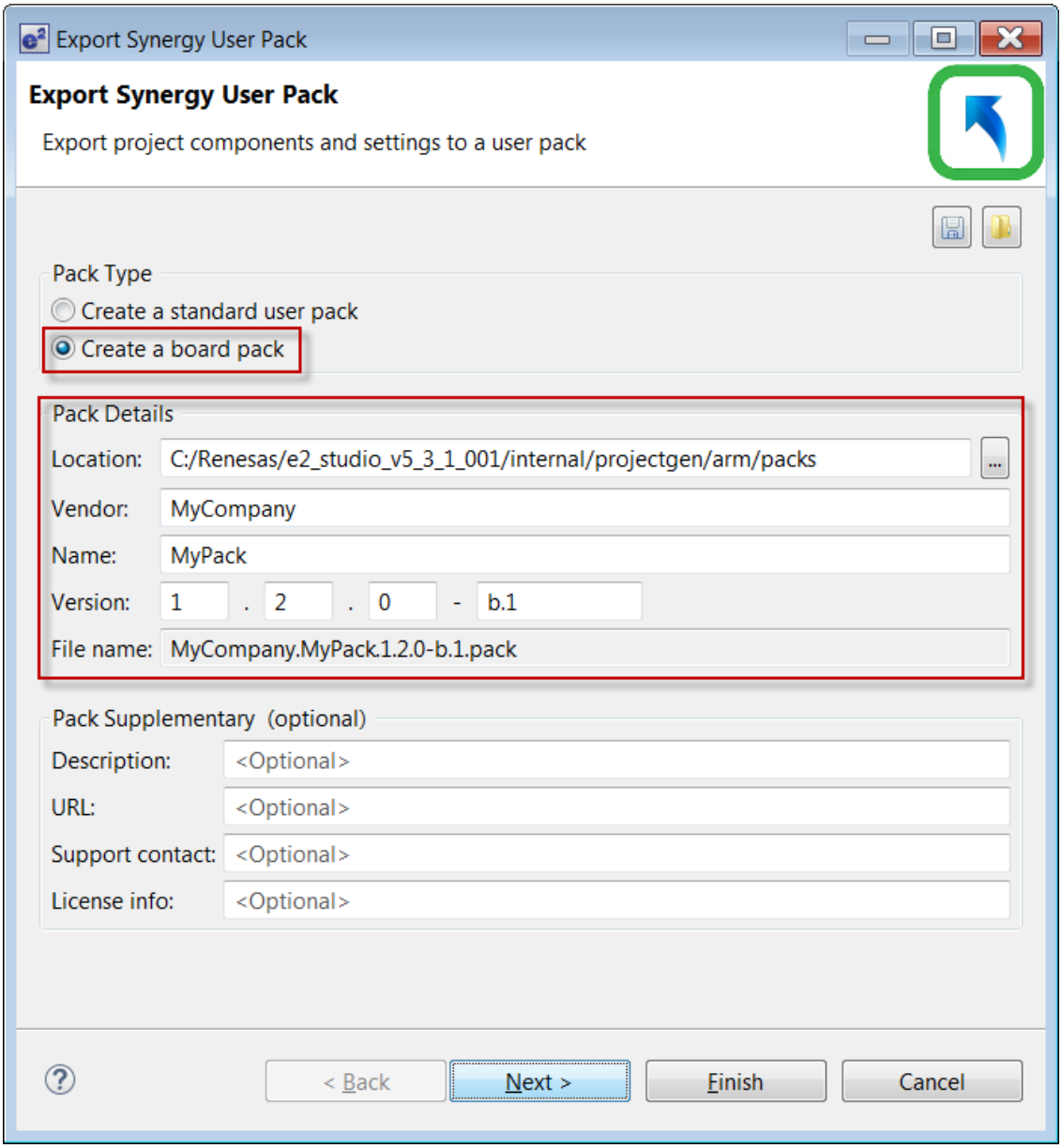


図 64: Synergy ユーザーパックのエクスポート画面 1

[Next] をクリックします。

14) 次の画面で、ボードコンポーネントの詳細を指定します。[Board Component Name]（[Sub-group] フィールド）、[Board Component Variant]（オプション）、短い [Board Component Description]（オプション）、[Board Component Version]（パックのバージョンと同じバージョンを使用）を入力します。

15) [Board Name] を入力します。

注：このボード名は、synergy/board/<board_folder_name> にあるボードフォルダ名に一致させる必要があります。

その結果、次のような画面になります。

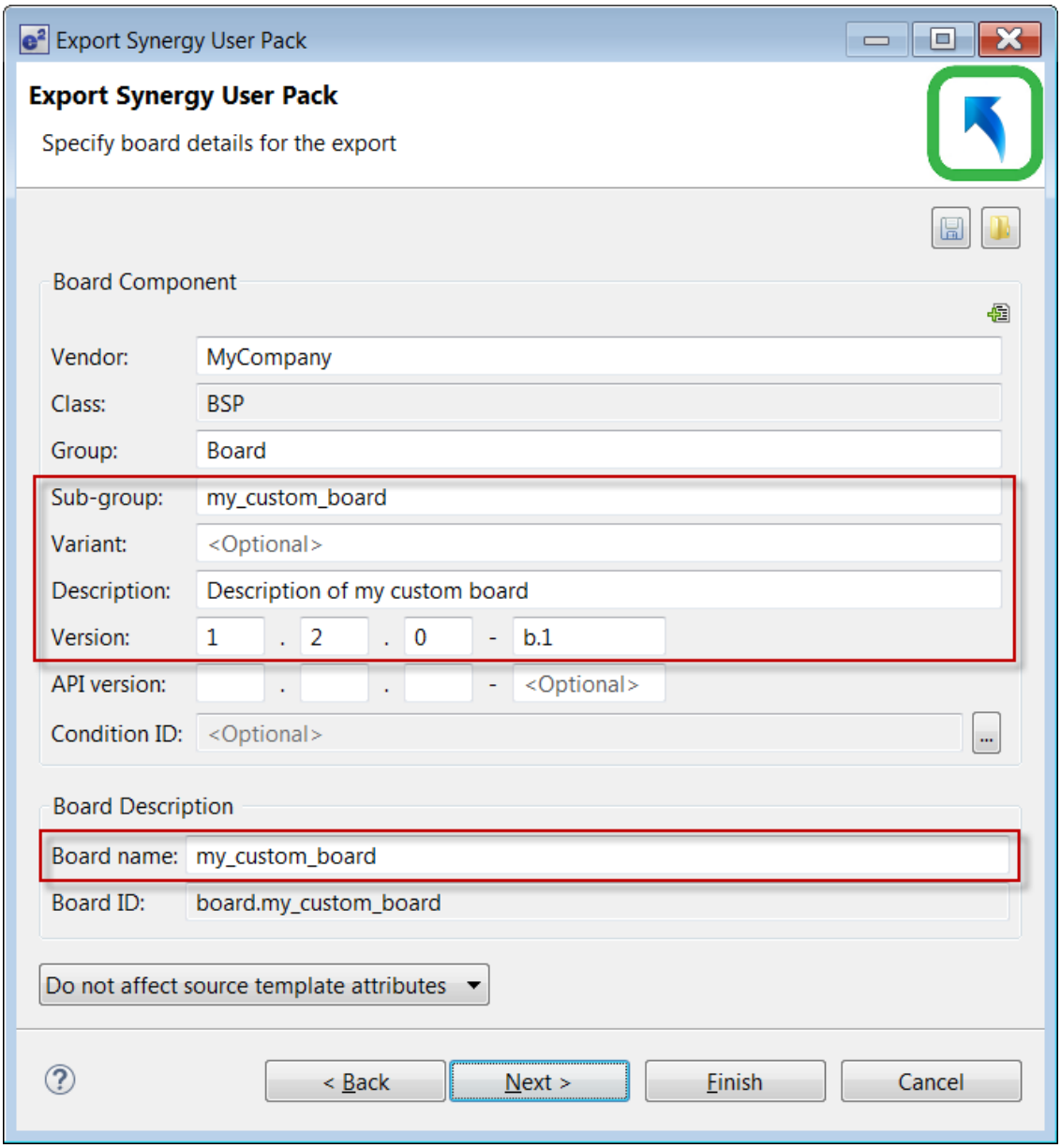


図 65: Synergy ユーザーパックのエクスポート画面 2

16) ウィザード画面右上の「+」アイコンを使用して、パックに含めるファイルを選択します。自身のカスタムピン構成ファイルとカスタムボードソースファイルをもれなく選択し、[Add] をクリックします。

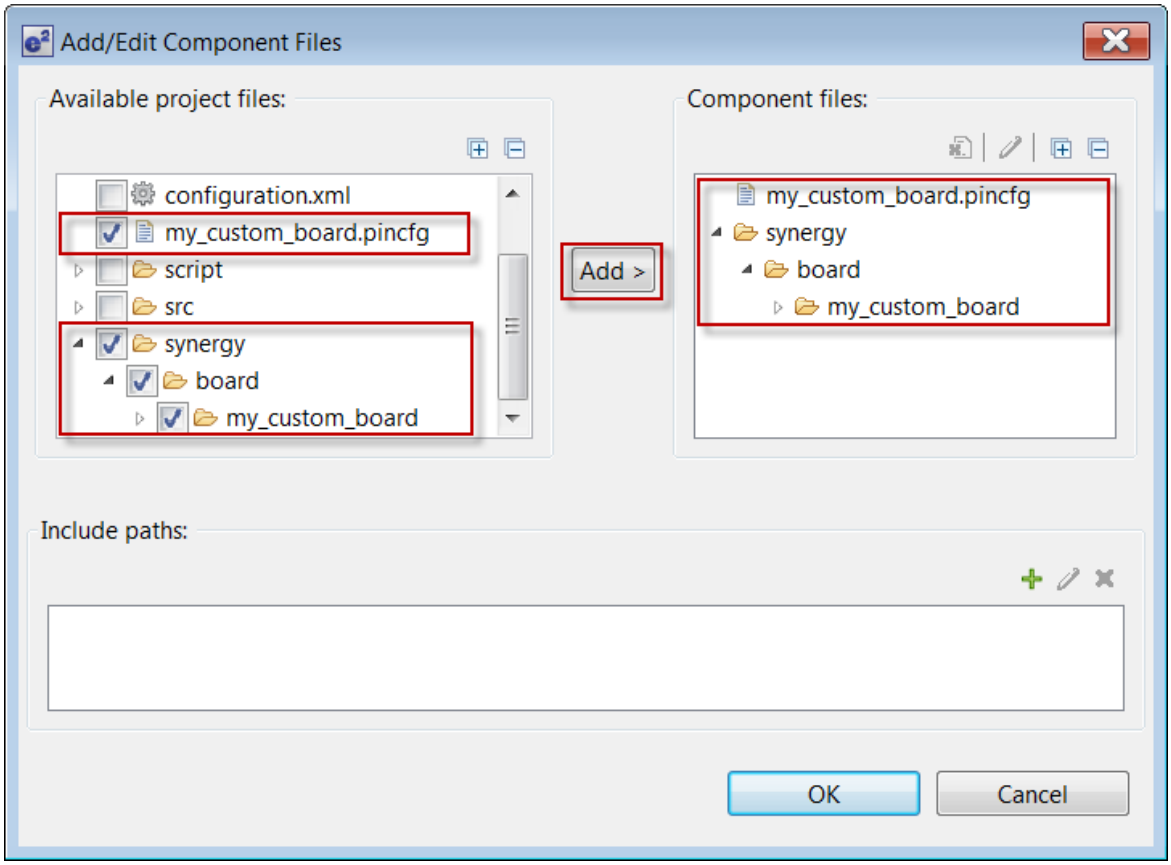


図 66: コンポーネントファイルの追加と編集

17) [Component files] ペインで *.pincfg ファイルを選択し、鉛筆アイコンをクリックしてファイルプロパティを次のように編集します。

- [Category] プロパティを source に変更します。
- [Attribute] プロパティを template に変更します。

[OK] をクリックします。

18) [OK]、[Finish] をクリックすると、指定した場所にカスタムボードパックが作成されます。

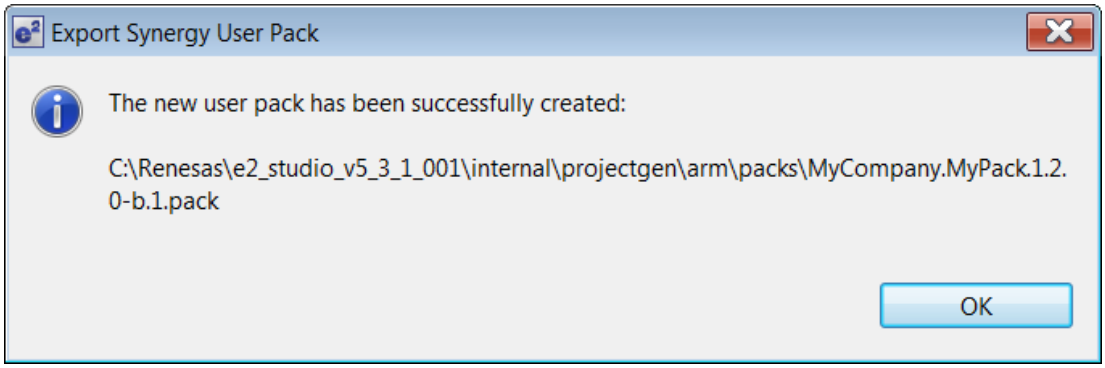


図 67:　正常に作成されたカスタムボードパック

新しいカスタムボードパックをテストするには、新しい Synergy C プロジェクトを作成し、ボードを選択する際に、リストから自身のカスタムボードを選択します。

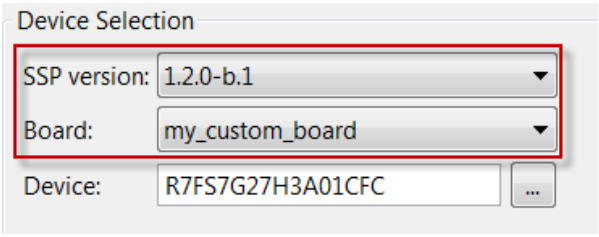


図 68:　カスタムボードに基づいた新しいプロジェクトの作成

プロジェクトを作成したら、Synergy Project Editor を開き、BSP、クロック、ピンの各タブでカスタマイズの内容を確認します。また、コンポーネントのタブも確認します。自身のカスタムボードが、BSP のボードコンポーネントとして選択されているはずです。コンポーネントの上にマウスカーソルを合わせると、カスタムボードパックファイルが表示されます。

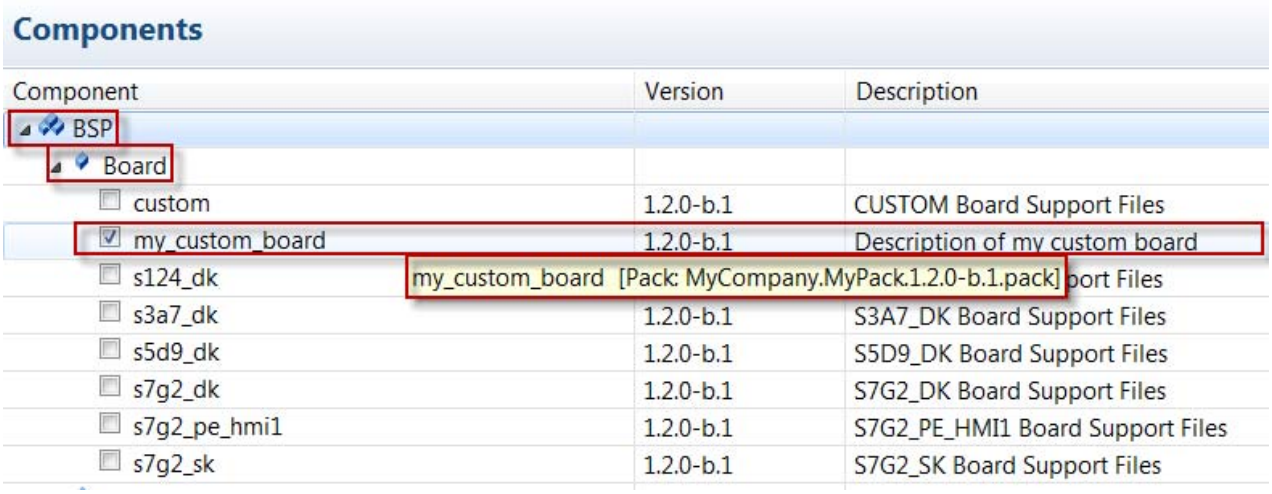


図 69: [Components] タブでのカスタムボードパックの表示

アプリケーションコードを追加し、カスタムボード上でプロジェクトのビルドとデバッグを行うことができるようになりました。

3.1.15 カスタムユーザーパックの作成

[Export Synergy User Pack] ウィザードでは、非 SSP（つまりカスタム）コンポーネント、RTOS スレッド、SSP コンポーネント設定のような項目を含むカスタムユーザーパックを作成することもできます。

次の要素はエクスポートできません。

- SSP コンポーネントと SSP ソースファイル（たとえば、<Project>\synergy\ssp）
- SSP 生成ファイル（たとえば、<Project>\synergy_cfg\ssp_cfg)

カスタムユーザーパックを作成するには、カスタムコンポーネントを含むプロジェクトで右クリックし、Synergy User Pack Exporter を呼び出します。

最初の画面で、パックの種類として [Create a standard user pack] を選択し、パックの詳細を入力します。[Next] をクリックします。

2 番目の画面で、エクスポートするプロジェクトコンポーネントを選択（または新規作成）し、コンポーネントごとにエクスポートするスレッド構成を選択します。[Next] をクリックします。

3 番目の画面で、エクスポートする既存のパック条件を選択するか（オプション）、新しい条件を作成するか（オプション）、それらの両方を行います。[Finish] をクリックすると、指定の場所にカスタムユーザーパックが作成されます。

e2 studio はカスタムユーザーパックを読み取り、パック内のコンポーネントを抽出して、[Components] タブにコンポーネントを表示します。[Components] タブでいずれかのコンポーネントを選択すると、それがプロジェクトに追加されます。

3.1.16 e2 studio ISDE 使用上の注意

3.1.16.1 SSP リリースの処理

Synergy Gallery ウェブサイトでは SSP のさまざまなリリースパックを公開しています。

SSP パックには 2 種類あります。

1) X.Y.0 – 完全な SQA を含む生産リリース。X.Y.0 パックだけが含まれます。
2) X.Y.Z – リリースされた X.Y.0 バージョン（X= メジャー、Y= マイナー、Z= パッチ）に基づいたパッチリリース。X.Y.Z パックで追加または更新されたモジュールだけが含まれます（完全な X.Y.0 パックは含まれません）。

X.Y.0 生産リリースに加えて、複数の X.Y.Z パッチリリースをインストールすることができます。

たとえば、次の SSP バージョンをインストールすることができます。

- SSP v1.1.0: 生産リリース
- SSP v1.1.1: 最初のパッチリリース
- SSP v1.1.2: 2 回目のパッチリリース
- SSP v1.1.3: 3 回目のパッチリリース

注 : SSP パッチリリースインストーラにより以前のパッチリリースや生産リリースがインストールされることもあります。詳細については、SSP インストールログを参照してください。

Synergy Project Generator または Synergy Project Editor の [BSP] タブで SSP バージョン（SSP v1.1.3 など）を選択すると、e2 studio 統合ソリューション開発環境は、プロジェクトに含めるために対応する SSP コンポーネント（入手できる場合）を自動的に選択します。この結果は、[Components] タブでモジュールをルックアップすると確認できます。

同一プロジェクトで使用できるのは、メジャーバージョンとマイナーバージョンが同じコンポーネントのみです。たとえば、次の SSP バージョンをインストールしたとします。

- SSP v1.1.3
- SSP v1.2.0
- SSP v1.2.1

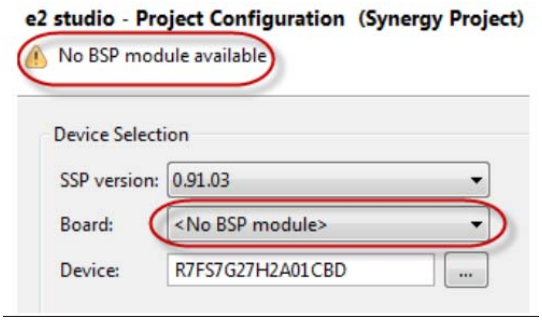


図 70: 複数の SSP バージョンのインストール

このシナリオでは、SSP v1.2.1 プロジェクトを使用している場合、SSP v1.2.1 と SSP v1.2.0 両方のコンポーネントが [Components] タブに表示され、そのプロジェクトで使用できます。ただし、SSP v1.1.3 コンポーネントはフィルタ処理によって除外され、[Components] タブには表示されません。

選択した SSP バージョンで必要な SSP コンポーネントが利用できない場合は、e2 studio 統合ソリューション開発環境により、利用可能なもののなかから最もバージョンの新しい SSP コンポーネントが選択されます。

[Components] タブの SSP コンポーネントが、選択された SSP バージョン（「パッチ」バージョンを含む）と正確に一致しない場合、[BSP] タブの [SSP Version] フィールドの横に警告アイコンが表示されます。

次の例では、SSP v1.1.3 が選択されています。しかし、SSP v1.1.3 はパッチリリースです。このため、プロジェクトには以前のパッチリリース（SSP v1.1.1 など）や生産リリース（SSP v1.1.0）に由来し、SSP v1.1.3 には存在しないモジュールが含まれている可能性があります。この警告は情報提供のみを目的としており、ユーザーのプロジェクトがビルドされない、または正常に実行されないことを示すものではありません。

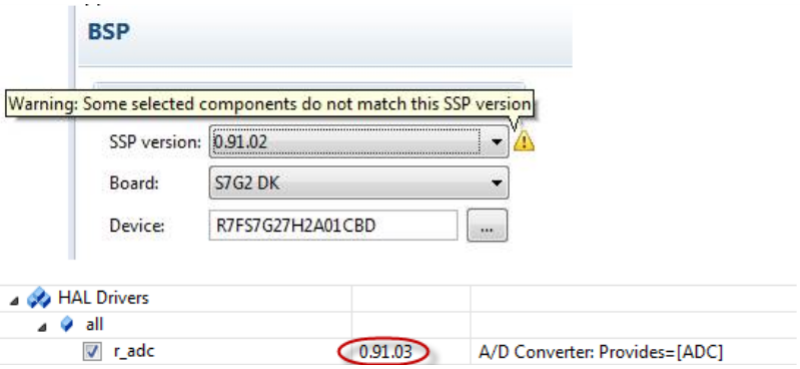


図 71: e2 studio コンポーネントで複数の SSP バージョンをインストールした場合の不一致の発生

最後にインストールされたパッチリリースより前の SSP バージョンを選択すると（たとえば、SSP v1.1.3 もインストールしている状況で SSP v1.1.1 を選択）、e2 studio 統合ソリューション開発環境は、たとえ特定の SSP コンポーネントの新しいバージョン（つまり SSP v1.1.3）が利用できる場合でも、利用可能な SSP v1.1.1 コンポーネントをプロジェクトに含めます。この挙動は、以前の SSP バージョンをユーザーが明確に指定したことによるものです。

3.1.16.2 ThreadX ソースコードの追加

[Threads] タブを使ってプロジェクトに ThreadX ソースコードを追加する手順は次のとおりです。

1) [Threads] ペインで [HAL/Common] をクリックします。[Modules] ペインが [HAL/Common Modules] に変わります。
2) [New]>[Framework]>[RTOS]>[ThreadX Source] を選択してください。
3) [Properties] ビューを使用して、ThreadX RTOS プロパティを構成します。
4) [Generate Project] をクリックします。

e2 studio ISDE で次のディレクトリーに ThreadX ソースコードを抽出します。

注 : 適応するライセンスを用いて ThreadX ソースコードを修正できます。詳細はライセンスの章を参照してください。ThreadX ソースコードの抽出はプロジェクトのコンパイル時間を増やすことになることに留意してください。

3.2 チュートリアル : Your First Synergy Project - Blinky

3.2.1 チュートリアル Blinky

このチュートリアルの目的は、e^2 studio を使用した単純なアプリケーションの作成と、そのアプリケーションの Synergy ボード上での実行手順を通じて、Synergy プラットフォームに素早く精通することです。

3.2.2 Blinky の目的

3.2.3 前提条件

このチュートリアルを実行するには、以下のものが必要です。

- Windows ベースの PC
- e^2 studio
- Synergy Software Package
- Synergy ボードキット

3.2.4 Blinky 用の新規プロジェクトの作成

Synergy プロジェクトの作成と設定は、アプリケーションを作成するための最初のステップです。ベース SSP パックには、シンプルですべての Renesas Synergy ボードで動作する、作成済みの Blinky サンプルアプリケーションが含まれています。

Synergy プロジェクトを作成するには以下の手順に従います。

1) e^2 studio ISDE で、[File] > [New] > [Synergy Project] の順にクリックします。
2) この新しいプロジェクトに名前を付けます。このチュートリアルでは Blinky という名前を使用します。
3) ライセンスウィンドウが空白の場合はライセンスファイルを特定します。このバージョンのプラットフォームのライセンスファイルは、<ISDE ベースディレクトリ > 内の ISDE\internal\projectgen\arm\Licenses\SSP_License_Example_EvalLicense_<rev>.xml にあります。
4) [Next] をクリックします。[Project Configuration] ウィンドウに選択内容が表示されます。

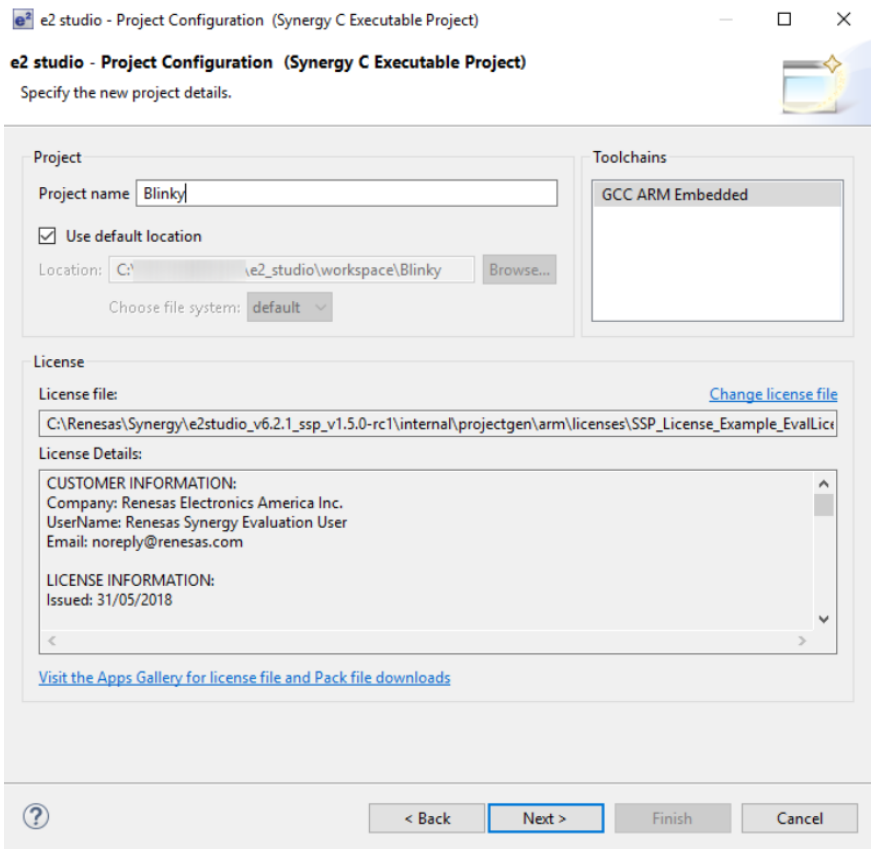


図 72: e² studio ISDE の [Project Configuration] ウィンドウ（パート 1）

5) ボードサポートパッケージを選択するため、[Device Selection] ドロップダウンリストから使用するボード名を選択し、[Next] をクリックします。

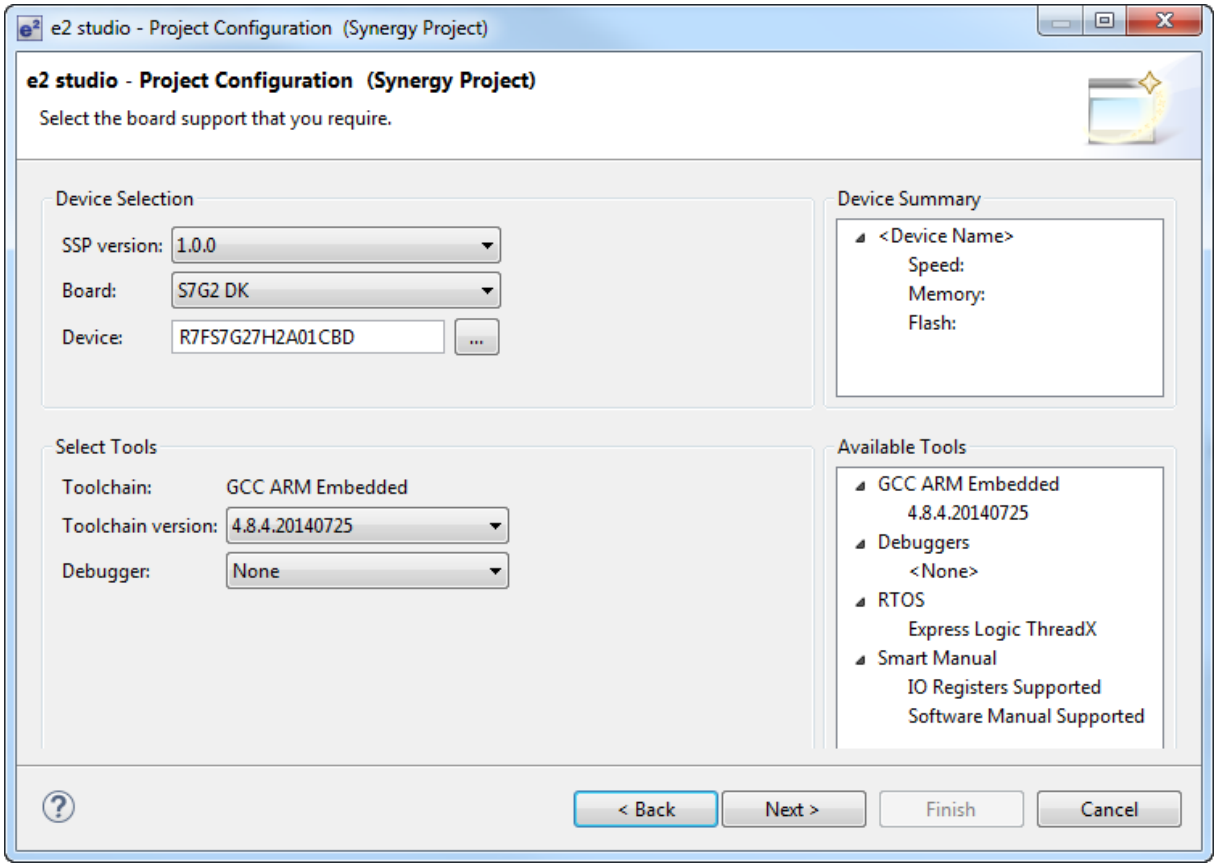


図 73: e² studio ISDE の [Project Configuration] ウィンドウ（パート 2）

6) 使用するボード用の Blinky テンプレートを選択し、[Finish] をクリックします。

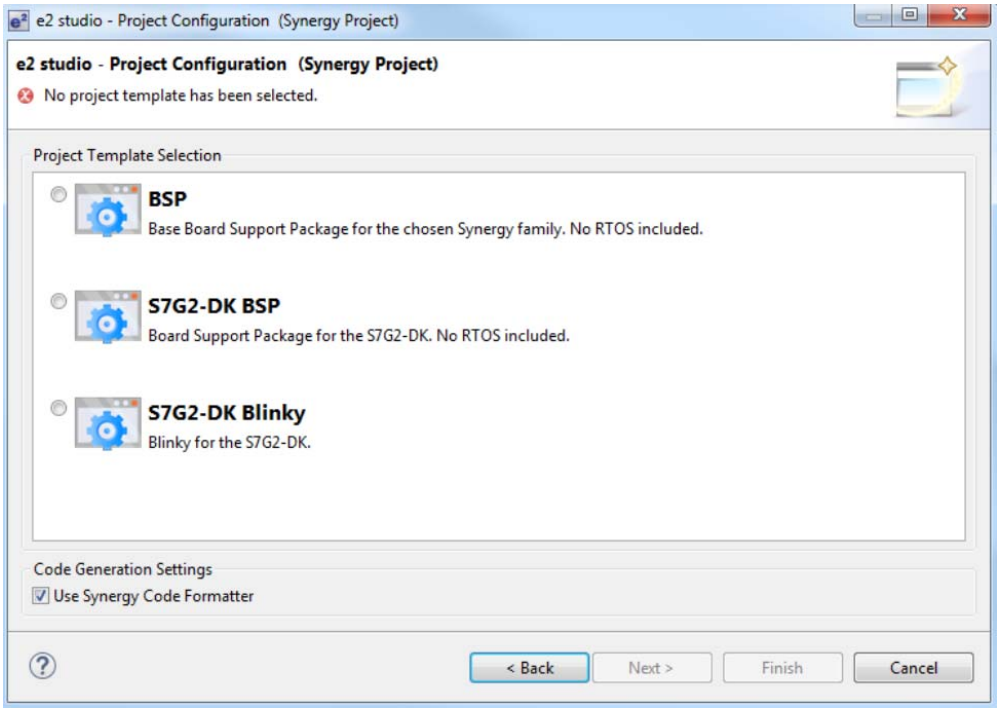


図 74: e² studio ISDE の [Project Configuration] ウィンドウ（パート 3）

プロジェクトが作成されると、プロジェクトの名前が ISDE の [Project Explorer] ウィンドウに表示されます。[Project Configuration] ウィンドウの右上隅にある [Generate Project Content] ボタンを押し、ボード固有のファイルを生成します。

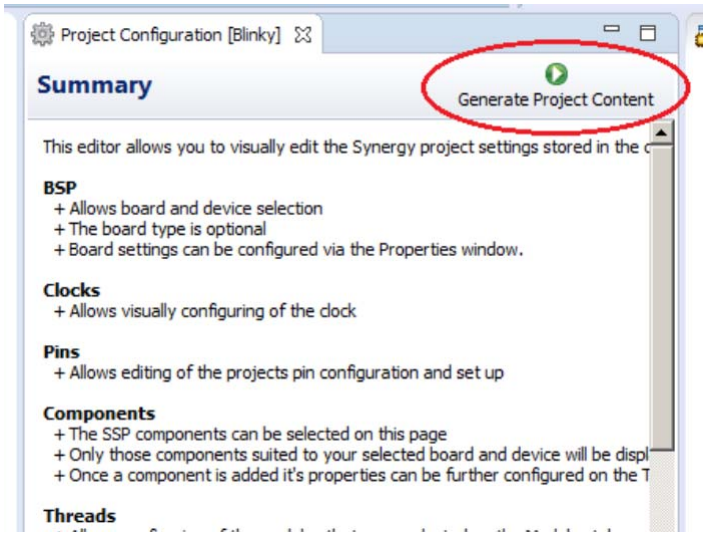


図 75: e² studio ISDE の [Project Configuration] タブ

これで新しいプロジェクトが作成、設定され、ビルドの準備ができました。

3.2.4.1 Blinky の設定に関する詳細

[Generate Project Content] ボタンは、構成ヘッダーファイルを作成し、テンプレートからソースファイルをコピーして、通常は [Project Configuration] 画面の状態に基づいてプロジェクトを構成します。

たとえば、[Components] タブのモジュールの横にあるチェックボックスをオンにし、[Generate Project Content] ボタンを押すと、そのモジュールをプロジェクトに追加するために必要なすべてのファイルがコピーまたは作成されます。その同じチェックボックスをオフにすると、それらのファイルが削除されます。

3.2.4.2 Blinky のクロックの設定

Blinky テンプレートを選択することで、ISDE により Blinky アプリケーション用にクロックが設定されます。Blinky のクロック構成は、ISDE クロック構成タブに表示されます（クロックの設定を参照）。Blinky のクロック構成は、BSP クロック構成ファイルに保存されます（BSP のクロック設定を参照）。

3.2.4.3 Blinky のピン設定

Blinky テンプレートを選択することで、LED1 を切り替えるために使用する GPIO ピンが、ISDE により Blinky アプリケーション用に設定されます。ISDE のピン構成タブには、Blinky アプリケーション用のピン構成が表示されます（ピンの設定を参照）。Blinky のピン構成は、BSP 構成ファイルに保存されます（BSP のピン設定を参照）。

3.2.4.4 Blinky コンポーネント用のパラメータの設定

Blinky プロジェクトは、ISDE コンポーネント内の次の HAL コンポーネントを自動的に選択します。

- r_cgc
- r_elc
- r_fmi
- r_ioport

いずれかのコンポーネントの構成パラメータを表示するには、それぞれのドライバーの [HAL] ウィンドウの [Properties] タブを確認します（HAL ドライバーの追加と設定を参照）。

3.2.4.5 main() の場所

main 関数は＜プロジェクト >/src/synergy_gen/main.c にあります。これは、プロジェクト作成段階で生成されるファイルの 1 つであり、hal_entry() の呼び出しのみを含んでいます。生成されるファイルの詳細については、HAL ドライバーの追加と設定を参照してください。

3.2.4.6 Blinky のサンプルコード

Blinky アプリケーションは、hal_entry.c ファイルに格納されます。このファイルは、Blinky プロジェクトテンプレートを選択したときに ISDE によって生成され、プロジェクトの src/ フォルダー内に配置されます。

アプリケーションは次の手順を実行します。

1) BSP HAL 関数を呼び出すことにより、選択されたボードの LED 情報を取得します (R_BSP_LedsGet)。
2) 選択されたボードの LED を制御する GPIO ピンの出力レベル HIGH を定義します。
3) 選択されたシステムクロック速度を取得し、LED のトグルを観察できるようにクロックをスケールダウンします。
4) 次のものを使用して GPIO ピンに書き込むことにより、LED をトグルします。

```
g_ioport.p_api->pinWrite()
```

3.2.5 Blinky プロジェクトのビルド

新しいプロジェクトを [Project Explorer] ウィンドウでハイライトし、ビルドします。

プロジェクトをビルドするには、以下の 3 つの方法があります。

a. メニューバーの [Project] をクリックし、[Build Project] を選択します。

b. ハンマーアイコンをクリックします。

c. プロジェクトを右クリックし、[Build Project] を選択します。

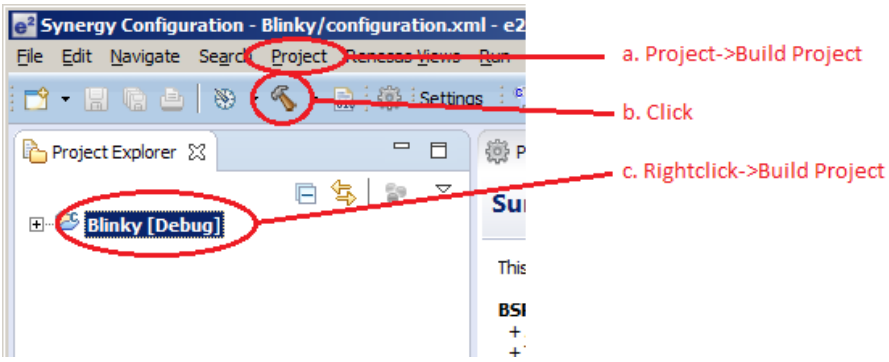


図 76: e² studio ISDE の [Project Explorer] ウィンドウ

ビルドが完了すると、メッセージがビルドの [Console] ウィンドウに表示され、最終的なイメージファイル名と、そのイメージ内のセクションサイズが表示されます。

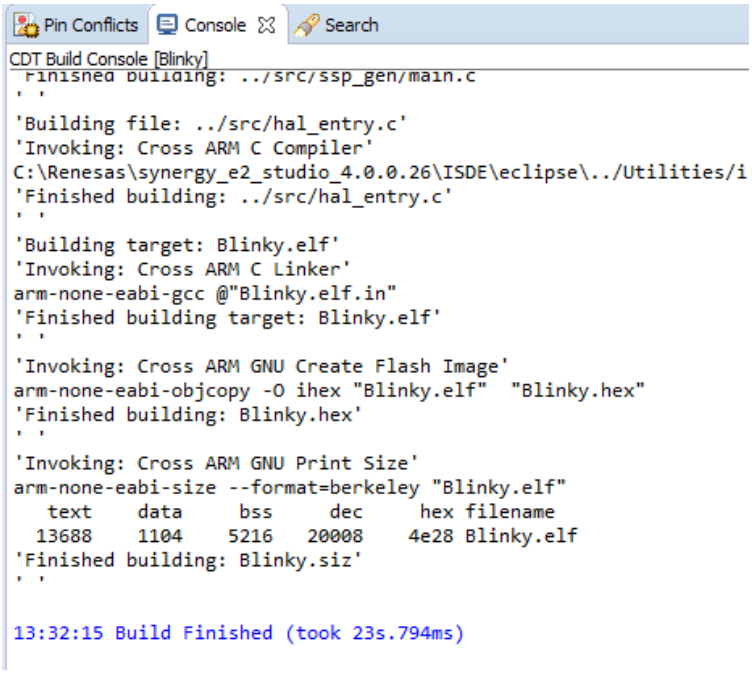


図 77: e² studio ISDE のプロジェクトビルドコンソール

3.2.6 Blinky プロジェクトのデバッグ

3.2.6.1 デバッグの前提条件

ボード上でプロジェクトをデバッグするには、以下のものが必要です。

- ISDE に接続するボード

- ボードと通信するように設定されるデバッガー
- マイクロコントローラにプログラムされるアプリケーション

アプリケーションは、マイクロコントローラの内蔵フラッシュから実行します。アプリケーションを実行またはデバッグするには、まずアプリケーションがマイクロコントローラのフラッシュにプログラムされている必要があります。そのための方法としては、以下の 2 つの方法があります。

- JTAG デバッガー
- UART または USB を通じた組み込みブートローダー

一部のボードにはオンボード JTAG デバッガーが搭載されていますが、他のボードでは、ボード上のヘッダーに接続された外部 JTAG デバッガーが必要です。

ボードのユーザーマニュアルで、JTAG デバッガーを ISDE に接続する方法を確認してください。

3.2.6.2 デバッグ手順

Blinky アプリケーションをデバッグするには、以下の手順を実行します。

1) プロジェクト用にデバッガを設定するため、[Run] > [Debugger Configurations] をクリックします。

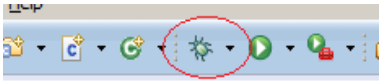


図 78: e² studio ISDE のデバッグアイコン

または、バグアイコンの横にあるドロップダウンメニューで [Debugger Configurations] を選択します。

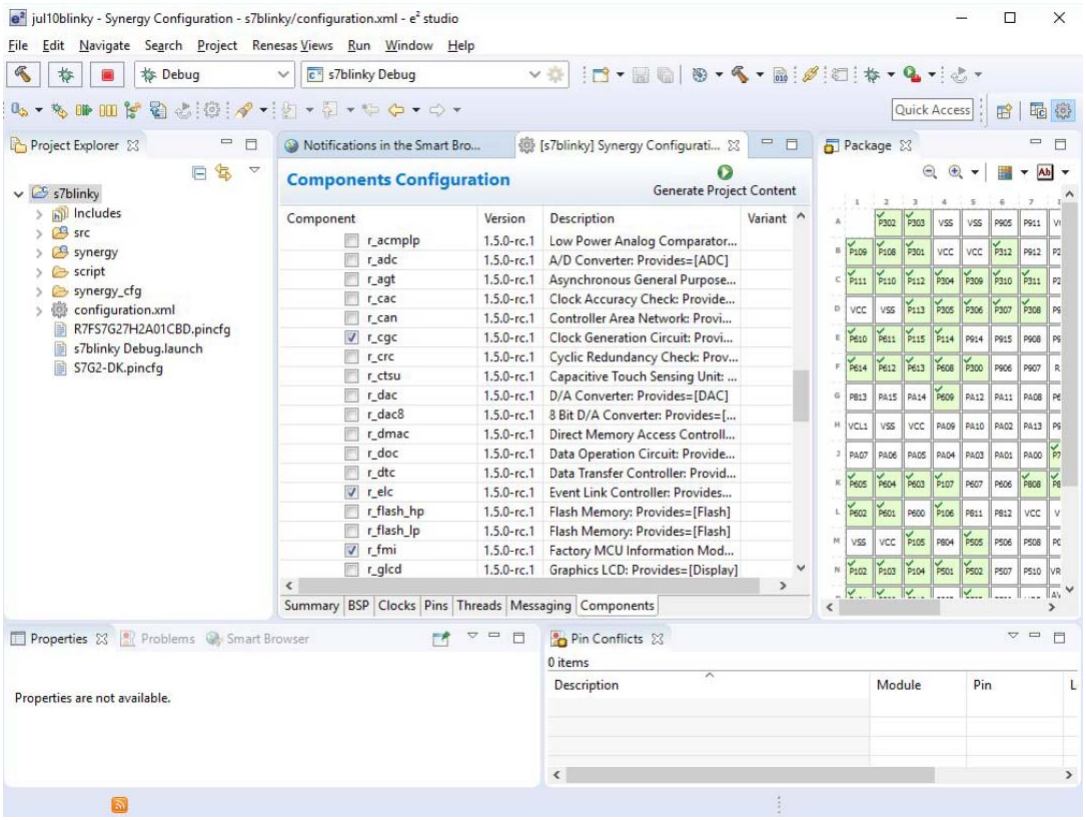


図 79: e² studio ISDE の [Debugger Configuration] ウィンドウ

2) ウィンドウでデバッガー設定を選択します。デバッガー設定が表示されていない場合は、ウィンドウの左上隅にある [New] アイコンをクリックして作成する必要があります。選択すると、[Debug Configuration] ウィンドウに、Blinky プロジェクトのデバッグ設定が表示されます。

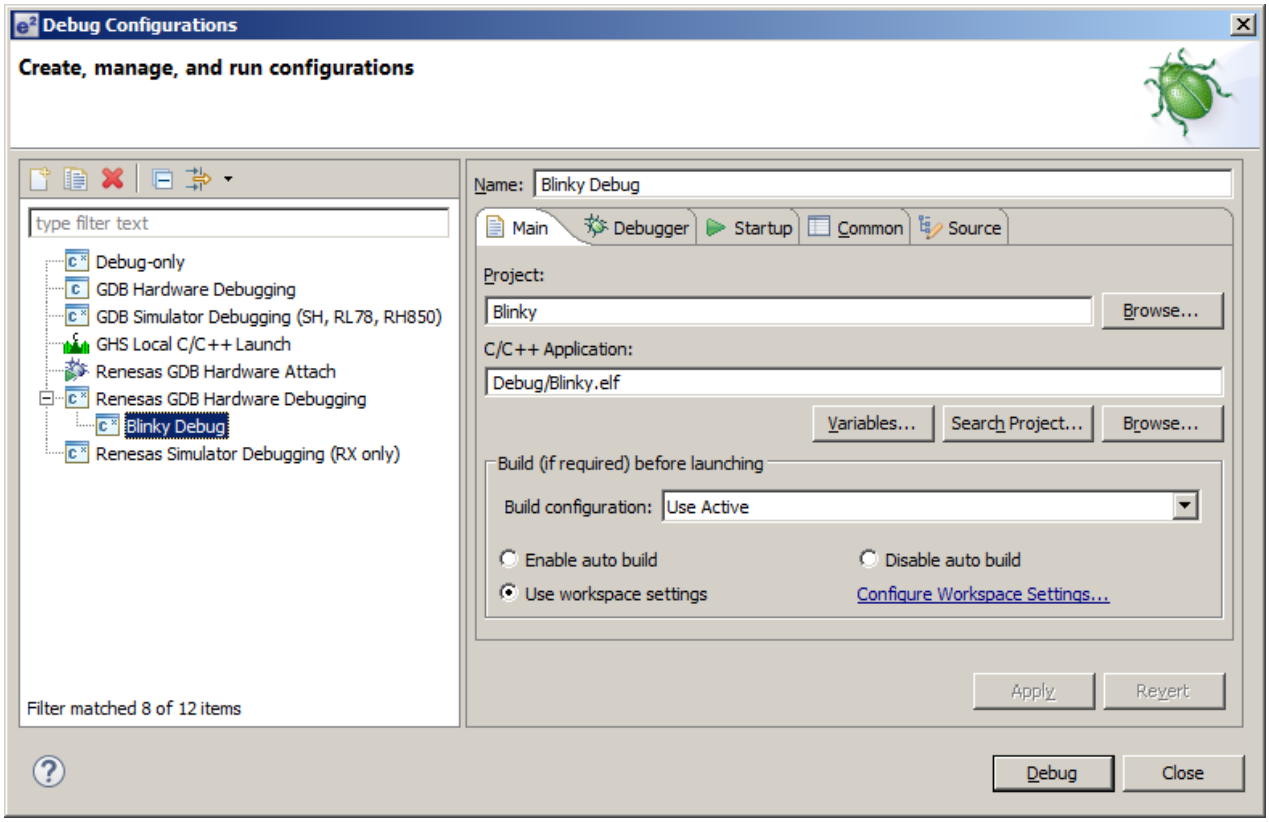


図 80: e² studio ISDE の [Debugger Configuration] ウィンドウと Blinky プロジェクト

3) [Debug] を押してアプリケーションのデバッグを開始します。

3.2.6.3 デバッグ手順の詳細

デバッグモードでは、ISDE は次のタスクを実行します。

1) アプリケーションイメージをマイクロコントローラにダウンロードし、イメージを内蔵フラッシュメモリにプログラミングします。
2) main() にブレークポイントを設定します。
3) スタックにスタックポインタレジスタを設定します。
4) プログラムカウンタレジスタに、リセットベクトルのアドレスをロードします。
5) プログラムカウンタが指しているスタートアップコードを表示します。

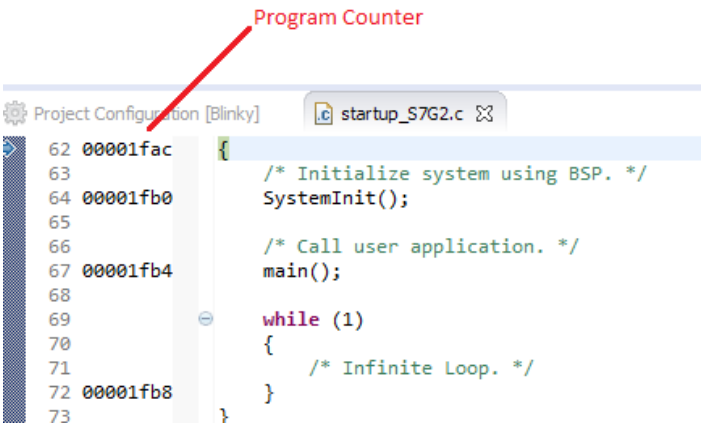


図 81: e² studio ISDE のデバッガメモリウィンドウ

3.2.7 Blinky プロジェクトの実行

デバッグモードで、[Run] > **[Resume]** をクリックするか、[Play] アイコンを 2 回クリックします。

図 82: e² studio ISDE デバッガの [Play] アイコン

ボード上の LED1 とマークされた LED が点滅します。

3.3 チュートリアル : Using HAL Drivers - Programming the WDT

3.3.1 WDT アプリケーション

このアプリケーションは、WDT HAL ドライバーの WDT によって実装されている WDT Interface を使用します。このドキュメントでは、ISDE と SSP を使用して、Synergy MCU のウォッチドッグタイマ (WDT) 周辺デバイス用のアプリケーションを作成する方法について説明します。このアプリケーションは、以下の SSP モジュールを利用します。

- Board Support Package（ボードサポートパッケージ）
- CGC（クロック生成回路）
- WDT（ウォッチドッグタイマ）
- IOPORT(GPIO)

3.3.2 Synergy SSP と ISDE を使用した WDT アプリケーションの作成

3.3.2.1 SSP および e^2 studio ISDE の使用方法

Renesas 製の Synergy Software Package (SSP) は、Synergy アプリケーションを開発するためのドライバーライブラリ一式を提供します。SSP は、ハードウェア抽象化レイヤー (HAL) ドライバー、ボードサポートパッケージ (BSP) ドライバー、および開発者がアプリケーションを作成するために使用する上位のフレームワークアプリケーションを提供します。SSP は、eclipse ベースの Renesas e^2 studio 統合ソリューション開発環境 (ISDE) に統合されており、ビルド（エディター、コンパイラ、リンカー）およびデバッグフェーズと、拡張された GNU Debug (GDB) インタフェースを提供します。

3.3.2.2 WDT アプリケーション

WDT アプリケーションのフローチャートを以下に示します。

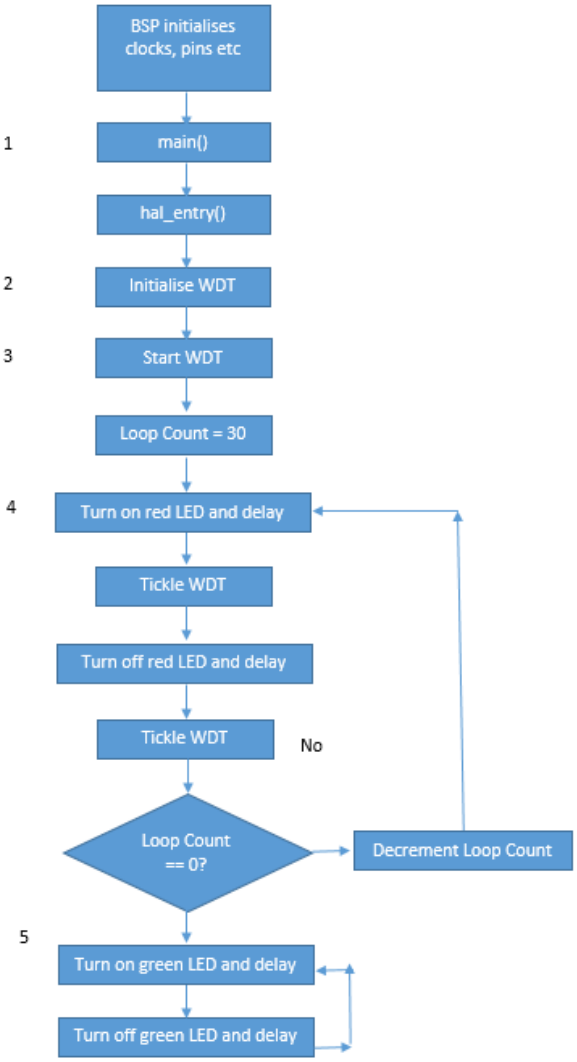


図 83: WDT アプリケーションのフロー図

3.3.2.3 WDT アプリケーションのフロー

WDT アプリケーションの主な部分は以下のとおりです。

1) main() が hal_entry() を呼び出します。関数 hal_entry() は SSP によって作成され、ユーザーコードのプレースホルダが含まれます。WDT 用のコードをこの関数に追加します。
2) WDT を初期化します。ただし、まだ開始しません。
3) WDT をリフレッシュすることによって、WDT を開始します。
4) 赤い LED が 30 回点滅し、LED の状態が変化するたびにウォッチドッグがリフレッシュされます。

5) 緑色の LED が点滅しますが、ウォッチドッグはリフレッシュされません。ウォッチドッグのタイムアウト期間の後、デバイスがリセットされます。その様子は、シーケンスが繰り返されるときに赤い LED が再度点滅することで確認できます。

3.3.3 ISDE でのプロジェクトの作成

ISDE を起動し、ワークスペースラウンチャーでワークスペースフォルダーを選択します。以下のようにして新しい Synergy プロジェクトを設定します。

1) [File]>[New]>[Synergy C Project] の順に選択します。

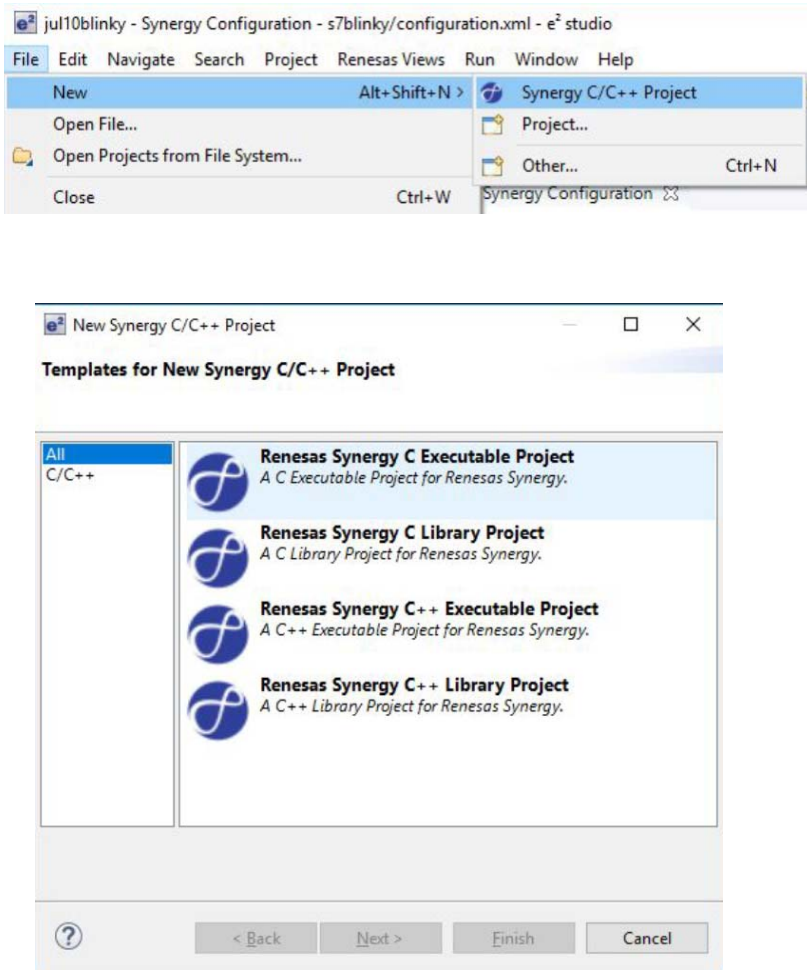


図 84: 新しいプロジェクトの作成

2) ISDE の [SynergyProject Generator (Synergy C Project)] ウィンドウにプロジェクト名（たとえば、WDT_Application). を入力します。また、ツールチェーンとライセンスファイルを選択します。プロジェ

クトの新しい場所を選択するには、[Use default location] チェックボックスをオフにします。[Next] をクリックします。

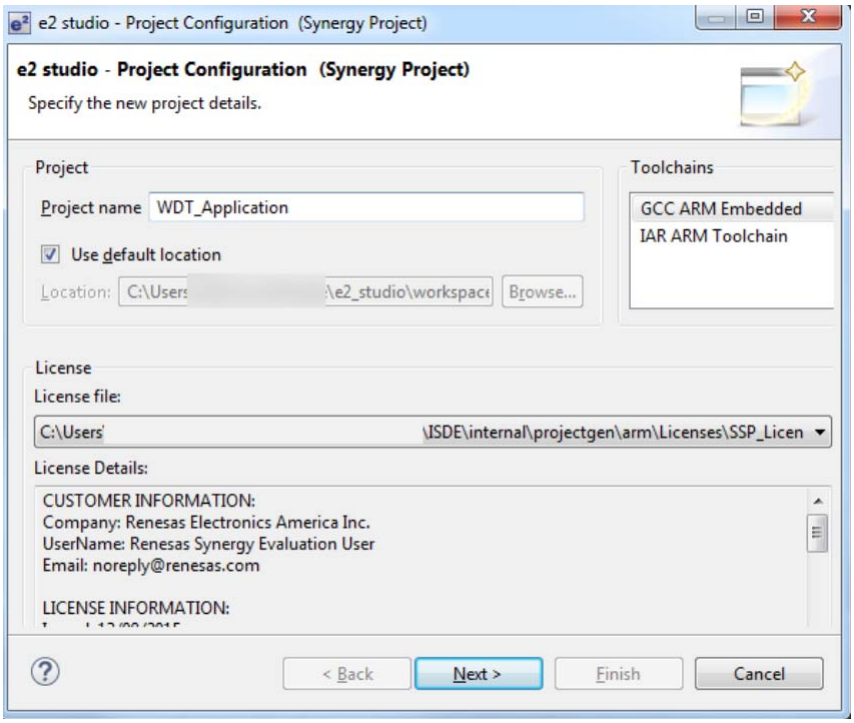


図 85: プロジェクトの構成（パート 1）

3) このアプリケーションは、Synergy S7G2 ベースの DK-S7G2 ボードで動作します。そのため、[Board] には [S7G2 DK] を選択します。これにより、[Device] ドロップダウンに、このボードで使用される正しいデイバイスが自動的に設定されます。ツールチェーンのバージョンを選択します。J-Link Arm をデバッガとして選択します。このアプリケーションでは RTOS は使用されていませんが、デフォルトの Express Logic ThreadX のままで構いません。[Next] をクリックしてプロジェクトを構成します。

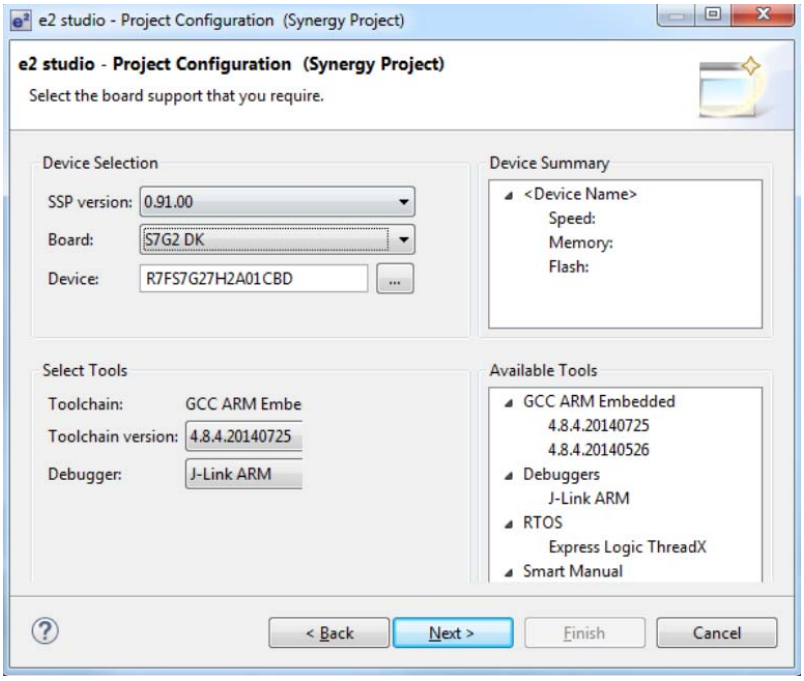


図 86: プロジェクトの構成（パート 2）

今度はプロジェクトテンプレートを選択します。RTOS は必要ないため、**S7G2-DK BSP** を選択します。

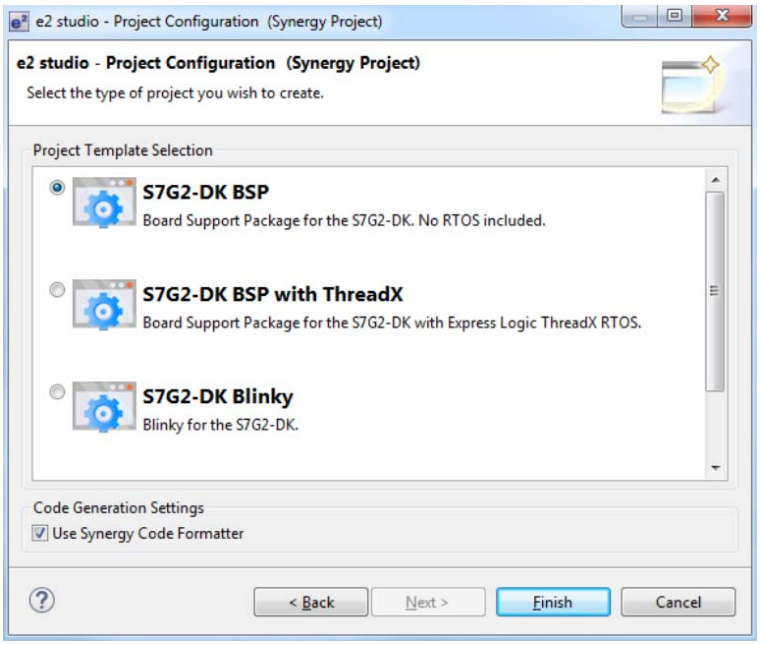


図 87: プロジェクトの構成（パート 3）

4) [Finish] をクリックします。

プロジェクトが作成され、[Project Explorer] ビューと [Synergy Project Editor] ビューが開き、後者の [Summary] タブにプロジェクト構成の概要が示されます。

3.3.4 ISDE でのプロジェクトの設定

e^2 studio ISDE は、プロジェクトを設定するためのオプションを選択する GUI インタフェースを備えているため、プロジェクト構成手順が簡略化され、時間も短縮されます。

ISDE には、進行中の操作に応じて各種のウィンドウを表示する、いくつかのパースペクティブが備わっています。デフォルトのパースペクティブは、[C/C++]、[Synergy Configuration]、[Debug] です。パースペクティブを変更するには、ISDE の右上にあるボタンから新しいパースペクティブを選択します。

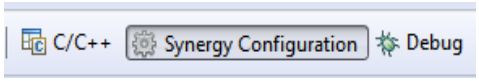


図 88: パースペクティブの選択

C/C++ パースペクティブは、コード編集用に選択されたレイアウトを提供します。[Synergy Configuration] パースペクティブには Synergy プロジェクトを設定するための要素があり、[Debug] パースペクティブはデバッグに適したビューを備えています。

1) プロジェクトを設定するには、[Synergy Configuration] パースペクティブが選択されていることを確認します。
2) [[WDT_Application]Synergy Configuration] が開いていることを確認します。要約情報が表示されている場合はすでに開かれています。ここで、または任意のタイミングで [Synergy Configuration] を開くには、[Synergy Configuration] パースペクティブが選択されていることを確認し、ISDE の右側にある [Project Explorer] ペインで configuration.xml ファイルをダブルクリックします。

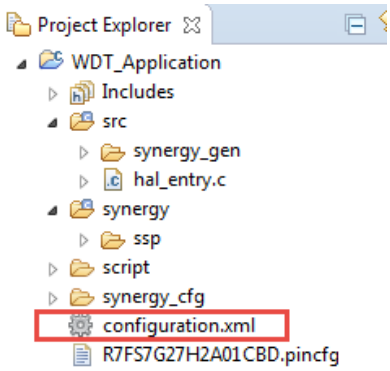


図 89: Synergy プロジェクトコンフィギュレーション設定

[Synergy Project Editor] ビューの下部には、プロジェクトを構成するためのタブがいくつかあります。プロジェクトでは、これらのタブの一部または全部に対する変更が必要になる可能性があります。これらのタブについて以下に説明します。

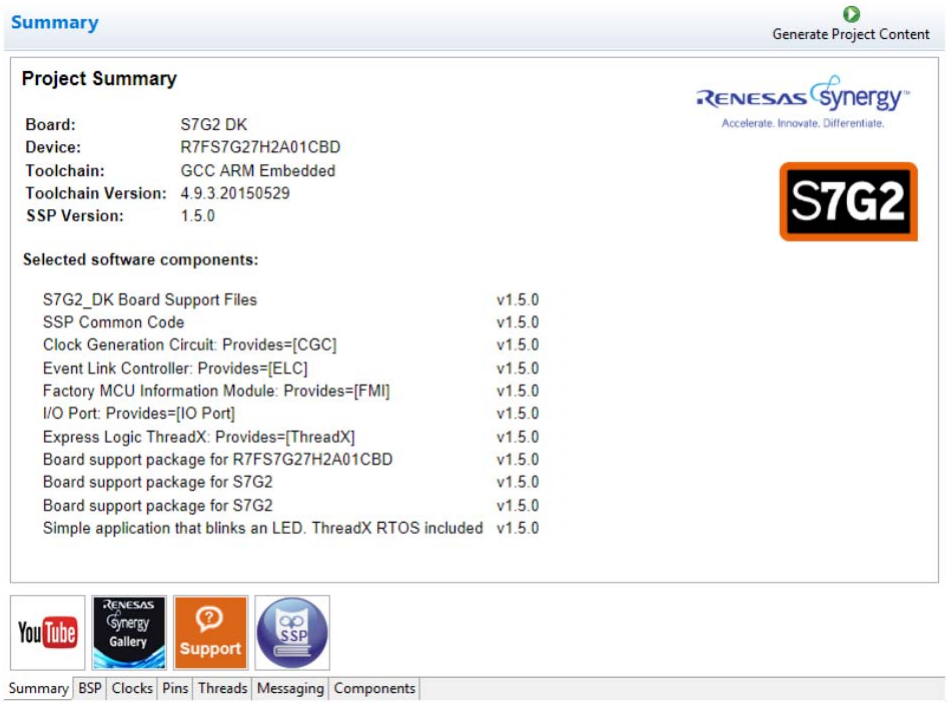


図 90: [Synergy Project Editor] タブ

3.3.4.1 BSP タブ

[BSP] タブでは、ボードサポートパッケージ (BSP) オプションをデフォルト値から変更できます。この特定の WDT プロジェクトについては、変更は不要です。ただし、WDT をオートスタートモードで使用する場合は、[BSP] タブで OFS0（オプション機能選択レジスタ 0) レジスタを設定できます。WDT のオートスタートモードの詳細については、Synergy ハードウェアユーザーズマニュアルを参照してください。

3.3.4.2 [Clocks] タブ

[Clocks] タブには、デバイスのクロックツリーがグラフィカルに表示されます。GUI のドロップダウンボックスを使用して、各種クロックを設定できます。WDT は PCLCKB を使用します。このクロックのデフォルト出力周波数は 60 MHz です。クロックがこの値を出力していることを確認してください。

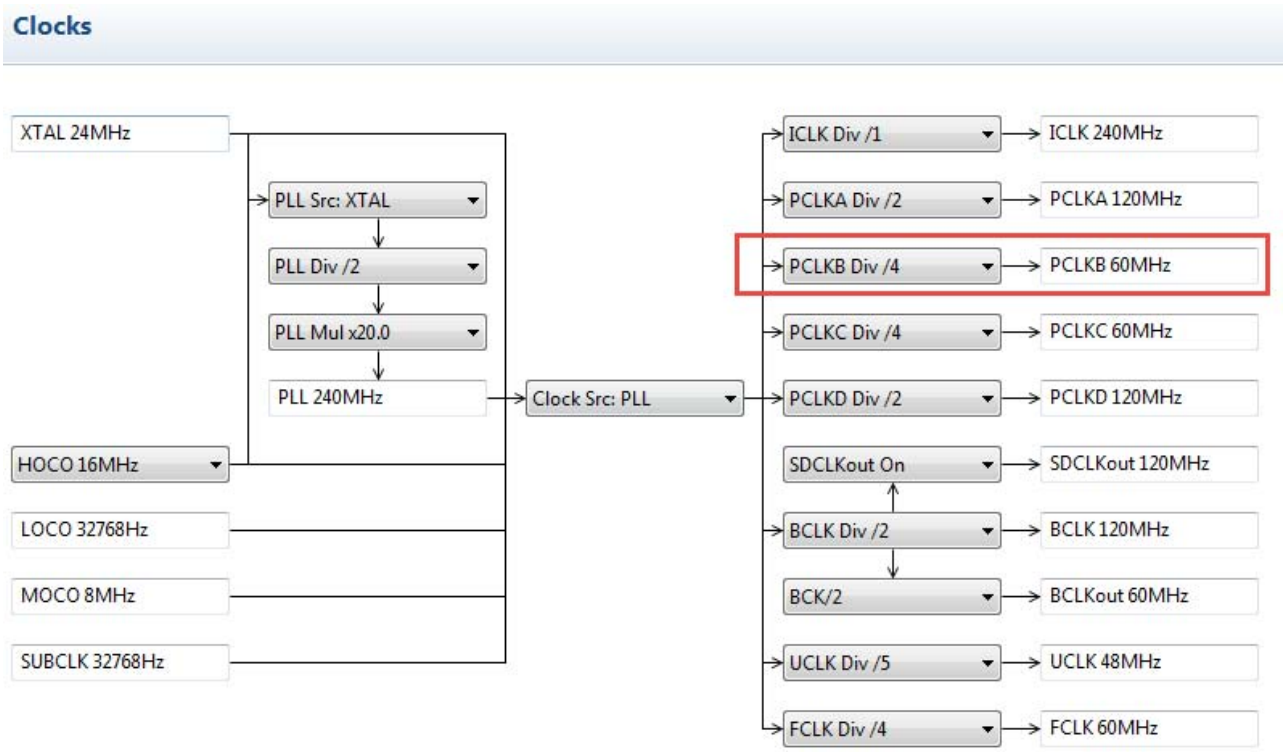


図 91: クロック設定

3.3.4.3 [Pins] タブ

[Pins] タブでは、デバイスのピンの機能を設定するためのグラフィカルツールを利用できます。WDT プロジェクトでは、ピンの設定は不要です。プロジェクトではデバイス上のピンに接続された 2 個の LED を使用しますが、これらのピンは出力 GPIO ピンとして BSP によりあらかじめ設定されています。

3.3.4.4 [Threads] タブ

[Threads] タブでは、任意のドライバーをプロジェクトに追加できます。クロック生成回路、イベントリンクコントローラ、および IO ポートピン用の HAL ドライバーは、プロジェクトの設定時に ISDE によって自動的に追加されます。WDT アプリケーションは ThreadX リソースを使用しないため、追加するのは HAL WDT ドライバーだけにしてください。

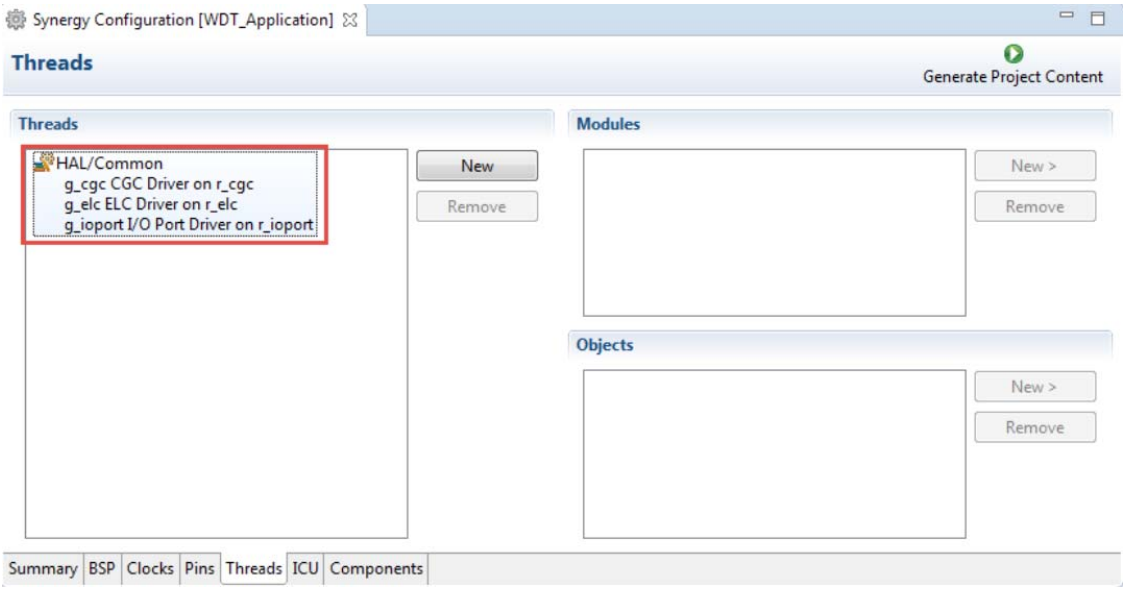


図 92: [Threads] タブ

1) 上の図に示すように、[Threads] ウィンドウで [HAL/Common] パネルをクリックします。[Modules Stacks] ペインが [HAL/Common Stacks] ペインになり、統合ソリューション開発環境によってあらかじめ選択されたモジュールが表示されます。
2) [New] をクリックして利用可能な HAL ドライバーを表示するウィンドウを開きます。
3) [WATCHDOG Driver on WDT] を選択します。

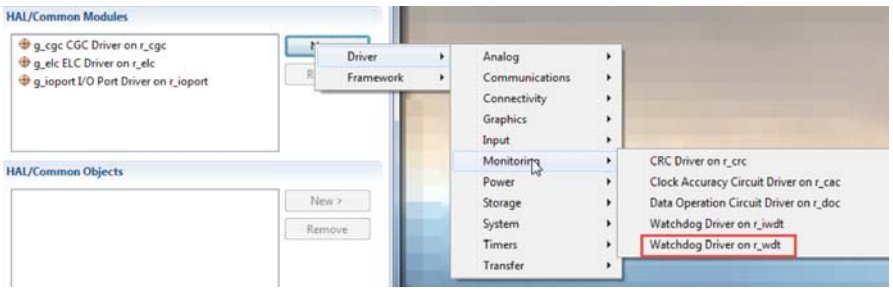


図 93: モジュール選択

選択した HAL WDT ドライバーが **[HAL/Common Modules]** ペインに追加され、選択されたモジュールのすべての構成オプションが **[Property]** ビューに表示されます。**[Property]** ビューが画面の左下に表示されます。表示されない場合は、**[Synergy Configuration]** パースペクティブが選択されていることを確認します。

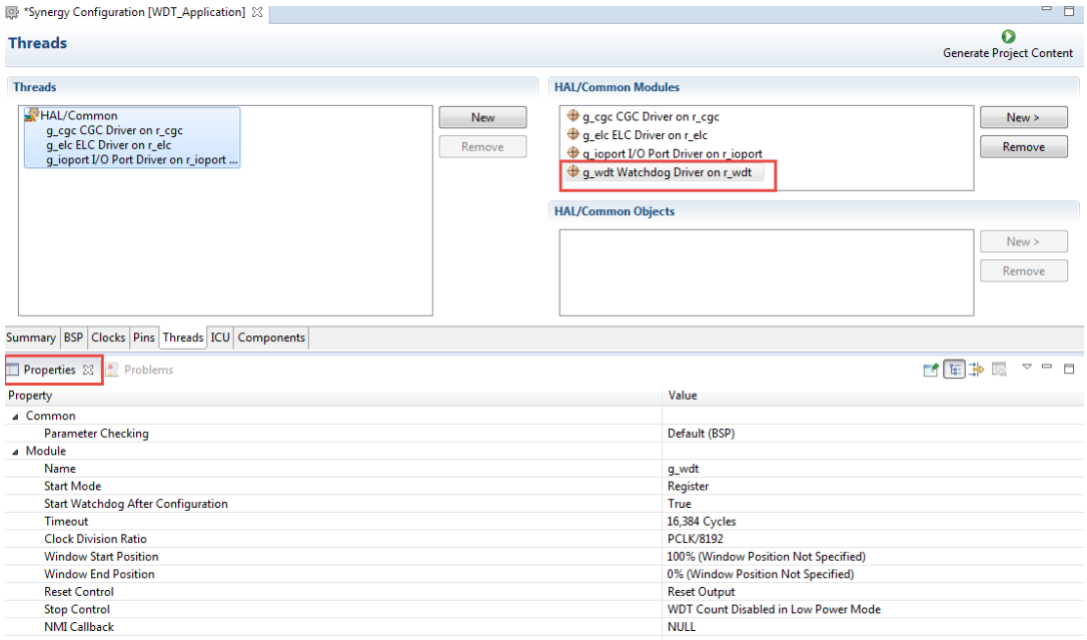


図 94: モジュールプロパティ

パラメータ **[Start Watchdog After Configuration]** を **[True]** から **[False]** に変更します。他のパラメータはデフォルト値のままで構いません。**[Start Watchdog After Configuration]** を **[False]** に設定すると、WDT ドライバーに対し（その open API コールを通じて）、WDT を設定するものの、開始しないように指示します。後で更新することで開始されます。

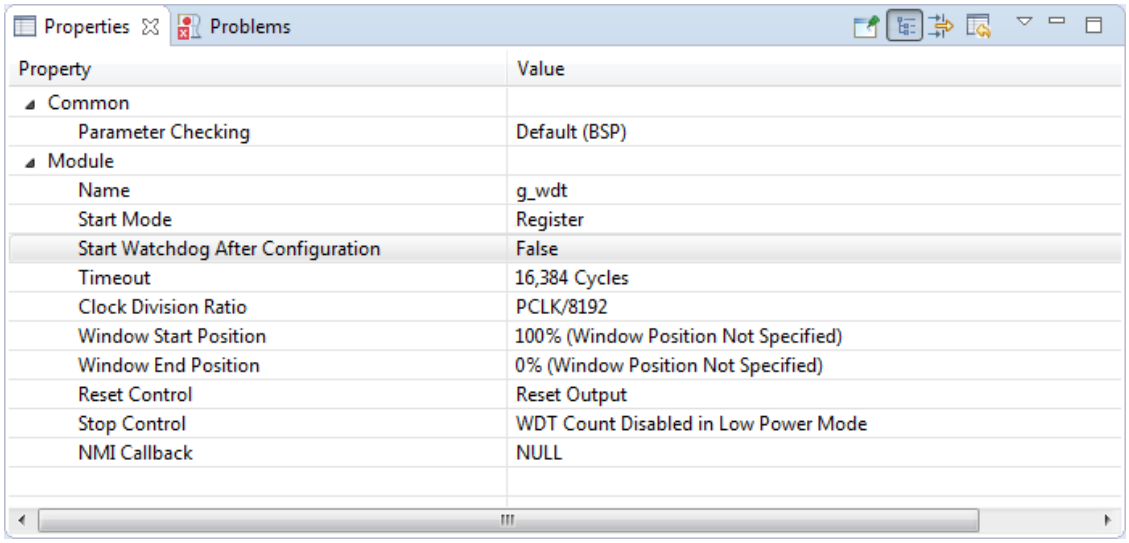


図 95: WDT プロパティ上の g_wdt ウォッチドッグドライバー

PCLKB が 60 MHz で動作している状態で、WDT は最後の更新から 2.23 秒後にデバイスをリセットします。

WDT クロック = 60 MHz / 8192 = 7.32 kHz

サイクル時間 = 1 / 7.324 kHz = 136.53 マイクロ秒

タイムアウト = 136.53 マイクロ秒 x 16384 = 2.23 秒

プロジェクト構成ファイルを保存して、[Synergy Project Editor] の右上隅にある **[Generate Project Content]** ボタンをクリックします。

図 96: [Generate Project Content] ボタン

ISDE はプロジェクトファイルを生成します。

3.3.4.5 [Components] タブ

[Components] タブは参照用に存在し、ここでプロジェクトに含まれているモジュールを確認できます。[Threads] タブでモジュールを追加すると、それらのモジュールが [Components] ビューで自動的に選択されます。

WDT プロジェクトでは、以下のモジュールが選択されていることを確認してください。

1) HAL_Drivers -> r_cgc
2) HAL_Drivers -> r_elc
3) HAL_Drivers -> r_ioport
4) HAL Drivers -> r_fmi
5) HAL_Drivers -> r_wdt

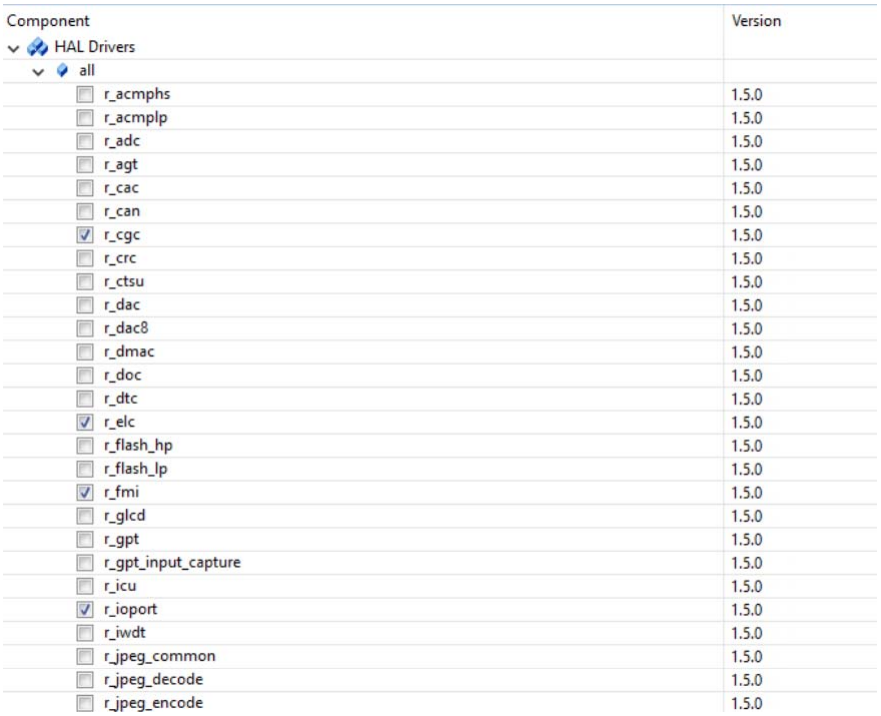


図 97: コンポーネントの選択

注 :[Components] タブに表示されるモジュールの一覧は、インストールされている SSP のバージョンによって異なります。

3.3.5 WDT の生成されるプロジェクトファイル

[Generate Project Content] ボタンを押すと以下のタスクが実行されます。

- r_wdt フォルダーと WDT ドライバーの内容が次の場所に作成されます。
  - synergy/ssp/src/driver/
- r_wdt_api.h が次の場所に作成されます。
  - synergy/ssp/inc/driver/api
- r_wdt.h が次の場所に作成されます。
  - synergy/ssp/inc/driver/instances

上記のファイルは、WDT HAL モジュールの標準的なファイルです。これらのファイルには、プロジェクト固有の内容は含まれていません。これは WDT のドライバーファイルです。これらのファイルの内容の詳細については、WDT HAL モジュールのドキュメントを参照してください。

WDT プロジェクトでの WDT HAL モジュールの設定情報は次の場所にあります。

- synergy_cfg/ssp_cfg/driver/r_wdt_cfg.h

上記のファイルの内容は、**[g_wdt0 Watchdog Driver on r_wdt Properties]** ビューの **[Common]** 設定に基づいています。

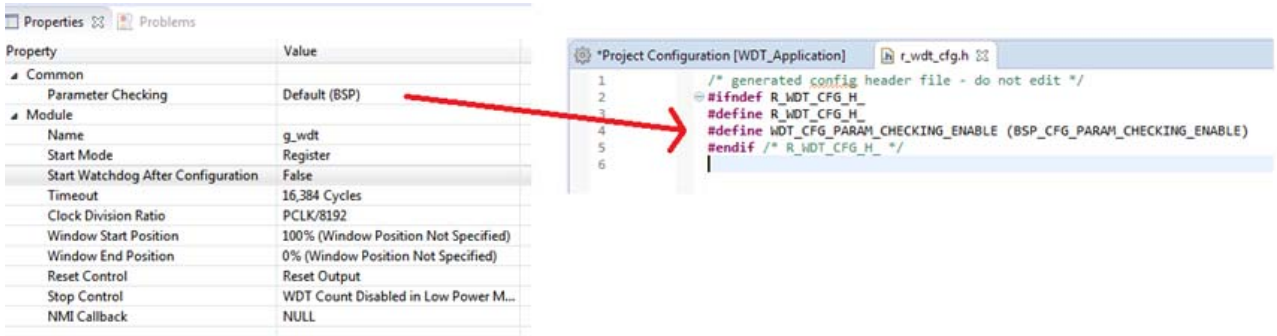


図 98: r_wdt_cfg.h の内容

注 : これらのファイルは、[Generate Project Content] ボタンを押すたびに再作成され、変更内容が上書きされるため、編集しないでください。

ISDE は、WDT 用の HAL ドライバーファイルを生成するのに加えて、WDT 用の構成データが格納されたファイルと、ユーザーコードを安全に追加できるファイルも生成します。これらのファイルを以下に示します。

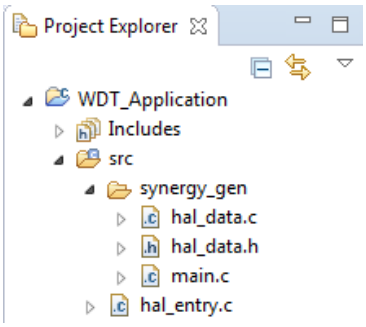


図 99: WDT プロジェクトファイル

3.3.5.1 WDT hal_data.h

hal_data.h をダブルクリックし、ファイルを開いて確認します。

hal_data.h には、ISDE で生成されたプロジェクトに必要なヘッダーファイルが格納されています。また、WDT HAL ドライバーに使用される構成、制御、API の各構造体へのポインタを含む **g_wdt0** インスタンス構造体への外部参照も含まれます。

注 : このファイルは、[Generate Project Content] ボタンを押すたびに再生成されるため、編集しないでください。

3.3.5.2 WDT hal_data.c

hal_data.c をダブルクリックし、ファイルを開いて確認します。

hal_data.c には、WDT HAL ドライバーのこのインスタンス用の制御構造体である g_wdt0_ctrl が含まれています。この構造体は、ドライバーがオープンされたときに初期化されるため、初期化しないでください。

g_wdt0_cfg の内容は、[Properties] ビューの **[g_wdt0 Watchdog Driver on r_wdt Properties]** を使用して、このファイルに設定されます。この構造体の内容に、ISDE で行った設定が反映されていない場合は、**[Project Configuration]** の設定が ISDE で保存されているのを確認してから、**[Generate Project Content]** ボタンを押してください。

注 : このファイルは、[Generate Project Content] ボタンを押すたびに再生成されるため、編集しないでください。

3.3.5.3 WDT main.c

BSP のスタートアップ コードで呼び出される main() が格納されています。main() は、ユーザーが開発したコードが含まれている hal_entry() を呼び出します (次のファイルを参照)。main.c の内容は以下のとおりです。

```
/* generated main source file - do not edit */

extern void hal_entry(void);

void main(void)
{
   hal_entry();

   return 0;
}
```

注 : このファイルは、[Generate Project Content] ボタンを押すたびに再生成されるため、編集しないでください。

3.3.5.4 WDT hal_entry.c

このファイルには、main() から呼び出される関数 hal_entry() が格納されています。ユーザーが開発したコードは、このファイルと関数に格納します。

WDT プロジェクトでは、このファイルの内容を編集して、以下のコードを追加します。このコードは、このドキュメントの概要セクションにあるフローチャートを実装します。

```
/* HAL-only entry function */

#include "hal_data.h"

#define RED_LED_NO_OF_FLASHES 30

#define RED_LED_PIN IOPORT_PORT_08_PIN_08

#define GREEN_LED_PIN IOPORT_PORT_08_PIN_07

#define RED_LED_DELAY_COUNT 1500000

#define GRN_LED_DELAY_COUNT 1200000
```

```
volatile uint32_t delay_counter;
volatile uint16_t loop_counter;
void hal_entry(void) {
    /* TODO: add your own code here */
    /* Open the WDT */
    g_wdt0.p_api->open(g_wdt0.p_ctrl, (wdt_cfg_t *const)g_wdt0.p_cfg);
    /* Start the WDT by refreshing it */
    g_wdt0.p_api->refresh(g_wdt0.p_ctrl);
    /* Flash the red LED and tickle the WDT for a few seconds */
    for(loop_counter=0; loop_counter<RED_LED_NO_OF_FLASHES; loop_counter++)
    {
        /* Turn red LED on */
        g_ioport.p_api->pinWrite(RED_LED_PIN, IOPORT_LEVEL_HIGH);
        /* Delay */
        for(delay_counter=0; delay_counter<RED_LED_DELAY_COUNT; delay_counter++)
;
        /* Refresh WDT */
        g_wdt0.p_api->refresh(g_wdt0.p_ctrl);
        /* Turn red off */
        g_ioport.p_api->pinWrite(RED_LED_PIN, IOPORT_LEVEL_LOW);
        /* Delay */
        for(delay_counter=0; delay_counter<RED_LED_DELAY_COUNT; delay_counter++)
;
        /* Refresh WDT */
        g_wdt0.p_api->refresh(g_wdt0.p_ctrl);
    }
```

```
    /* Flash green LED but STOP tickling the WDT. WDT should reset the
    device */

    while(1)

    {

    /* Turn green LED on */

    g_ioport.p_api->pinWrite(GREEN_LED_PIN, IOPORT_LEVEL_HIGH);

    /* Delay */

    for(delay_counter=0; delay_counter<GRN_LED_DELAY_COUNT; delay_counter++);

    /* Turn green off */

    g_ioport.p_api->pinWrite(GREEN_LED_PIN, IOPORT_LEVEL_LOW);

    /* Delay */

    for(delay_counter=0; delay_counter<GRN_LED_DELAY_COUNT; delay_counter++);

    }

}
```

WDT HAL ドライバーは、**r_wdt.h** で定義されているインタフェース **g_wdt_on_wdt** を通じて呼び出されます。WDT HAL ドライバーは、**r_wdt_api.h** で定義されているインスタンスを使用して、open API 呼び出しを通じてオープンされます。

```
g_wdt0.p_api->open(g_wdt0.p_ctrl, (wdt_cfg_t *const)g_wdt0.p_cfg);
```

最初に渡されるパラメータは、hal_data.c でインスタンス化される制御構造体 **g_wdt0_ctrl** へのポインタです。2 番目のパラメータは、同じ hal_data.c ファイルでインスタンス化される構成データ **g_wdt0_cfg** へのポインタです。

WDT は、次の API 呼び出しによって開始および更新されます。

```
g_wdt0.p_api->refresh(g_wdt0.p_ctrl);
```

ここでも、この API に渡される最初の （この場合は唯一の）パラメータは、ドライバーのこのインスタンスの制御構造体へのポインタです。

3.3.6 プロジェクトのビルドとテスト

プロジェクトは、**[Build] > [Build Project]** でビルドします。プロジェクトはエラーなくビルドされる必要があります。

プロジェクトをデバッグするには

1) ターゲットボードとホスト PC の間に J-Link デバッガを接続します。ボードの電源を投入します。
2) e^2 studio ISDE の左側にある **[Project Explorer]** ビューで、WDT プロジェクト **WDT_Application** を右クリックし、**[Debug As] > [Debug Configurations]** を選択します。
3) [Renesas GDB Hardware Debugging] で、以下に示すように [WDT_Application Debug] を選択します。

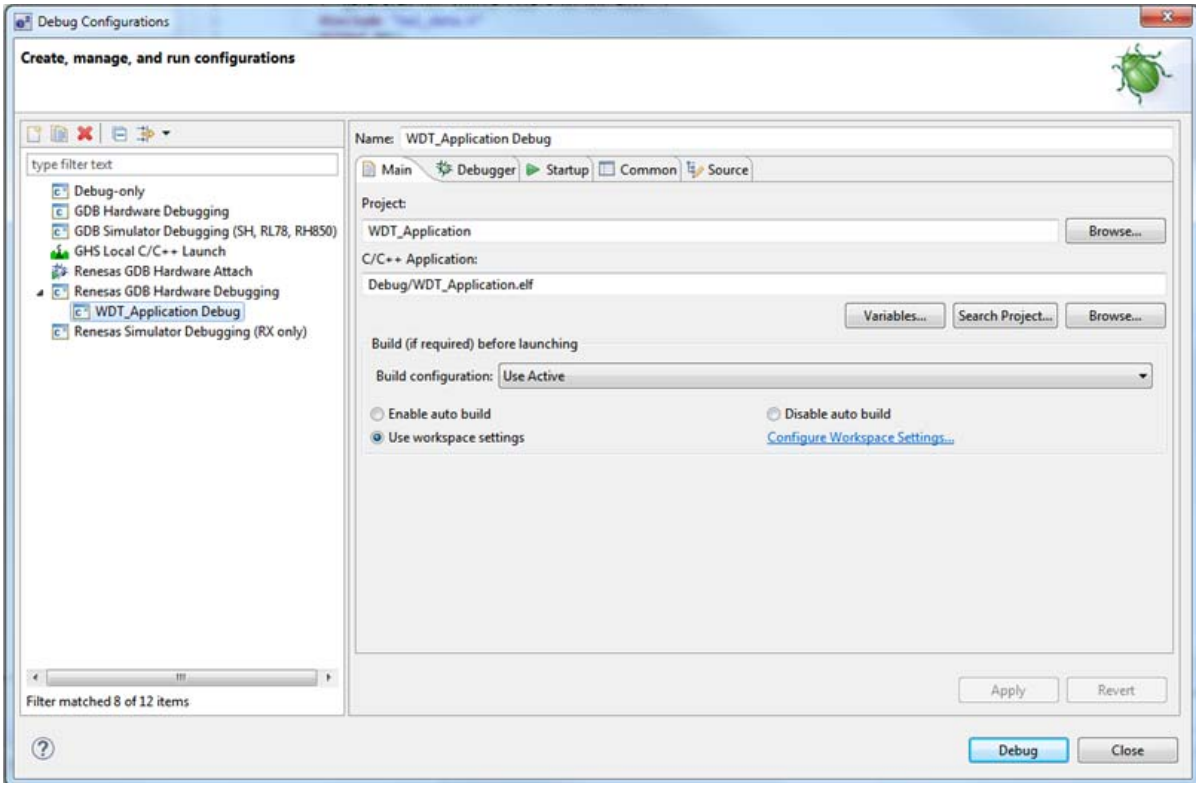


図 100: デバッグ設定

4) **[Debug]** ボタンを押します。質問されたらデバッグパースペクティブに切り替えます（[Yes] をクリック）。
5) コードは Reset_Handler() 関数まで実行されます。
6) **[Run]** > **[Resume]** で実行を再開します。main() の **hal_entry()** の呼び出しで実行が停止します。
7) 再度実行を再開します。

赤い LED が点滅を開始します。30 回点滅した後、緑の LED が点滅を開始し、赤い LED の点滅が止まります。

緑の LED が点滅している間、WDT がアンダーフローしてデバイスをリセットし、その結果シーケンスが繰り返されて再び赤い LED が点滅します。ただし、デバッガに接続されているときには WDT は動作しないため、このシーケンスはデバッガを使用しているときには発生しません。

1) ISDE で、**[Run]** > **[Terminate]** を選択してデバッガを停止します。
2) ターゲットボード上のリセットボタンを押します。LED が点滅を開始します。

3.4 IAR Embedded Workbench for Renesas Synergy

3.4.1 IAR Embedded Workbench for Renesas Synergy の使用

このセクションでは、IAR Embedded Workbench for Renesas Synergy（IAR EW for Synergy）と Renesas Synergy Standalone Configurator（SSC）を組み合わせて使用し、Renesas Synergy Software Package（SSP）を用いたアプリケーションを開発する方法について説明します。SSP のアーキテクチャは、IAR EW for Synergy と SSC をどのように使用して Synergy アプリケーションを開発するかを直接決定します。このマニュアルに含まれる SSP アーキテクチャの詳細については、次のドキュメントを参照してください。

- SSP アーキテクチャ
- BSP アーキテクチャ

3.4.2 IAR EW for Synergy とは

IAR Embedded Workbench は現在、Renesas Synergy プラットフォームに完全に統合されています。新製品の IAR EW for Synergy は、アドオン機能を提供してソフトウェア開発を簡略化し加速するほか、最適なパフォーマンスを実現し、コードサイズを最小限に抑えます。

e^2 studio と同じく、IAR EW for Synergy は安全なソースレベルの可視性を Synergy Software Package（SSP）にもたらし、さらに安全なソースレベルのデバッグを可能にします。開発者は保護されたソースコードを見ることはできますが、変更したり保存したりすることはできません。アプリケーションコードの開発後に、IAR EW for Synergy が IAR C-STAT® アナライザと C-RUN® アナライザを組み入れるので、ユーザーはこれらのツールを利用してアプリケーションコードの品質改善を進めることができます。

3.4.3 主な特長

- プロジェクト管理ツールとエディタを備えた統合開発環境
- Renesas Synergy デバイス用に高度な最適化を実現する C および C++ コンパイラとリンカ
- Renesas Synergy Standalone Configurator（SSC）に対する統合サポート
- コード分析ツール C-STAT および C-RUN を内蔵
- 広範な HW ターゲットシステムをサポート
- ソースコードと電力消費の相関関係を視覚化するパワーデバッグ

- JTAG/SWD をサポートし、ハードウェア上の RTOS 対応デバッグもサポートする C-SPY® デバッガ
- ETM トレースのサポート
- 包括的なユーザーガイドやリファレンスガイドと状況に応じたヘルプ機能
- Arm 組み込みアプリケーションバイナリインタフェース（EABI）と Arm Cortex® マイクロコントローラソフトウェアインタフェース標準（CMSIS）に準拠

IAR EW for Synergy のダウンロードとインストールの詳しい方法については、Synergy Gallery の IAR EW for Synergy リリースノートを参照してください。

3.4.4 Synergy Standalone Configurator（SSC）とは

Synergy Standalone Configurator（SSC）は、Renesas e^2 studio 統合ソリューション開発環境に実装されている Eclipse Rich Client Platform（RCP）アプリケーションで、Synergy Project Generator と Synergy Project Editor が含まれています。SSC は、クロックコンフィギュレータ、ピンコンフィギュレータ、RTOS コンフィギュレータ、SSP モジュールセレクタ / コンフィギュレータ、割り込みコントローラユニット（ICU）コンフィギュレータといった各種のコンフィギュレータを備えており、それらを IAR EW for Synergy などのサードパーティ IDE で使用することができます。

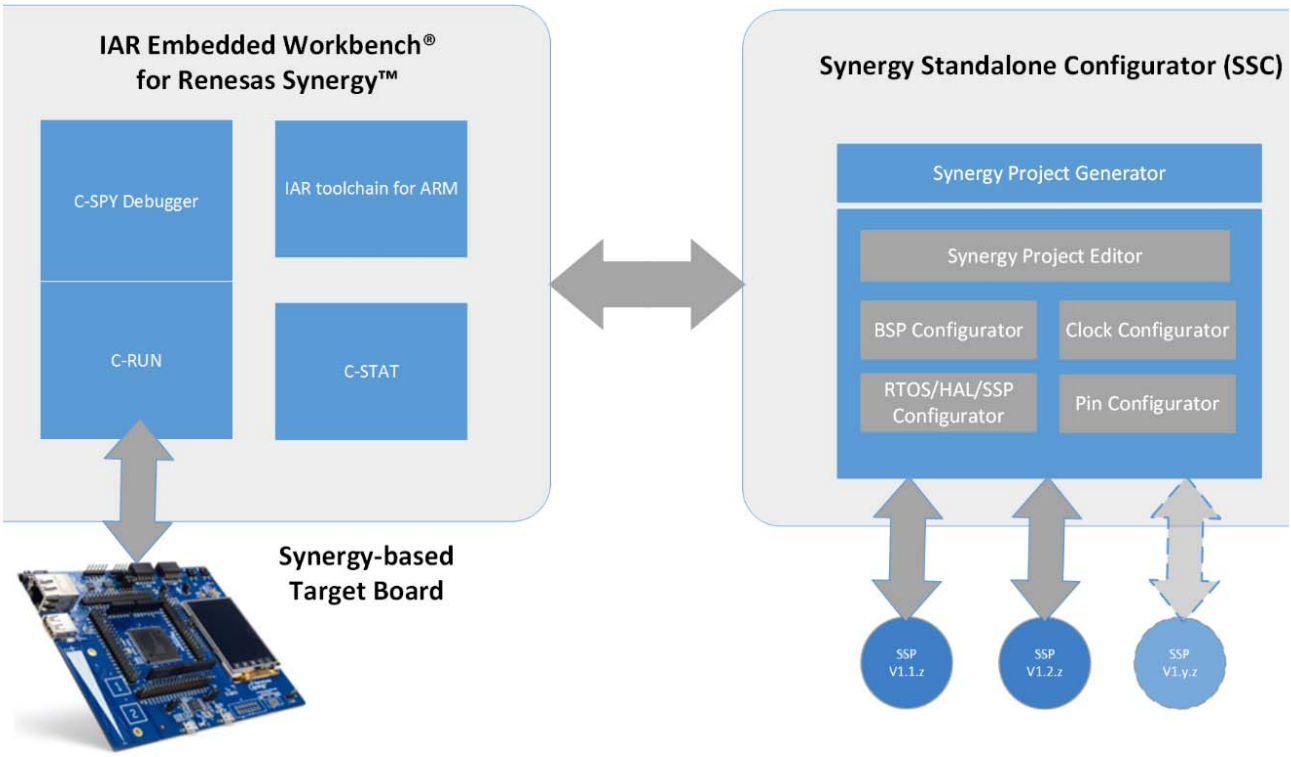


図 101:　IAR EW for Synergy と SSC の機能ブロック図

SSC の機能は Renesas e^2 studio 統合ソリューション開発環境に実装されている Synergy Project Generator および Synergy Project Editor と同様であるため、使用法については e^2 studio ISDE ユーザーガイドを参照してください。

IAR EW for Synergy とともに使用する SSC と SSP をダウンロードおよびインストールする方法については、Synergy Gallery の SSC リリースノートと SSP リリースノートを参照してください。

3.4.5 ツールのインストール

ツールをインストールするには、次の手順を実行します。

1) Renesas Synergy Gallery から Renesas Synergy Standalone Configurator（SSC）をダウンロードして、インストールします。[Development Tools] の下に配置されています。デフォルトのインストールディレクトリは C:\Renesas\Synergy\SSC_<SSCversion> です。
2) Renesas Synergy Gallery から Renesas Synergy Software Package（SSP）をダウンロードして、インストールします。インストール中に、SSP のインストールディレクトリを指定するよう求められます。SSP インストーラに対し、直前に SSC をインストールしたディレクトリを指定します（たとえば、C:\Renesas\Synergy\SSC_<SSCversion>）。
3) Renesas Synergy Gallery から IAR Embedded Workbench for Renesas Synergy をダウンロードして、インストールします。次の手順で IAR Embedded Workbench をインストールします。
   a) Web ブラウザに URL https://synergygallery.renesas.com を入力し、Renesas Synergy Gallery から IAR Embedded Workbench for Renesas Synergy をダウンロードします。ライセンスを入手してライセンス番号を取得する方法についても、同じ場所に情報があります。
   b) ダウンロードしたファイルに含まれているインストーラを実行します。
   c) IAR License Manager の指示に従ってライセンス番号を入力します。

      注 : IAR EW for Synergy ライセンスにより、ユーザーは IAR Embedded Workbench のこのエディションを使用する権利を付与されますが、Synergy Standalone Configurator の使用にはまた別のライセンスが必要です。

3.4.6 IAR EW for Synergy と SSC での Renesas Synergy プロジェクトの作成

IAR EW for Synergy と SSC を使用して Synergy プロジェクトを作成するには、次の手順に従います。

1) IAR Embedded Workbench IDE で、**[Project]>[Create New Project]** を選択します。
2) **[Create New Project]** ダイアログボックスで、**[Renesas Synergy Project]** を選択し、**[OK]** をクリックします。

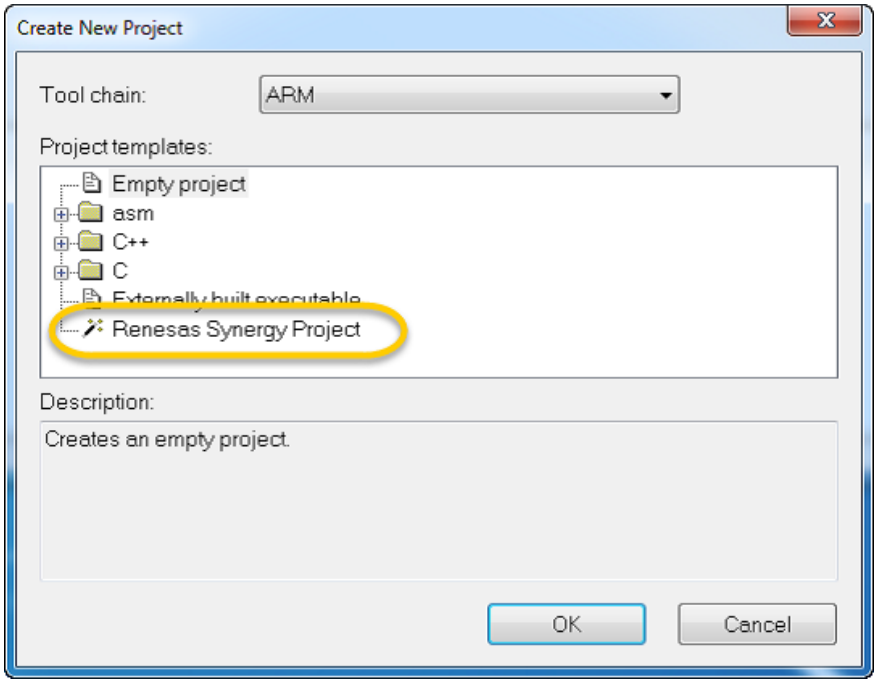


図 102: IAR EW for Synergy での新しい Synergy プロジェクトの作成

3) **[Save As]** ダイアログボックスで、ワークスペース（プロジェクトを格納するコンテナ）の適切な保存先ディレクトリ（「MyWorkspace」など）を選択し、**[Save]** をクリックします。

   注：ワークスペースは、オペレーティングシステムのルートディレクトリ（C:）に保存しないでください。

4) **[Renesas Synergy Setting]** ダイアログボックスで、次のように指定します。

   - インストールされた Synergy Standalone Configurator（SSC）の場所。デフォルトでは、C:\Renesas\Synergy\SSC_<SSCversion> にインストールされます。
   - Synergy ライセンスファイル。これは、SSC のインストールディレクトリにあります（たとえば、C:\Renesas\Synergy\SSC_<SSCversion>\internal\projectgen\arm\Licenses）。
   - 暗号化されたソースファイルをすべて復号されたソースファイルに置き換えるかどうか。この機能には、Renesas Synergy ソースライセンスが必要であることに注意してください。

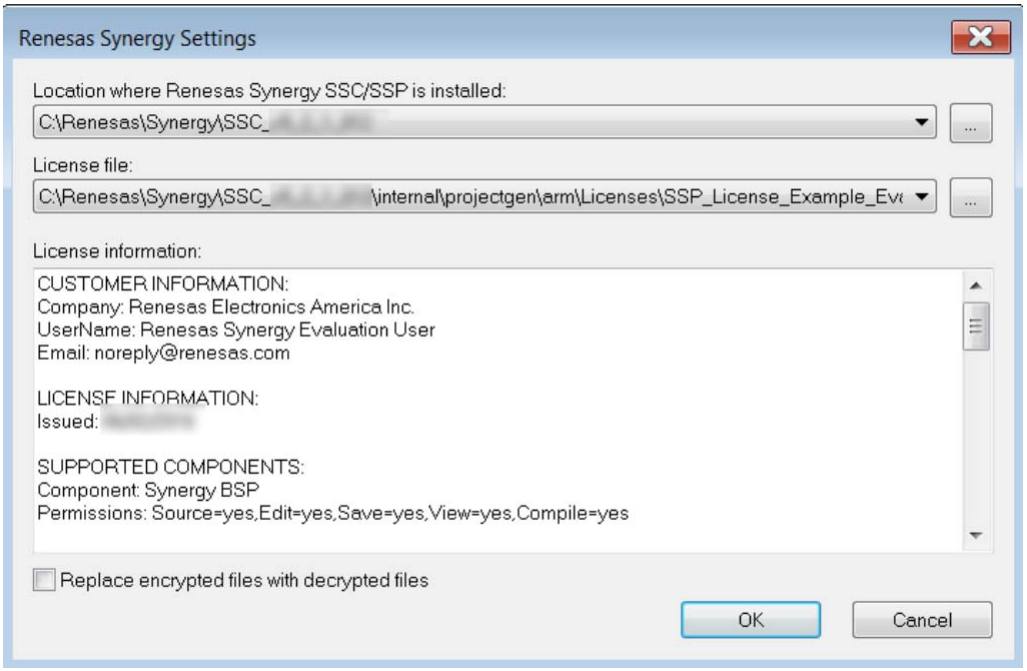


図 103: Renesas Synergy 設定（IAR EW for Synergy）

5) **[OK]** をクリックします。
6) **[Save As]** ダイアログボックスで、プロジェクトの名前（「MyProject」など）を指定します。

   注 : プロジェクトは、オペレーティングシステムのルートディレクトリ（C:）に保存しないでください。
7) ここで IAR Embedded Workbench IDE は Renesas Synergy Standalone Configurator（SSC）に接続します。必要なボードサポートを指定します。

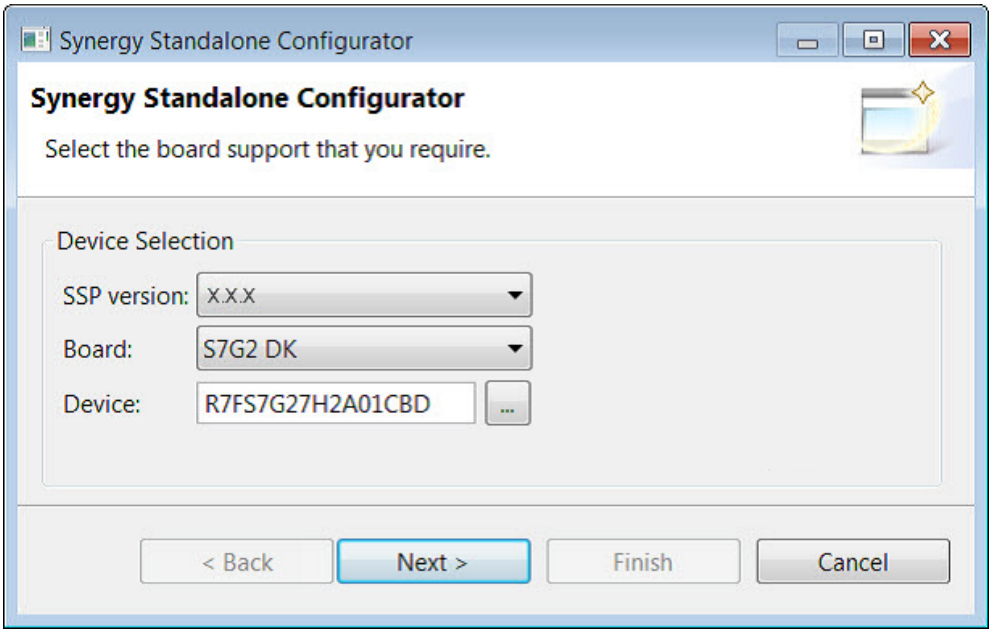


図 104: 使用する SSP、ボード、MCU に関する SSC 選択

注 : ドロップダウンリストには、ご使用のコンピュータにインストール済みの SSP バージョンが表示されます。

[Next] をクリックします。

8) Synergy Software Package にはいくつかのサンプルプロジェクトが用意されており、使用するデバイスに応じたソースコードファイル、ヘッダーファイル、リンカ構成ファイルなどが含まれています。プロジェクトに追加したいサンプルパッケージを選択します。

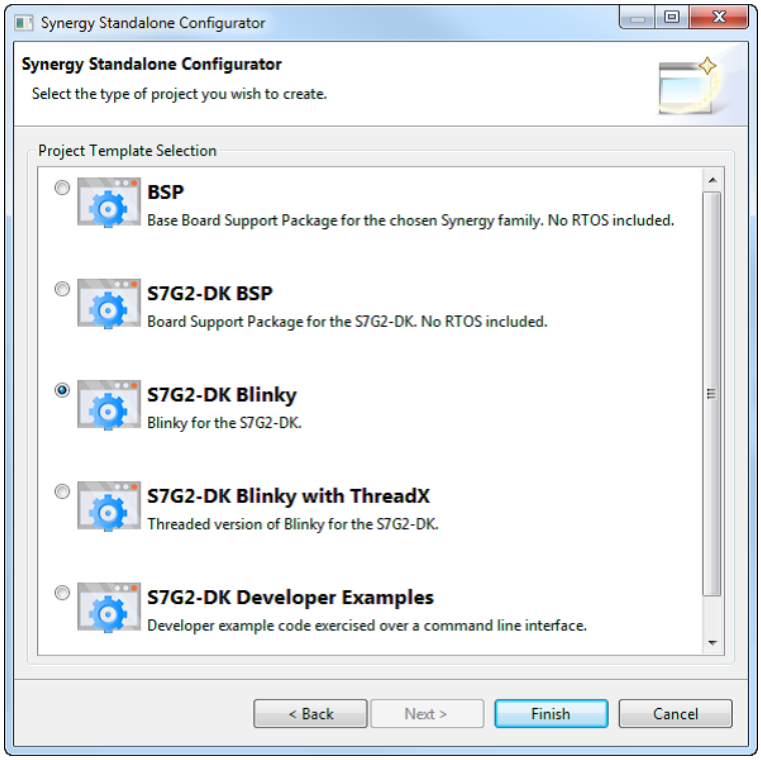


図 105: [Project Template Selection] ダイアログ

[Finish] をクリックします。

9) Synergy Project Editor が表示され、MCU ピン機能の割り当て、クロックとペリフェラルの設定、割り込み要因の割り当てを構成することができます。構成が完了したら、**[Generate Project Content]** ボタンをクリックします。するとソースコードが生成されます。

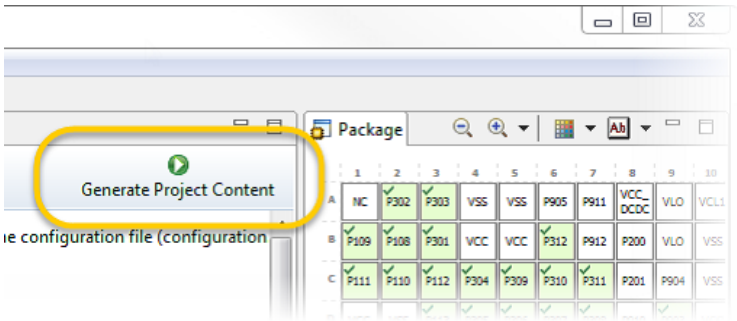


図 106: [Generate Project Content] ボタン

注 : Synergy プロジェクトの構成は、後からいつでも追加したり変更したりすることができます。

10) 数秒後、IAR EW の [Workspace] ウィンドウに Renesas Synergy プロジェクトが表示されます。

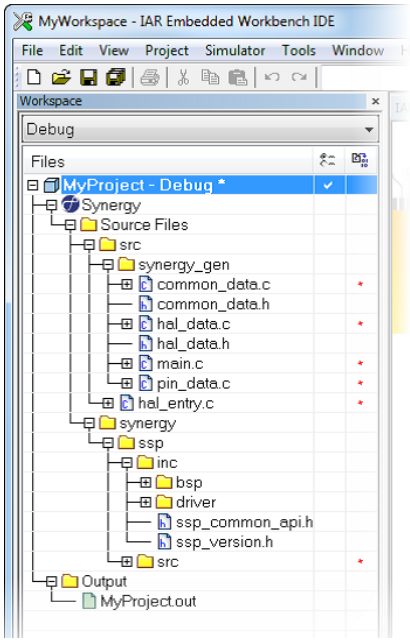


図 107: IAR EW for Synergy のワークスペースとプロジェクトの表示ウィンドウ

11) SSC を閉じた場合は、ツールバーの [Synergy Configuration] ボタンをクリックするか、

メニューから **[Renesas Synergy]>[Configurator]** を選択して開くことができます。

12) 構成設定を変更するために SSC に切り替える場合は常に、完了時に **[Generate Project Content]** ボタンをクリックします。該当するソースコードファイルが再生成されます。

13) この後は IAR Embedded Workbench IDE の標準の操作に従って、ビルドとデバッグを実行できます。IAR ウェブサイトの『IAR Embedded Workbench®IDE User Guide』を参照してください。SSC は e^2 の Synergy Project Editor と同様に動作するので、詳しい使用法については、『e^2 studio ISDE ユーザーガイド』を参照してください。Synergy Project はデフォルトで J-Link Debug Probe 用に構成されていることに注意する必要があります。I-jet または I-jet Trace などの他のデバッグプローブを使用する場合は、IDE で **[Project]>[Options]>[Debugger]** を選択し、[Driver] ドロップダウンリストから「I-jet/JTAGjet」を選択します。

第 4 章モジュールの概要

「モジュールの概要」のリストは以下のページで見つけることができます。

- フレームワークレイヤー
- HAL ドライバー
- Express Logic 社のモジュール

各 SSP モジュールの「モジュールの概要」は、前のリリースから大幅に向上されています。新しい「モジュールの概要」では、ターゲットアプリケーションでの使用について特定のモジュールの適合性を評価するために開発者が必要なすべての情報が提供され、開発プロセスに非常に役立ちます。このような説明の目的は、ターゲットモジュールを使用して開発を始めるために必要なすべての情報を、わかりやすい場所にまとめて記載することです。

それぞれの「モジュールの概要」には次のセクションが含まれます。

1) モジュールの簡単な紹介には、短い説明、主要モジュールコンポーネントのブロック図、機能リストが含まれます。
2) API 表には、使用可能なすべての API、API の使用例、API 機能の簡単な説明が含まれます。主要なステータス戻り値のリストもあり、API 呼び出しの結果を判別する際に役立ちます。
3) 機能の概要では、主要なモジュールの動作について説明します。モジュールの重要な制限事項のリストも含まれます。
4) ISDE の [Threads] タブやスタック選択プロセスを使用してモジュールをアプリケーションに組み込む詳しい手順。
5) モジュールおよび主要なローレベルモジュールの構成パラメータを説明する一連の表が提供され、開発者がモジュールの重要機能をすぐに把握できます。構成可能なこれらのプロパティは、MCU シリーズや SSP リリースごとに異なることに注意してください。これらの表の説明を例として参照しながら、ターゲット MCU および選択した SSP リリースの ISDE で設定できる実際のパラメータを確認してください。ピンとクロックの構成例と選択に関する情報も提供され、開発を進める際に役立ちます。
6) ターゲットモジュールを使用する単純な実装について説明し、典型的なアプリケーションで使用される手順、関連するフロー図、各手順での API の使用を示します。これによって、API の一般的な使用方法がわかり、開発者が効率よく実装を始めることができます。

モジュールガイドのアプリケーションの注意事項

これら 6 個のセクションはそれぞれのモジュールの「モジュールガイドのアプリケーションの注意事項」にも含まれます。アプリケーションの注意事項には、関連するアプリケーションプロジェクトの詳しい説明（実際の設計におけるモジュールの仕組み）が含まれるその他のセクションがあります。新しい設計の出発点または参考としてアプリケーションプロジェクトを使用すると、開発を大幅に簡略化することができます。モジュールガイドのアプリケーションの注意事項は、[Renesas Synergy Tools & Kits] ページの [Sample Code] タブの下にあります（https://www.renesas.com/en-us/products/synergy/tools-kits.html#sampleCodes）。モジュールガイドのアプリケーションの注意事項は随時追加されているため、新しいノートがリリースされているかどうかときどき注意してください。

モジュールガイドのモジュールの概要の使用

「モジュールの概要」によって開発を始めるために十分な詳細情報が提供されますが、設計を実装する際にはさらに多くの情報が役立つケースがあります。『SSP ユーザーズマニュアル』には、API の実装、構造、列挙などの詳細が豊富に提供されています。API リファレンスのセクションに移動し、関心があるモジュールを見つけて必要な追加情報を探します。一般的な項目の例を次に示し、それらがどこに記載されているかを説明します。

4.1 フレームワークレイヤー

一部の SSP フレームワークモジュールは、次のモジュールの概要リストに含まれていません。それは、モジュールの中には、選択し、スレッドに追加できるにもかかわらず、ローレベルのモジュールでしか使用されず、開発者が個別に使用することが想定されていないものがあるためです。ただし、これらのモジュールは、上位レベルモジュールの「モジュールの概要」には含まれています。そのため、追加情報が必要な場合は、関連する上位レベルの「モジュールの概要」を参照してください。次のリストは、これらの「見つからない」モジュールを探す方法を示しています。

Module	Included In
D/AVE 2D Port on sf_tes_2d_drw	GUIX Port on sf_el_gx under the Express Logic Modules section as GUIX Synergy Port Framework
D/AVE 2D Driver on dave2d	GUIX Port on sf_el_gx under the Express Logic Modules section as GUIX Synergy Port Framework

ADC 周期フレームワーク
オーディオ再生フレームワーク
オーディオ再生 DAC フレームワーク
オーディオ再生 I2S フレームワーク
オーディオ記録 ADC フレームワーク
オーディオ記録 I2S フレームワーク
sf_block_media_lx_nor のブロックメディアフレームワーク
ブロックメディア QSPI フレームワーク
ブロックメディア RAM フレームワーク
ブロックメディア SDMMC フレームワーク
BLE フレームワーク
セルラーフレームワーク
NX 通信フレームワーク
Telnet 通信フレームワーク
USBX ™通信フレームワーク
USBX ™ V2 通信フレームワーク
コンソールフレームワーク
暗号化フレームワーク
静電容量式タッチフレームワーク
静電容量式タッチボタンフレームワーク
静電容量式タッチスライダフレームワーク
外部 IRQ フレームワーク
I2C フレームワーク

JPEG デコード フレームワーク
sf_memory_qspi_nor 上のメモリフレームワーク
メッセージング フレームワーク
パワー プロファイル フレームワーク
パワープロファイル V2 フレームワーク
SPI フレームワーク
スレッド監視フレームワーク
タッチ パネル フレームワーク
タッチパネル V2 フレームワーク
UART 通信フレームワーク
Wi-Fi フレームワーク

4.1.1 ADC 周期フレームワーク

ADC 周期フレームワークには、信号処理アプリケーション用のハイレベル API が用意されています。このモジュールは、使用可能な任意のチャネルを指定したレートで（シングルスキャンモードを使用して）サンプリングするように ADC を構成し、指定したサンプリング繰り返し回数分のデータをバッファに格納してからアプリケーションに通知します。ADC 周期フレームワークでは、Renesas Synergy ™マイクロコントローラの ADC、GPT または AGT、および DTC ペリフェラルを使用します。ユーザー定義のコールバックを作成して、新しいサンプルが利用可能になるたびにデータを処理することができます。

4.1.1.1 ADC 周期フレームワークモジュールの特長

- 16 ビット A/D コンバータ（S1JA）
- 14 ビット A/D コンバータ（S3A7、S3A6、S3A3、S124、S128）
- 12 ビット A/D コンバータ（S7G2、S5D9、S5D5）
- 複数の動作モード
  - シングルスキャン
  - グループスキャン
  - 連続スキャン
- 複数チャネル
  - 1 チャネル（S1JA）
  - 13 チャネル（ユニット 0）、12 チャネル（ユニット 1）（S7G2 および S5D9）
  - 13 チャネル（ユニット 0）、9 チャネル（ユニット 1）（S5D5）
  - 18 チャネル（S124）
  - 21 チャネル（S128）
  - 25 チャネル（S3A6）
  - 28 チャネル（S3A7）
  - 温度センサーチャネル

– 電圧センサーチャネル

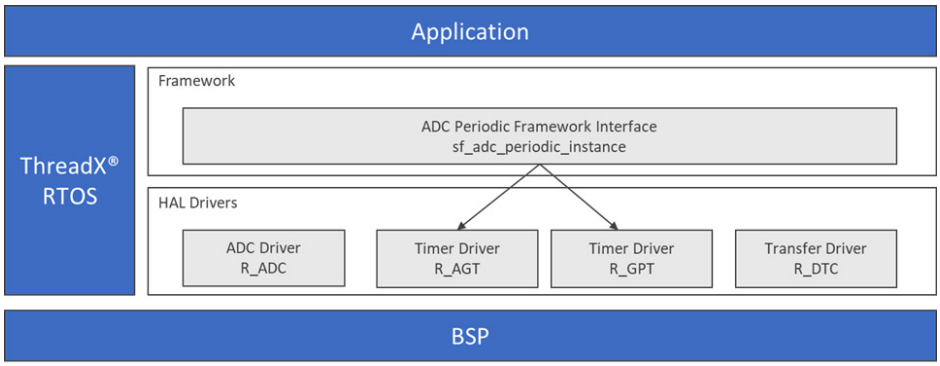


図 1: ADC 周期フレームワークモジュールのブロック図

4.1.1.2 ADC 周期フレームワークモジュール API の概要

ADC 周期フレームワークは、ADC スキャンのオープン、クローズ、開始、および停止の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

ADC 周期フレームワークモジュール API の要約

Function Name	Example API Call and Description
open	g_sf_adc_periodic.p_api->open(g_sf_adc_periodic.p_ctrl, g_sf_adc_periodic.p_cfg); Acquires mutex, then initializes module at the HAL layer
start	g_sf_adc_periodic.p_api->start(g_sf_adc_periodic.p_ctrl); Starts the scan.
stop	g_sf_adc_periodic.p_api->stop(g_sf_adc_periodic.p_ctrl); Stops the hardware trigger (timer) from triggering any more ADC scans.

Function Name	Example API Call and Description
close	g_sf_adc_periodic.p_api->close(g_sf_adc_periodic.p_ctrl); Releases channel mutex and closes channel at HAL layer.
versionGet	g_sf_adc_periodic.p_api->versionGet(&version); Retrieve the API version using the version pointer.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_UNSUPPORTED	Command not found in the current menu
SSP_ERR_NOT_OPEN	Driver control block not valid. Call SF_ADC_PERIODIC_Open to configure.
SSP_ERR_ASSERTION	Version get error - p_version was NULL
SSP_ERR_INTERNAL	An internal ThreadX® error has occurred. This is typically a failure to create/use a mutex or to create an internal thread.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.1.3 ADC 周期フレームワークモジュール動作の概要

ADC 周期フレームワークモジュールは、ADC データをサンプリングしてバッファに格納します。設定されたサンプル数のバッファリングが完了すると、アプリケーションに通知されます。ADC 周期フレームワークは以下のように機能します。

- 初期構成とスキャンプロセスの開始後、フレームワークはハードウェアタイマを使用してワンショットモードで ADC スキャンをトリガします。各スキャンは 1 つ以上のチャネルで構成できます。各ス

キャンが完了すると、DTC によって ADC 割り込みが捕捉され、スキャン結果がユーザー バッファに移動します。

- 各スキャンは " サンプリングの一定回数の繰り返し " と定義され、各スキャンで行われるサンプリング回数はチャネルと等しくなります。チャネルが連続している場合（たとえば、チャネル 1、2、3、4 の場合）、データは順序どおりにキャプチャされます。チャネルが連続していない場合（たとえば、チャネル 1、3、4、5 の場合）は、スキャンごとに、間にある未使用のチャネルからもデータがサンプリングされます。したがって、2 つ目の例では、毎回 5 個のサンプルがユーザー バッファに格納されます。
- ユーザーは、その回数のサンプリングが完了した後に通知を受ける、サンプリング繰り返し回数の総数を指定します。指定されたサンプリング繰り返し回数が実行され、各回のデータがユーザー バッファに格納されたら、バッファ内の有効なデータのインデックスを含むコールバックと、指定された繰り返し回数のサンプリングが完了したことを示すイベントが発生して、ユーザーに通知されます。

ユーザーがスキャンプロセスを停止しない限り、引き続きタイマによってスキャンがトリガされ（AGT または GPT を使用）、データがユーザーバッファに書き込まれます。ユーザーバッファはフレームワークによって循環バッファとして扱われます。バッファの名前と長さは ISDE コンフィギュレータによって指定します。

ADC 周期フレームワークモジュールの動作に関する重要な注意事項と制限事項

1) API によってエラーが返されないようにするには、ADC HAL ドライバーを構成するときに少なくとも 1 つのチャネルを選択する必要があります。
2) ADC フレームワークのスキャンレート（GPT または AGT タイマ期間）を構成する際に、選択したすべてのチャネルをスキャンできる十分な長さの周期を構成してください（Synergy S7G2 デバイスでは、各チャネル変換に約 2 マイクロ秒かかります）。
3) ADC 周期フレームワークは、各スキャンで収集したすべてのチャネルのデータをユーザー指定のバッファに格納します。指定されたサンプリング繰り返し回数が完了すると、ユーザーに通知されます。5 つのチャネルを選択しており（チャネル 1、2、3、4、5）、サンプルカウントを 3 に設定している場合は、5 x 3 = 15 のサンプルが使用可能になるとユーザーに通知されます。サンプルの順序は次のようになります。

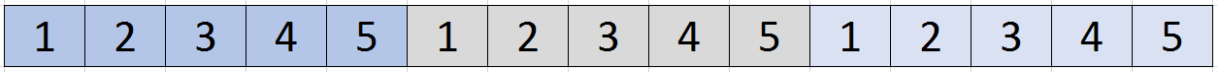


図 2: ADC 周期フレームワークモジュールのサンプル順序

ADC 周期フレームワーク構成でデータバッファ長を選択するときは、バッファの長さが、生成されるサンプル数の長さの 2 倍以上（この例では 15 x 2 = 30 以上）になるようにしてください。そのようにする理由は、ユーザーアプリケーションがデータ使用可能の通知を受けた後も、引き続き新しいデータがサンプルレートでバッファリングされるためです。このバッファは循環バッファとして扱われるため、データが意図せず上書きされる可能性があります。つまり、バッファサイズが生成されるサンプル数以下の場合は、アプリケーションでデータが使用可能になる前にデータが上書きされてしまいます。

アプリケーション コールバックによって、有効なデータが存在するバッファ内の適切な位置を指すインデックスが通知されます。

- ADC 周期フレームワークでは現在、以下の機能はサポートされていません。
  - グループスキャンモードの使用
  - DMA の使用
- このフレームワークで使用する ADC チャネルを構成する際に、他の使用可能なチャネルも併せて選択する場合は、温度センサーまたは電圧センサーを選択しないでください。温度センサーだけ、電圧センサーだけ、または任意の数の通常 ADC チャネルを使用できます。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.1.4 アプリケーションへの ADC 周期フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに ADC 周期フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

ADC 周期フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(ADC 周期フレームワークのデフォルト名は g_adc_periodic0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

ADC 周期フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_adc_periodic0 ADC Periodic Framework on sf_adc_periodic	Threads	New Stack> Driver> Analog> ADC Periodic Framework on sf_adc_periodic

次の図に示すように、sf_adc_periodic の ADC 周期フレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

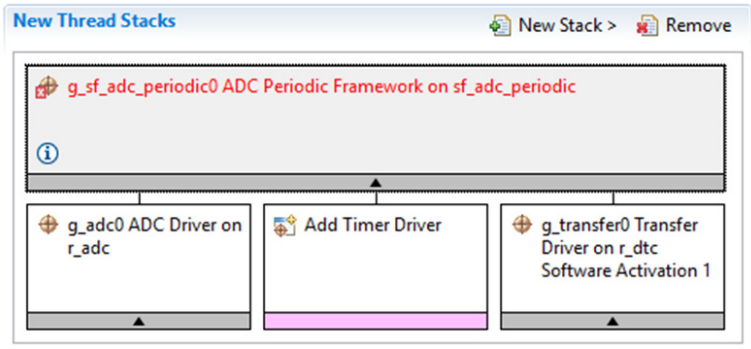


図 3: ADC 周期フレームワークモジュールのスタック

4.1.1.5 ADC 周期フレームワークモジュールの構成

ユーザーは、必要な動作に合わせて ADC 周期フレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_adc_periodic 上の ADC 周期フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_sf_adc_periodic0	Module name.
Name of the data-buffer to store samples	g_user_buffer	Name of the 16-bit data buffer to store samples.
Length of the data-buffer	128	Length of the buffer to which data is to be stored.
Number of sampling iterations	10	Priority of ADC Periodic Framework internal thread.
Callback	g_adc_framework_user_callback	User function that will be called once “sample_counts” number of data has been buffered.
Name of generated initialization function	sf_adc_periodic_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ADC 周期フレームワークモジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_adc の ADC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: Enabled	If selected code for parameter checking is included in the build.
Name	g_adc0	Module name
Unit	0, 1 (S7G2 Only) Default: 0	Specify the ADC Unit to be used. The S7G2 has two units; 0 and 1.
Resolution	14-Bit (S3A7/S124 Only), 12-Bit, 10-Bit (S7G2) Default: 8-Bit (S7G2 Only)	Specify the conversion resolution for this unit.
Alignment	Right, Left Default: Right	Specify the conversion result alignment.
Clear after read	Off, On Default: On	Specify if the result register must be automatically cleared after the conversion result is read. Note: If this is enabled, then watching the result register using a debugger always results in a 0.

ISDE Property	Value	Description
Mode	Single Scan	The ADC Framework preconfigures and locks this field.
Channels 0-6	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 7-10 (S3A7/S124 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 11-15 (S3A7 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 16-20	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.

ISDE Property	Value	Description
Channel 21 (Unit 0 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channel 22 (S3A7/S124 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 23-27 (S3A7 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Temperature Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	Temperature sensor use selection for Channel Scan Mask
Voltage Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	Voltage sensor use selection for Channel Scan Mask
Normal/Group A Trigger	ELC Event	The ADC Framework preconfigures and locks this field.

ISDE Property	Value	Description
Group B Trigger (Valid Only in Group Scan Mode)	ELC Event (The only valid trigger for either group in Group Scan Mode)	The ADC Framework preconfigures and locks this field.
Group Priority (Valid only in Group Scan Mode)	Group A cannot interrupt Group B, Group A can interrupt Group B; Group B scan restarts at next trigger, Group A can interrupt Group B; Group B scan restarts immediately, Group A can interrupt Group B; Group B scan restarts immediately and scans continuously Default: Group A cannot interrupt Group B	Do not use with ADC Framework since the mode is locked to Single Scan Mode.
Add/Average Count	Disabled, Add two samples, Add three samples, Add four samples, Add sixteen samples, Average two samples, Average four samples Default: Disabled	Specify if addition or averaging needs to be done for any of the channels in this unit. The actual channels are specified by using a channel mask adc_channel_cfg_t::add_mask.
Channels 0-27	Disabled, Enabled Default: Disabled	This field is valid only if adc_cfg_t::add_average_count is enabled. This field determines what channels results are to be averaged or summed.
Temperature Sensor	Disabled, Enabled Default: Disabled	Temperature sensor use selection for Addition/Averaging Mask
Voltage Sensor	Disabled, Enabled Default: Disabled	Voltage sensor use selection for Addition/Averaging Mask

ISDE Property	Value	Description
Channels 0-2	Disabled, Enabled Default: Disabled	Determines which of channels 0, 1 and 2 are using the updated sample-and-hold states value specified in adc_channel_cfg_t::sample _hold _states. This field must only be set if it is desired to modify the default sample and hold count value for channels 0, 1 and 2.
Sample Hold States (Applies only to the 3 channels selected above)	24	Specifies the updated sampleand-hold count for the channel dedicated sample-and-hold circuit. This field is valid only if adc_channel_cfg_t::sample _hold _mask is not 0. Only channels 0, 1 and 2 have dedicated sample and hold circuits. Note: Use this to modify the default number of states (24) for which the value is sampled. Each state is equal to 1/ADCLK time.
Callback	NULL	The ADC Framework uses the callback internally.
Scan End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Scan End Interrupt Priority selection
Scan End Group B Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Scan End Group B Interrupt Priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_agt 上の AGT HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables parameter checking.
Name	g_timer0	Module name.
Channel	0	Physical hardware channel.
Mode	Periodic	Warning: One Shot functionality is not available in the GPT hardware, so it is implemented in software by stopping the timer in the ISR called when the period expires. For this reason, ISR's must be enabled for one-shot mode even if the callback is unused.
Period Value	10	See Timer Period Calculation
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz Default: Microseconds	See Timer Period Calculation
Auto Start	False	Set to true to start the timer after configuring or false to leave the timer stopped until timer_api_t::start is called.
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub Default: PCLKB	The clock source for the AGT counter.
AGTO Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for AGT (AGTO pin). Set to false for no output of the timer signal.

ISDE Property	Value	Description
AGTIO Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for AGT (AGTIO pin). Set to false for no output of the timer signal.
Output Inverted	True, False Default: True	Set to false to start the output signal low. Set to true to start the output signal high.
Enable comparator A output pin	True, False Default: False	Enable comparator A output pin selection
Enable comparator B output pin	True, False Default: False	Enable comparator B output pin selection
Callback	NULL	A user callback function can be registered in timer_api_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the timer period elapses. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Underflow Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Timer interrupt priority. 0 is the highest priority.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_gpt 上の GPT HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_timer0	Module name.
Channel	0	The ADC Framework preconfigures and locks this field based on channel selected in the ADC Framework.
Mode	Periodic	The ADC Framework preconfigures and locks this field.
Period Value	10	Configure timer period to trigger ADC scans.
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz Default: Milliseconds	Configure units of the timer period set above.
Duty Cycle Value	50	Duty cycle value selection
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	Duty cycle unit selection
Auto Start	False	The ADC Framework preconfigures and locks this field.

ISDE Property	Value	Description
GTIOCA Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
GTIOCB Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
Callback	NULL	The ADC Framework preconfigures and locks this field.
Overflow Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Software Activation 上の DTC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build

ISDE Property	Value	Description
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Block	Mode selection
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	1	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	1	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation 1	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection

ISDE Property	Value	Description
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ADC 周期フレームワークモジュールのピン構成

チャネルにアクセスするには、ISDE の [Pins] タブで ADC チャネルを設定する必要があります。次の表は、[SSP configuration] ウィンドウ内でのピンの選択方法を示しています。

r_adc での ADC HAL モジュール用のピンの選択

Resource	ISDE Tab	Pin selection Sequence
ADC	Pins	Select Peripherals > Analog Pins > ADC0/1 > AN_XX

注 : 内部温度センサーおよび内部電圧センサーの場合、ピン構成は必要ありません。

4.1.1.6 アプリケーションでの ADC 周期フレームワークモジュールの使用

一般的なアプリケーションの sf_audio_record_adc で ADC 周期フレームワークモジュールを使用する際の手順は次のとおりです。

1) sf_adc_periodic_api_t::open を使用して ADC を初期化します。
2) sf_adc_periodic_api_t::start API を使用してチャネルのスキャンを開始します。
3) sf_adc_periodic_api_t::stop API を使用してスキャンを停止します。
4) アプリケーションコードで callback を使用して変換結果を読み取ります。
5) sf_adc_periodic_api_t::close API を使用してインスタンスを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

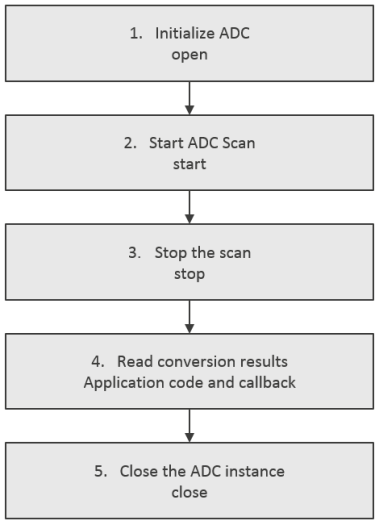


図 4: **一般的な ADC 周期フレームワークモジュールアプリケーションのフロー図**

4.1.2 オーディオ再生フレームワーク

sf_audio_playback HAL モジュールのオーディオ再生フレームワークはオーディオ再生用のハイレベル API を実装し、sf_audio_playback_hw_dac または sf_audio_playback_hw_i2s. のいずれかに実装されます。このモジュールは、16 および 8 ビットパルス符号変調（PCM）サンプルを再生するために必要な同期を処理します。オーディオ再生フレームワークは、Synergy MCU の DAC（DAC12 または DAC8）または I2S、タイマ（AGT または GPT）、データ転送（DMA または DTC）ペリフェラルを使用します。ユーザー定義のコールバックを作成し、追加データが必要な場合に対応できます。

4.1.2.1 オーディオ再生フレームワークモジュールの特長

- データを扱いやすいチャンクに分割することによって長いバッファを再生します。
- ThreadX タイムアウトが発生するまで再生を繰り返します（正弦波音やループするバックグラウンドミュージックのように音声が繰り返される場合）。
- 最後のバッファの再生が開始された後、コールバックを使用して次のデータを要求します。
- ソフトウェア音量制御。
- 一時停止と再開機能。
- スケーリング。たとえば、有符号 16 ビット PCM データを無符号 12 ビット DAC または無符号 8 ビット DAC8 の範囲にシフトします。
- 複数のストリームについての基本的なミキシング。

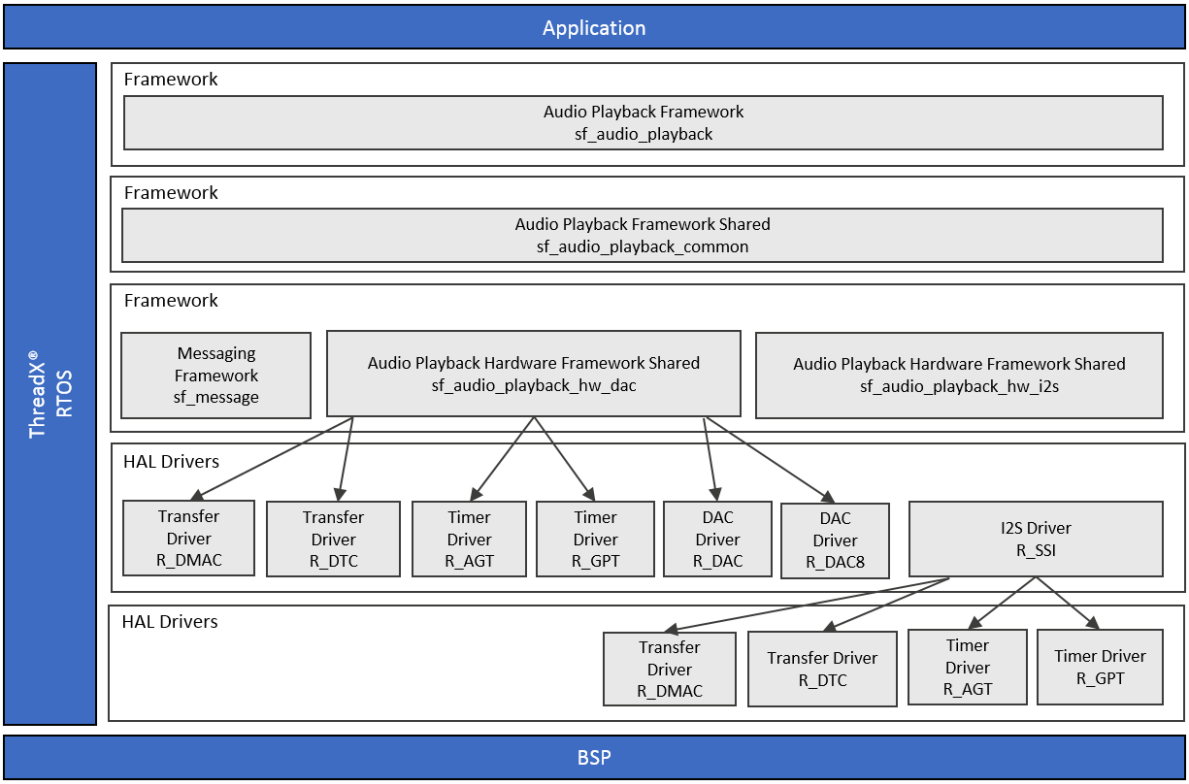


図 5: **オーディオ再生フレームワークモジュールのブロック図**

4.1.2.2 オーディオ再生フレームワークモジュールの API の概要

オーディオ再生フレームワークモジュールは、オープン、開始、再生、停止などの操作の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

オーディオ再生フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_audio_playback0.p_api->open(g_sf_audio_playback0.p_ctrl, g_sf_audio_playback0.p_cfg); Open a device channel for read/write and control.

Function Name	Example API Call and Description
start	g_sf_audio_playback0.p_api->start(g_sf_audio_playback0.p_ctrl, p_data, Timeout); Start Audio Playback Hardware.
pause	g_sf_audio_playback0.p_api->pause(g_sf_audio_playback0.p_ctrl); Pause Audio Playback Hardware.
stop	g_sf_audio_playback0.p_api->stop(g_sf_audio_playback0.p_ctrl); Stop Audio Playback Hardware.
resume	g_sf_audio_playback0.p_api->resume(g_sf_audio_playback0.p_ctrl, &buffer, length); Resume playback.
volumeSet	g_sf_audio_playback0.p_api->volumeSet(g_sf_audio_playback0.p_ctrl, volume); Sets volume.
close	g_sf_audio_playback0.p_api->close(g_sf_audio_playback0.p_ctrl); Close the audio module.
versionGet	g_sf_audio_playback0.p_api->versionGet(& version); Return the version of the module using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successful.
SSP_ERR_ASSERTION	A pointer is NULL or a parameter is invalid.
SSP_ERR_OUT_OF_MEMORY	The number of streams open at once is limited to SF_AUDIO_PLAYBACK_CFG_MAX_STREAMS. If this number is exceeded, an out of memory error occurs.
SSP_ERR_TIMEOUT	Timeout occurred before playback finished.
SSP_ERR_NOT_OPEN	The stream control block p_ctrl is not initialized.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.2.3 オーディオ再生フレームワークモジュールの動作の概要

オーディオ再生フレームワークモジュールは、オーディオ再生をサポートするためのスレッドを内部で作成します。下図に、オーディオ再生フレームワークのスレッドとパブリックオーディオ再生フレームワーク API 関数とのやり取りを表すフローチャートを示します。

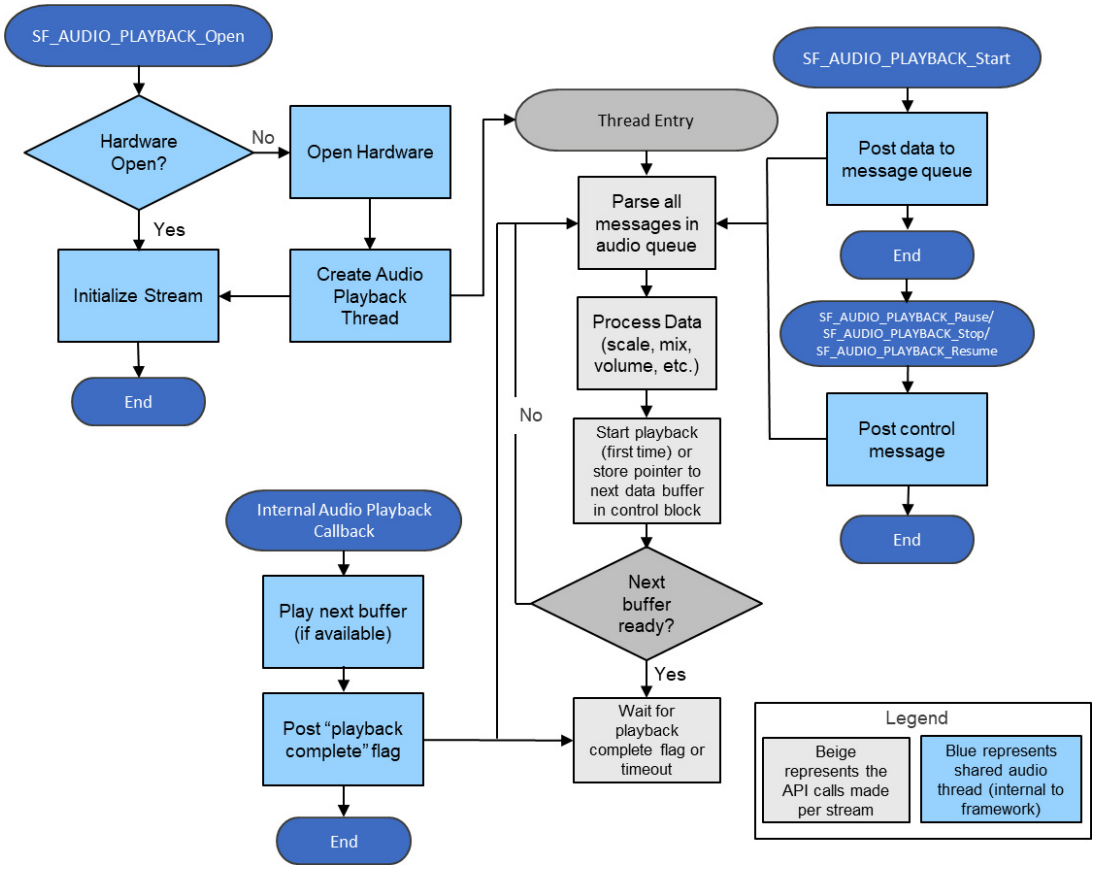


図 6: オーディオ再生フレームワークモジュールのフローチャート

オーディオ再生フレームワークの推奨される使用方法を以下に示します。

- セマフォを作成します（たとえば、g_sf_audio_playback_semaphore）。これは [スレッド] タブで行うことができます。初期値を 2 に設定します (オーディオ再生フレームワークでは、ストリームあたり最大 2 つのデータ メッセージを保持できます)。
- コールバック関数を作成します（たとえば、sf_audio_playback_callback）。オーディオ再生フレームワーク インスタンスにコールバック関数の名前を入力します。このコールバック関数は、オーディオ再生フレームワークがデータの処理を終了したときに呼び出されます。コールバックで、上記で作成したセマフォを配置します。
- メイン ループ内で、データを再生する前にセマフォを取得します。データを再生するには、まずメッセージング フレームワークからバッファを取得し、次にオーディオ再生データ構造をバッファ内に作成します。

オーディオ再生フレームワークは、単一のハードウェアポート上の複数のオーディオストリームをサポートします。2 つのストリームを使用する場合に必要なモジュールのブロック図を下図に示します。

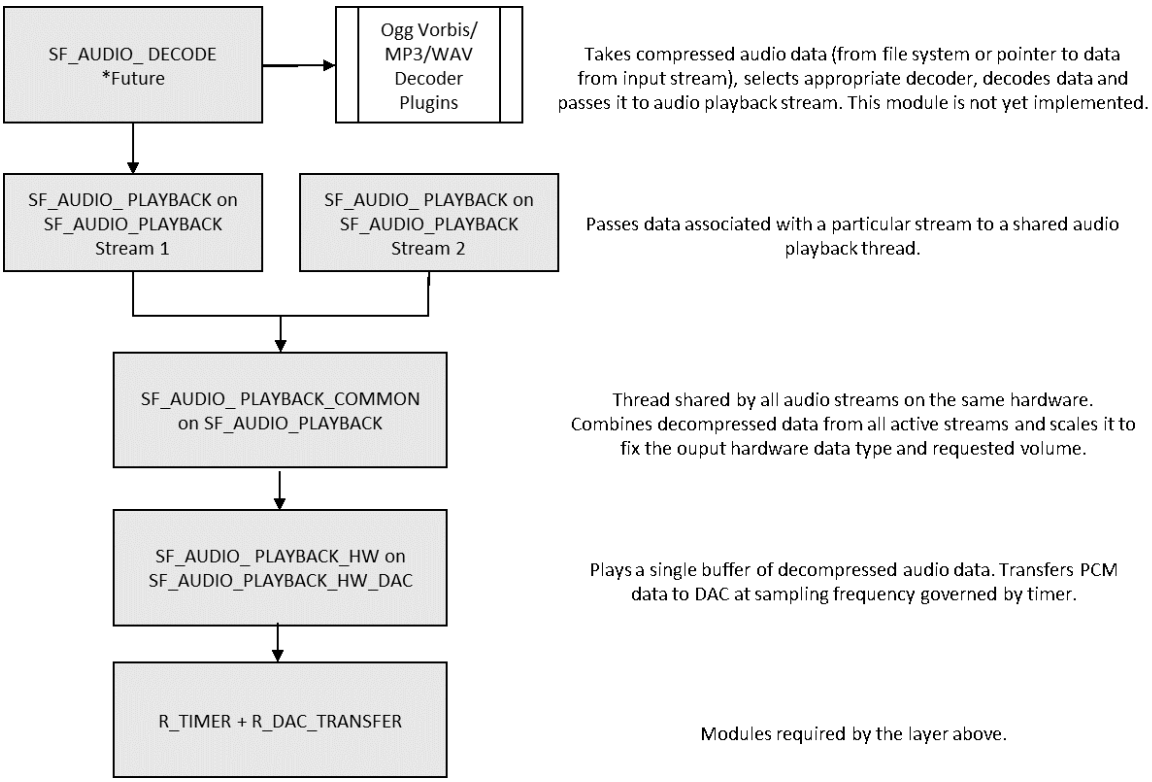


図 7: オーディオ再生フレームワークモジュールのオーディオストリーム

オーディオ再生フレームワークモジュールの動作に関する重要な注意事項と制限事項

メッセージの構成

[Messaging] タブのメッセージフレームワークコンフィギュレータを使用して、メッセージフレームワークを設定します。

1) オーディオ再生イベント クラスをハイライトします。
2) 新しいサブスクライバを追加します。以下の構成を選択して、sf_audio_playback モジュール上のオーディオ再生フレームワークの [Properties] タブでメッセージクラスインスタンスが開始インスタンスと終了インスタンスの間に適切に設定されていることを確認します。
   - スレッド : アプリケーション内の任意のスレッド。
   - 開始 : アプリケーションで使用されている最初のオーディオ インスタンス。
   - 終了 : アプリケーションで使用されている最後のオーディオ インスタンス。
3) オーディオ再生サブスクライバーで新しいサブスクライバーをハイライトします。シンボル名を記録します。
4) [スレッド] タブに戻ります。
5) HAL/ 共通のオーディオ再生フレームワーク共有モジュールをハイライトして、オーディオ メッセージ キュー名をオーディオ再生サブスクライバーのシンボル名に設定します。

I2S 実装の使用

オーディオ フレームワーク I2S ハードウェア ポートには、I2S モジュールに対する依存関係があります。I2S ドライバー モジュールは DTC を使用して高速化できます (推奨)。

- I2S ドライバーモジュールの統合ソリューション開発環境プロパティを設定します。
  - オーディオ クロック周波数 (ヘルツ単位) を、使用されている入力オーディオ クロックの周波数に設定します。
  - サンプリング周波数 (ヘルツ単位) をオーディオ データのサンプリング周波数に設定します。
  - データ ビットとワード長を 16 ビットに設定します (オーディオ フレームワークでは 16 ビットのみが受け入れられます)。
  - SSIn TXI および SSIn INT 割り込みを有効にします。
- r_dtc 上の転送モジュールは自動で追加されます。

DAC 実装の使用

オーディオ フレームワーク DAC ハードウェア ポートには、タイマ、DAC、および転送 API モジュールに対する依存関係があります。

- タイマ モジュールを追加します。
  - 周波数 (Hz 単位) をオーディオ データのサンプリング周波数に設定します。
  - DTC を転送モジュールとして使用する場合は、割り込みを有効にします。
- DAC（DAC12 または DAC8）モジュールを追加します。
- r_dtc. 転送モジュールを追加します。
  - DAC チャネル 0 を使用する場合は宛先ポインタを &R_DAC->DADRn[0] に設定し、DAC チャネル 1 を使用する場合は &R_DAC->DADRn[1] に設定します。
  - DAC8 チャネル 2（S128）を使用する場合は宛先ポインタを &R_DAC8->DADRn[2] に設定し、DAC8 チャネル 0（S1JA）を使用する場合は &R_DAC8->DADRn[0] に設定します。
  - アクティベーション ソースを上で選択したタイマ割り込みに設定します。

動作に関するその他の注意事項

使用されるキューは、[Audio Playback Framework Shared on sf_audio_playback] の [Properties] で指定された名前と一致する必要があります (デフォルトは g_sf_audio_playback_queue).

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.2.4 アプリケーションへのオーディオ再生フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにオーディオ再生フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

オーディオ再生フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。(オーディオ再生フレームワークモジュールのデフォルト名は g_sf_audio_playback0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

オーディオ再生フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_audio_playback Audio Playback Framework on g_sf_audio_playback	Threads	New Stack> Framework> Audio> Audio Playback Framework on g_sf_audio_playback

次の図に示すように、sf_audio_playback のオーディオ再生フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

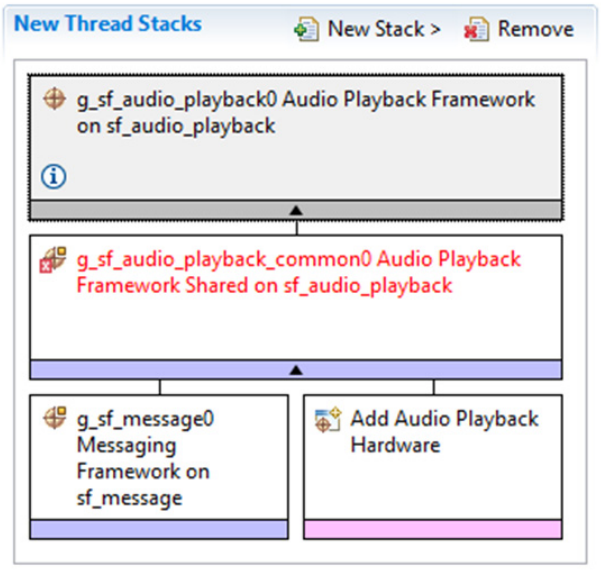


図 8: オーディオ再生フレームワークモジュールスタック

4.1.2.5 オーディオ再生フレームワークモジュールの構成

オーディオ再生フレームワークモジュールは、必要な動作に合わせてユーザーが構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成テーブルの設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_audio_playback のオーディオ再生フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Buffer Size Bytes	512	Buffer size bytes selection.
Maximum Number of Streams	1	Maximum number of streams selection.
Thread Stack Size	512	Thread stack size selection.
Name	g_sf_audio_playback0	Module name.
Message Class Instance	0	Message class instance selection.
Callback	NULL	Callback selection.
Name of generated initialization function	sf_audio_playback_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、ターゲットハードウェアの実装に基づいた DAC チャネルの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

オーディオ再生フレームワークのローレベルモジュールの構成

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

sf_audio_playback 上のオーディオ再生フレームワーク共有の構成設定

ISDE Property	Value	Description
Name	g_sf_audio_playback_common0	Module name.
Thread Priority	3	Thread priority selection.
Audio Message Queue Name	g_sf_audio_playback_queue	Audio message queue name selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_message 上のメッセージフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Message Queue Depth (Total number of messages to be enqueued in a Message Queue)	16	Specify the size of Thread X Message Queue in bytes for Message Subscribers. This value is applied to all the Message Queues.

ISDE Property	Value	Description
Name	g_sf_message0	The name of Messaging Framework module control block instance.
Work memory size in bytes	2048	Specify the work memory size in bytes. Choosing a small number results a small number of buffers which can be allocated at the same time (You can confirm the total buffer number on: sf_message_ctrl_t:: number_of_buffers). If the value is smaller than the peak number of messages to be posted at the same time, the Framework occurs a buffer allocation failure affecting system performance.
Pointer to subscriber list array	p_subscriber_lists	Specify the name of pointer to the Subscriber List array.
name of the block pool internally used in the messaging framework	sf_msg_blk_pool	The name of memory block memory the Framework creates in the control block. This parameter might be useful for debugging purpose but NULL can be specified for saving memory.
Name of generated initialization function	sf_message_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_audio_playback_hw_dac 上のオーディオ再生ハードウェアフレームワーク共有の構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
DMAC Support	Disabled, Enabled Default: Disabled	DMAC support selection.
Name	g_sf_audio_playback_hw0	Module name.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dmac Software Activation の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Name	g_transfer0	Module name
Channel	0	Channel selection
Mode	Normal	Mode selection
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Destination Pointer	NULL	Destination pointer selection

ISDE Property	Value	Description
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source	Software Activation, Peripheral Events Default: Software Activation	Activation source selection
Auto Enable	False	Auto enable selection
Callback	NULL	Callback selection
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dmac Software Activation 1 の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section selection
Name	g_transfer0	Module name

ISDE Property	Value	Description
Mode	Normal	Mode selection
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events Default: Software Activation 1	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_agt のタイマドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables parameter checking.
Name	g_timer0	Module name.
Channel	0	Physical hardware channel.
Mode	Periodic	Warning: One Shot functionality is not available in the GPT hardware, so it is implemented in software by stopping the timer in the ISR called when the period expires. For this reason, ISR's must be enabled for one-shot mode even if the callback is unused.
Period Value	10	See Timer Period Calculation
Period Unit	Hertz	See Timer Period Calculation
Auto Start	False	Set to true to start the timer after configuring or false to leave the timer stopped until timer_api_t::start is called.
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub Default: PCLKB	The clock source for the AGT counter.
AGTO Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for AGT (AGTO pin). Set to false for no output of the timer signal.

ISDE Property	Value	Description
AGTIO Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for AGT (AGTIO pin). Set to false for no output of the timer signal.
Output Inverted	True, False Default: False	Set to false to start the output signal low. Set to true to start the output signal high.
Enable comparator A output pin	True, False Default: False	Enable comparator A output pin selection
Enable comparator B output pin	True, False Default: False	Enable comparator B output pin selection
Callback	NULL	A user callback function can be registered in timer_api_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the timer period elapses. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Timer interrupt priority. 0 is the highest priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_gpt のタイマドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_timer0	Module name.
Channel	0	Channel selection.
Mode	Periodic	Warning: One Shot functionality is not available in the GPT hardware, so it is implemented in software by stopping the timer in the ISR called when the period expires. For this reason, ISR's must be enabled for one-shot mode even if the callback is unused.
Period Value	10	See Timer Period Calculation
Period Unit	Hertz	See Timer Period Calculation
Duty Cycle Value	50	Duty cycle value selection
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	Duty cycle unit selection
Auto Start	False	Set to true to start the timer after configuring or false to leave the timer stopped until timer_api_t::start is called.
GTIOCA Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.

ISDE Property	Value	Description
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
GTIOCB Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
Callback	NULL	A user callback function can be registered in timer_api_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the timer period elapses. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dac の DAC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_dac0	Module name.
Channel	0	Set to 0 for output DA0 or 1 for output DA1.
Synchronize with ADC	Enabled, Disabled Default: Disabled	Set to true for anti-interference synchronization with the Analog-to-Digital Converter (ADC) Module. Set to false false if power supply interference between the analog modules is not a problem, or if asynchronous conversion by the DAC Module is desired.
Data Format	Right Justified	Set to zero, if 12-bit data values are loaded in bits 11 through 0, or right justified. Set to one, if 12-bit data values are loaded in bits 15 through 4, or left justified.
Output Amplifier	Enable, Disable Default: Disable	Set to true, if output amplifier hardware function is desired. Set to false to bypass output amplifier hardware function.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_audio_playback_hw_i2s のオーディオ再生フレームワーク共有の構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_sf_audio_playback_hw0	Module name.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_ssi の I2S HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_i2s0	Module name.
Channel	0	Physical hardware channel.
Audio Clock Frequency (Hertz)	2822400	Input audio clock frequency, used to generate the I2S clock. Must be a multiple between 1 and 128 of: (sampling_freq_hz * word_length_in_bits)
Sampling Frequecy (Hertz)	44100	Sampling frequency of audio data.
Data Bits	8 bits, 16, 18, 20, 22, 24 Default: 16 bits	Bit depth of audio data, which is the size in bits of one sample of audio data.
Word Length	8 bits, 16, 24, 32 Default: 16 bits	Word length of audio data, must be at least the same size as the bit depth (Data Bits field).

ISDE Property	Value	Description
WS Continue Mode	Enabled, Disabled Default: Disabled	Enable WS continue mode to continue to output the word select line when the peripheral is idle. Disable to stop outputting the word select line when the peripheral is idle.
Name of I2S callback function to be defined by user	NULL	A user callback function must be registered in open. The callback will be called from the interrupt service routine (ISR) when the transmission FIFO reaches the high watermark point after all data for transmission is transmitted or when reception is complete (the requested number of bytes have been received). Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Transmit interrupt priority selection
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Receive interrupt priority selection

ISDE Property	Value	Description
Idle/Error Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Idle/error interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Software Activation 1 の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection.
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Driver name.
Mode	Normal	Mode selection.
Transfer Size	4 Bytes	Transfer size selection.
Destination Address Mode	Fixed	Destination address mode selection.
Source Address Mode	Incremented	Source address mode selection.
Repeat Area (Unused in Normal Mode	Source	Repeat area selection.
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection.

ISDE Property	Value	Description
Destination Pointer	NULL	Destination pointer selection.
Source Pointer	NULL	Source pointer selection.
Number of Transfers	0	Number of transfers selection.
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection.
Activation Source (Must enable IRQ)	Software Activation 1, Software Activiation 2, Peripheral Events Default: Software Activation 1	Activation source selection.
Auto Enable	False	Auto enable selection.
Callback (Only valid with Software start)	NULL	Callback selection.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Software Activation 1 の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection.

ISDE Property	Value	Description
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer1	Driver name.
Mode	Normal	Mode selection.
Transfer Size	4 Bytes	Transfer size selection.
Destination Address Mode	Incremented	Destination address mode selection.
Source Address Mode	Fixed	Source address mode selection.
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection.
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection.
Destination Pointer	NULL	Destination pointer selection.
Source Pointer	NULL	Source pointer selection.
Number of Transfers	0	Number of transfers selection.
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection.
Activation Source (Must enable IRQ)	Software Activation 1, Software Activiation 2, Peripheral Events Default: Software Activation 1	Activation source selection.
Auto Enable	False	Auto enable selection.
Callback (Only valid with Software start)	NULL	Callback selection.

ISDE Property	Value	Description
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_agt のタイマドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection.
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer1	Driver name.
Mode	Normal	Mode selection.
Transfer Size	4 Bytes	Transfer size selection.
Destination Address Mode	Incremented	Destination address mode selection.
Source Address Mode	Fixed	Source address mode selection.
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection.
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection.

ISDE Property	Value	Description
Destination Pointer	NULL	Destination pointer selection.
Source Pointer	NULL	Source pointer selection.
Number of Transfers	0	Number of transfers selection.
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection.
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events Default: Software Activation 1	Activation source selection.
Auto Enable	False	Auto enable selection.
Callback (Only valid with Software start)	NULL	Callback selection.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_gpt のタイマドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection
Name	g_timer0	Module name
Channel	0	Channel selection

ISDE Property	Value	Description
Mode	Periodic	Mode selection
Period Value	2822400 *2	Period value selection
Period Unit	Hertz	Period unit selection
Duty Cycle Value	50	Duty cycle value selection
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	Duty cycle unit selection
Auto Start	FALSE	Auto start selection
GTIOCA Output Enabled	True, False Default: False	GTIOCA output enabled selection
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCA stop level selection
GTIOCB Output Enabled	True, False Default: False	GTIOCB output enabled selection
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCB stop level selection
Callback	NULL	Callback selection
Overflow Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

オーディオ再生フレームワークモジュールのクロック構成

オーディオ再生フレームワークハードウェアモジュールは、[Clock configuration] ウィンドウにあるペリフェラルクロックを使用します。

オーディオ再生フレームワークモジュールのピン構成

DAC または SSI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は、[SSP configuration] ウィンドウ内でのピンの選択方法と、関連するピンの選択例を示します。

オーディオ再生フレームワークモジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
DAC	Pins	Select Peripherals > Analog:DAC12 > DAC120/121
SSI	Pins	Select Peripherals > Connectivity:SSI > SSI/0/1

注 : 選択シーケンスでは、DAC0/DAC1 または SSI0/SSI1 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_dac の DAC ドライバーのピン構成設定

Pin Configuration Property	Value	Description
Operation Mode	Enabled, Disabled	Operation selection
DAC	None, P014 (Default: P014)	DAC pin selection

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.1.2.6 アプリケーションでのオーディオ再生フレームワークモジュールの使用

アプリケーションでオーディオ再生フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_audio_playback_api_t::open API を使用してオーディオストリームを初期化します。
2) コールバック関数を使用して、必要なバッファ数と初期数が等しいセマフォにポストします。アプリケーションコードに実装します。

注 : アプリケーションスレッドでこのセマフォを取得してから sf_audio_playback_api_t::start API をコールします。

3) sf_message_api_t::bufferAcquire API を使用して、メッセージフレームワークからバッファを取得します。

   注 : バッファ内で sf_audio_playback_data_t を使用してオーディオフレームワークデータ構造体を作成します。

4) sf_audio_playback_api_t::start API を使用してオーディオ再生フレームワークを開始します。

注 : 複数のストリームが必要な場合は、追加のオーディオストリームのために手順 1 ～ 4 を繰り返します。個別のオーディオストリームを使用する必要があるのは、ミキシングを使用してストリームを同時に再生する必要がある場合のみです。オーディオが常に連続で順次再生され、オーバーラップしない場合は、そのストリームを再利用できます。

これらの共通の手順を、以下の図に示す一般的な動作フローで説明します。

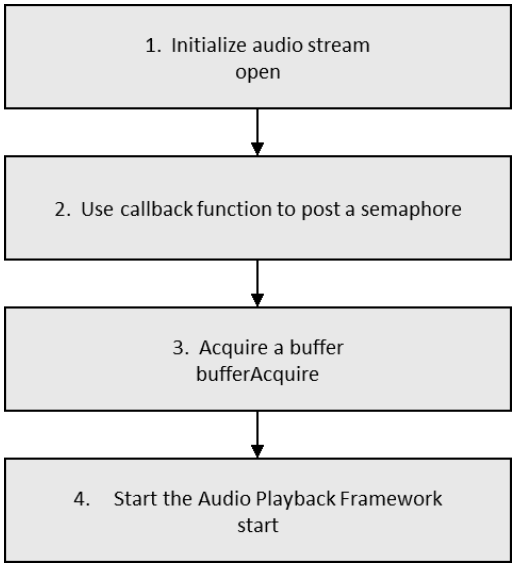


図 9: 通常のオーディオ再生フレームワークモジュールアプリケーションのフロー図

4.1.3 オーディオ再生 DAC フレームワーク

このオーディオ再生フレームワーク DAC モジュールは、オーディオ再生アプリケーション用のハイレベルの API を提供し、8 ビットおよび 16 ビットパルス符号変調（PCM）サンプルを再生するために必要な同期を処理します。オーディオ再生 DAC フレームワークは、Synergy MCU の DAC/DAC8、タイマ（AGT または GPT）、データ転送（DMA または DTC）ペリフェラルを使用します。ユーザー定義のコールバックを作成し、追加データが必要な場合に対応できます。

4.1.3.1 オーディオ再生 DAC フレームワークモジュールの特長

- データを扱いやすいチャンクに分割することによって長いバッファを再生します。
- ThreadX タイムアウトが発生するまで再生を繰り返します（正弦波音やループするバックグラウンドミュージックのように音声が繰り返される場合）。
- 最後のバッファの再生が開始された後、コールバックを使用して次のデータを要求します。

- ソフトウェア音量制御。
- 一時停止と再開の機能。
- スケーリング。たとえば、有符号 16 ビット PCM データを無符号 12 ビットまたは 8 ビット DAC の範囲にシフトします。
- 複数のストリームについての基本的なミキシング。

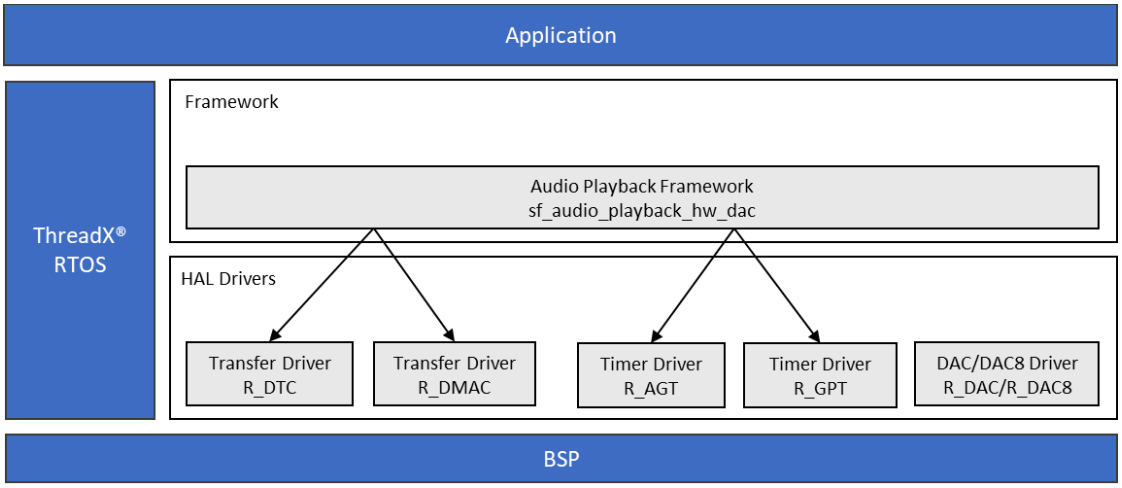


図 10:オーディオ再生 DAC フレームワークモジュールのブロック図

注 :DAC または DAC8 ドライバーの選択オプションは MCU 固有です。

4.1.3.2 オーディオ再生 DAC フレームワークモジュールの API の概要

オーディオ再生 DAC フレームワークモジュールは、オープン、開始、再生、および停止などの操作の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

オーディオ再生 DAC フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_audio_playback_hw0.p_api->open(g_sf_audio_playback_hw0.p_ctrl, g_sf_audio_playback_hw0.p_cfg); Open a device channel for read/write and control.

Function Name	Example API Call and Description
start	g_sf_audio_playback_hw0.p_api->start(g_sf_audio_playback_hw0.p_ctrl, &p_data, Timeout); Start Audio Playback Hardware.
stop	g_sf_audio_playback_hw0.p_api->stop(g_sf_audio_playback_hw0.p_ctrl); Stop Audio Playback Hardware.
play	g_sf_audio_playback_hw0.p_api->play (g_sf_audio_playback_hw0.p_ctrl, p_buffer, length); Play audio buffer.
dataTypeGet	g_sf_audio_playback_hw0.p_api->dataType Get(g_sf_audio_playback_hw0.p_ctrl, p_data_type); Stores expected data type in provided pointer p_data_type.
close	g_sf_audio_playback_hw0.p_api->close(g_sf_audio_playback_hw0.p_ctrl); Close the audio module.
versionGet	g_sf_audio_playback_hw0.p_api->versionG et(&version); Return the version of the module with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successful.
SSP_ERR_ASSERTION	A pointer is NULL or a parameter is invalid.
SSP_ERR_OUT_OF_MEMORY	The number of streams open at once is limited to SF_AUDIO_PLAYBACK_CFG_MAX_STREAMS. If this number is exceeded, an out of memory error occurs.
SSP_ERR_TIMEOUT	Timeout occurred before playback finished.
SSP_ERR_NOT_OPEN	The stream control block p_ctrl is not initialized.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.3.3 オーディオ DAC フレームワークモジュールの動作の概要

DAC オーディオ再生フレームワークモジュールは、オーディオ再生をサポートするためのスレッドを内部で作成します。下図に、オーディオ再生 DAC フレームワークのスレッドと API とのやり取りを表すフローチャートを示します。

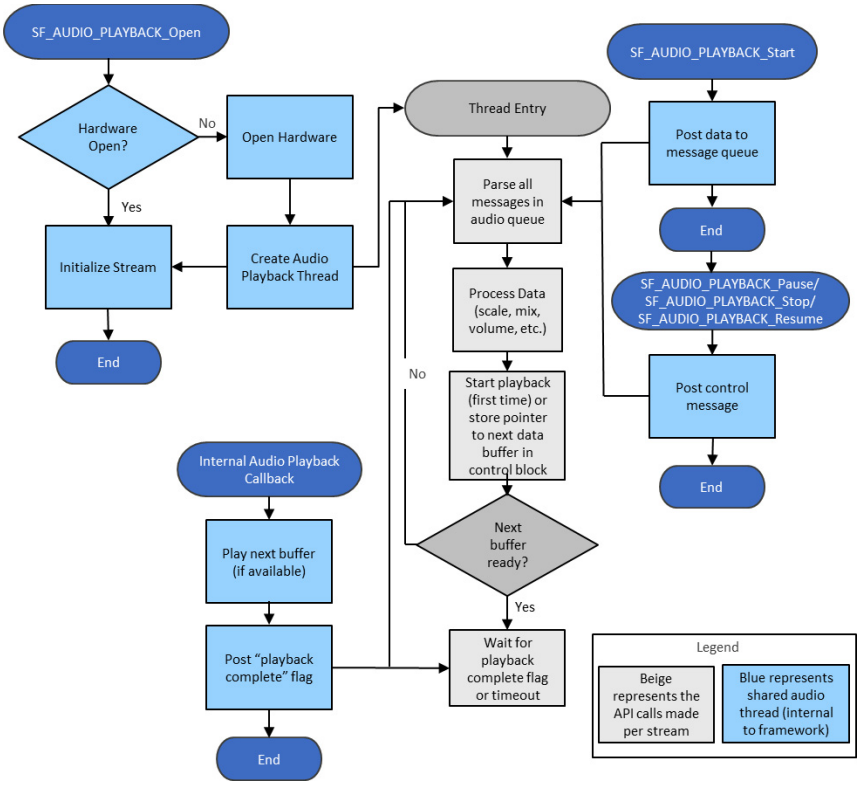


図 11:オーディオ再生 DAC フレームワークモジュールのフローチャート

*注 :DAC または DAC8 ドライバーの選択オプションは MCU 固有です。*オーディオ再生フレームワークの使用例：

- セマフォを作成します（たとえば、g_sf_audio_playback_semaphore）.。これは [スレッド] タブで行うことができます。初期値を 2 に設定します (オーディオ再生フレームワークでは、ストリームあたり最大 2 つのデータ メッセージを保持できます)。
- コールバック関数を作成します（たとえば、sf_audio_playback_callback）.。オーディオ再生フレームワーク インスタンスにコールバック関数の名前を入力します。このコールバック関数は、オーディオ再生フレームワークがデータの処理を終了したときに呼び出されます。コールバックで、上記で作成したセマフォを配置します。
- メイン ループ内で、データを再生する前にセマフォを取得します。データを再生するには、まずメッセージング フレームワークからバッファを取得し、次にオーディオ再生データ構造をバッファ内に作成します。

オーディオ再生 DAC フレームワークは、単一のハードウェアポート上の複数のオーディオストリームをサポートします。2 つのストリームを使用する場合に必要なモジュールのブロック図を下図に示します。

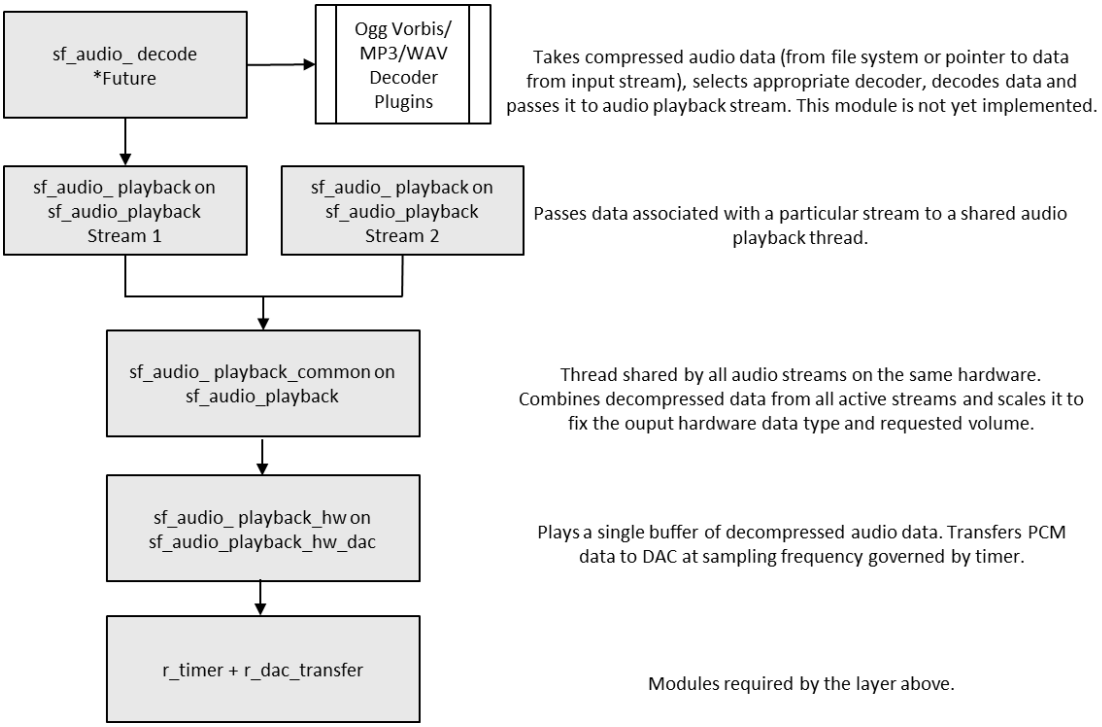


図 12:複数のオーディオストリームを実装するオーディオ再生 DAC フレームワーク

オーディオ再生 DAC フレームワークモジュールの動作に関する重要な注意事項と制限事項

- オーディオフレームワークの DAC ハードウェアポートは、転送 API を使用して、内蔵タイマで定義されたサンプリング周波数で再生バッファから DAC にオーディオデータを転送します。
- オーディオフレームワーク DAC ハードウェアポートには、タイマ、DAC/DAC8、および転送 API モジュールに対する依存関係があります。
- タイマ モジュールを追加します。
  - 周波数 (Hz 単位) をオーディオ データのサンプリング周波数に設定します。
  - DTC を転送モジュールとして使用する場合は (推奨)、割り込みを有効にします。
- 転送モジュールの追加：DTC または DMAC のいずれかを選択します。
- DTC の場合
  - DAC チャネル 0 を使用する場合は宛先ポインタを &R_DAC -> DADRn[0] に設定し、DAC チャネル 1 を使用する場合は &R_DAC -> DADRn[1] に設定します。
  - DAC8 チャネル 2（S128）を使用する場合は宛先ポインタを &R_DAC8 -> DADRn[2] に設定し、DAC8 チャネル 0（S1JA）を使用する場合は &R_DAC8 -> DADRn[0] に設定します。
  - アクティベーション ソースを上で選択したタイマ割り込みに設定します。
- DMAC の場合
  - dmac サポートを有効にします。

  - DAC チャネル 0 を使用する場合は宛先ポインタを &R_DAC -> DADRn[0] に設定し、DAC チャネル 1 を使用する場合は &R_DAC -> DADRn[1] に設定します。
  - DAC8 チャネル 2（S128）を使用する場合は宛先ポインタを &R_DAC8 -> DADRn[2] に設定し、DAC8 チャネル 0（S1JA）を使用する場合は &R_DAC8 -> DADRn[0] に設定します。
  - アクティベーション ソースを上で選択したタイマ割り込みに設定します。
- オーディオ再生 DAC フレームワークは、API への変更なしに、以下の MCU ファミリをサポートするように設計されています。
  - S7G2
  - S3A3
  - S5D9
  - S124
  - S3A6
  - S5D5
  - S3A1
  - S128
  - S1JA
  - S5D3
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.3.4 アプリケーションでのオーディオ再生 DAC フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにオーディオ再生 DAC フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

オーディオ再生 DAC フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。（オーディオ再生 DAC フレームワークモジュールのデフォルト名は g_sf_audio_playback_hw0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

オーディオ再生 DAC フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_audio_playback_hw0 Audio Playback Hardware Framework on g_sf_audio_playback_hw0	Threads	New Stack> Framework> Audio> Audio Playback Hardware Framework on g_sf_audio_playback_hw_dac

次の図に示すように、sf_audio_playback_hw_dac のオーディオ再生 DAC フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

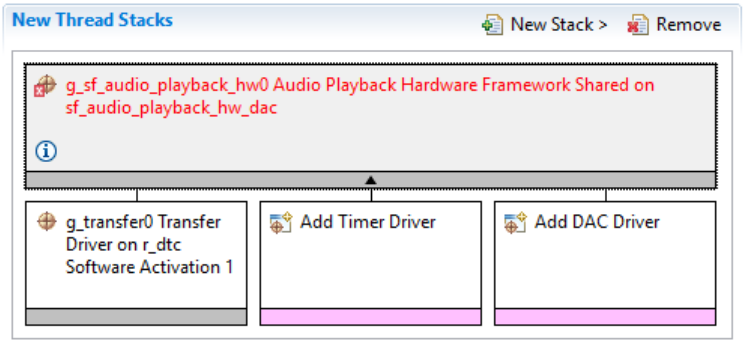


図 13:**オーディオ再生 DAC フレームワークモジュールスタック**

4.1.3.5 オーディオ再生 DAC フレームワークモジュールの構成

オーディオ再生 DAC フレームワークモジュールは、必要な動作に合わせてユーザーが構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます (ブロックが赤で強調表示される)。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成テーブルの設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_audio_playback_hw_dac のオーディオ再生 DAC フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
DMAC Support	Disabled, Enabled Default: Disabled	DMAC support selection.
Name	g_sf_audio_playback_hw0	Module name.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、ターゲットハードウェアの実装に基づいた DAC チャネルの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

オーディオ再生 DAC フレームワークのローレベルモジュールの構成

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

r_dmac Software Activation の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Name	g_transfer0	Module name
Channel	0	Channel selection
Mode	Normal	Mode selection

ISDE Property	Value	Description
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source	Software Activation, Peripheral Events Default: Software Activation	Activation source selection
Auto Enable	False	Auto enable selection
Callback	NULL	Callback selection
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dmac Software Activation 1 の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection

ISDE Property	Value	Description
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events Default: Software Activation 1	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_agt のタイマドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables parameter checking.
Name	g_timer0	Module name.
Channel	0	Physical hardware channel.
Mode	Periodic	Warning: One Shot functionality is not available in the GPT hardware, so it is implemented in software by stopping the timer in the ISR called when the period expires. For this reason, ISR's must be enabled for one-shot mode even if the callback is unused.

ISDE Property	Value	Description
Period Value	10	See Timer Period Calculation
Period Unit	Hertz	See Timer Period Calculation
Auto Start	False	Set to true to start the timer after configuring or false to leave the timer stopped until timer_api_t::start is called.
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub Default: PCLKB	The clock source for the AGT counter.
AGTO Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for AGT (AGTO pin). Set to false for no output of the timer signal.
AGTIO Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for AGT (AGTIO pin). Set to false for no output of the timer signal.
Output Inverted	True, False Default: False	Set to false to start the output signal low. Set to true to start the output signal high.
Enable comparator A output pin	True, False Default: False	Enable comparator A output pin selection
Enable comparator B output pin	True, False Default: False	Enable comparator B output pin selection

ISDE Property	Value	Description
Callback	NULL	A user callback function can be registered in timer_api_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the timer period elapses. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Timer interrupt priority. 0 is the highest priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_gpt のタイマドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_timer0	Module name.
Channel	0	Channel selection.

ISDE Property	Value	Description
Mode	Periodic	Warning: One Shot functionality is not available in the GPT hardware, so it is implemented in software by stopping the timer in the ISR called when the period expires. For this reason, ISR's must be enabled for one-shot mode even if the callback is unused.
Period Value	10	See Timer Period Calculation
Period Unit	Hertz	See Timer Period Calculation
Duty Cycle Value	50	Duty cycle value selection
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	Duty cycle unit selection
Auto Start	False	Set to true to start the timer after configuring or false to leave the timer stopped until timer_api_t::start is called.
GTIOCA Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
GTIOCB Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.

ISDE Property	Value	Description
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
Callback	NULL	A user callback function can be registered in timer_api_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the timer period elapses. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dac の DAC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_dac0	Module name.

ISDE Property	Value	Description
Channel	0	Set to 0 for output DA0 or 1 for output DA1.
Synchronize with ADC	Enabled, Disabled Default: Disabled	Set to true for anti-interference synchronization with the Analog-to-Digital Converter (ADC) Module. Set to false false if power supply interference between the analog modules is not a problem, or if asynchronous conversion by the DAC Module is desired.
Data Format	Right Justified	Set to zero, if 12-bit data values are loaded in bits 11 through 0, or right justified. Set to one, if 12-bit data values are loaded in bits 15 through 4, or left justified.
Output Amplifier	Enable, Disable Default: Disable	Set to true, if output amplifier hardware function is desired. Set to false to bypass output amplifier hardware function.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

DAC8 HAL モジュール r_dac8 の構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_dac8_0	Module name
Channel	0	Channel selection

ISDE Property	Value	Description
Synchronize with ADC	Enabled, Disabled Default: Disabled	Choose whether to sync with the ADC module
Data Format	Right Justified	Data format selection
DAC Mode	Normal Mode, Real-time (Event Link) Mode Default: Normal Mode	DAC mode selection
Charge Pump Enabled (Requires MOCO active)	Enabled, Disabled Default: Enabled	Enable or disable the charge pump

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

オーディオ再生 DAC フレームワークモジュールのクロック構成

オーディオ再生 DAC フレームワークハードウェアモジュールは、[Clocks] 構成ウィンドウにある周辺クロックを使用します。

オーディオ再生 DAC フレームワークモジュールのピン構成

DAC または SSI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は、[SSP configuration] ウィンドウ内でのピンの選択方法と、関連するピンの選択例を示します。

注 : 一部のペリフェラルでは、動作モードの選択によって使用可能なペリフェラル信号が決まり、それに従って必要な MCU ピンが決定されます。

オーディオ再生 DAC フレームワークモジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
DAC	Pins	Select Peripherals > Analog:DAC12 > DAC120/121
SSI	Pins	Select Peripherals > Connectivity:SSI > SSI/0/1

注 : 選択シーケンスでは、DAC0/DAC1 または SSI0/SSI1 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_dac の DAC HAL モジュールのピン構成設定

Pin Configuration Property	Value	Description
Operation Mode	Enabled, Disabled	Operation selection
DAC	None, P014 Default: P014	DAC pin selection

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.1.3.6 アプリケーションでのオーディオ再生 DAC フレームワークモジュールの使用

アプリケーションでオーディオ再生 DAC フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_audio_playback_hw_api_t::open API を使用してオーディオ再生 DAC フレームワークを初期化します。
2) sf_audio_playback_hw_api_t::start API を使用してオーディオ再生 DAC フレームワークのローレベルハードウェアを開始します。
3) sf_audio_playback_hw_api_t::play API を使用して、PCM オーディオサンプルを再生します。
4) ハードウェアでサポートされる PCM オーディオサンプルは sf_audio_playback_hw_api_t::dataTypeGet API によって提供されます。
5) sf_audio_playback_hw_api_t::stop API を使用して、オーディオ再生 DAC フレームワークのローレベルハードウェアを停止します。
6) sf_audio_playback_hw_api_t::close API を使用して、オーディオ再生 DAC フレームワークを閉じます。

これらの一般的な手順を、次の図の通常の動作フローに示します。

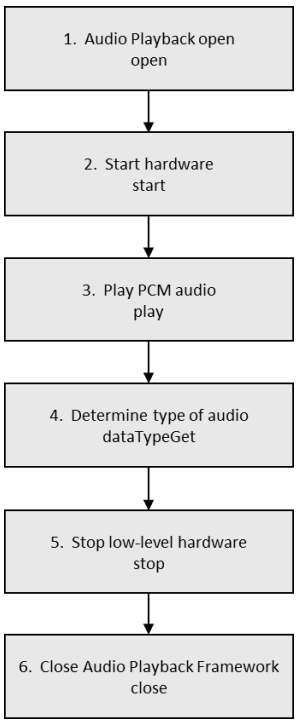


図 14:一般的なオーディオ再生 DAC フレームワークモジュールアプリケーションのフロー図

4.1.4 オーディオ再生 I2S フレームワーク

この I2S オーディオ再生フレームワークモジュールは、オーディオ再生アプリケーション用のハイレベルの API を提供し、8 ビットおよび 16 ビットパルス符号変調（PCM）サンプルを再生するために必要な同期を処理します。オーディオ再生フレームワークは、Synergy MCU の I2S、タイマ（AGT または GPT）、データ転送（DMA または DTC）ペリフェラルを使用します。ユーザー定義のコールバックを作成し、追加データが必要な場合に対応できます。

4.1.4.1 オーディオ再生 I2S フレームワークモジュールの特長

- データを扱いやすいチャンクに分割することによって長いバッファを再生します。
- ThreadX タイムアウトが発生するまで再生を繰り返します（正弦波音やループするバックグラウンドミュージックのように音声が繰り返される場合）。
- 最後のバッファの再生が開始された後、コールバックを使用して次のデータを要求します。
- ソフトウェア音量制御。
- 一時停止と再開の機能。
- 複数のストリームについての基本的なミキシング。

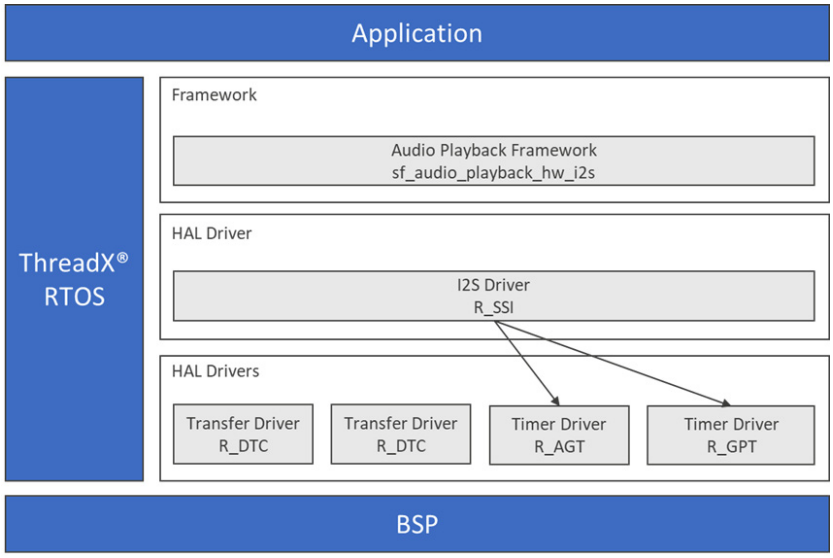


図 15:オーディオ再生 I2S フレームワークモジュールのブロック図

4.1.4.2 オーディオ再生 I2S フレームワークモジュールの API の概要

オーディオ再生 I2S フレームワークモジュールは、オープン、開始、再生、停止などの操作の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

オーディオ再生 I2S フレームワークモジュールの API 要約

Function Name	Example API Call and Description
open	g_sf_audio_playback_hw0.p_api->open(g_sf_audio_playback_hw0.p_ctrl, g_sf_audio_playback_hw0.p_cfg); Open a device channel for read/write and control.
start	g_sf_audio_playback_hw0.p_api->start(g_sf_audio_playback_hw0.p_ctrl, &p_data, Timeout); Start Audio Playback Hardware.

Function Name	Example API Call and Description
stop	g_sf_audio_playback_hw0.p_api->stop(g_sf_audio_playback_hw0.p_ctrl); Stop Audio Playback Hardware.
play	g_sf_audio_playback_hw0.p_api->play (g_sf_audio_playback_hw0.p_ctrl, p_buffer, length); Play audio buffer.
dataTypeGet	g_sf_audio_playback_hw0.p_api->dataType Get(g_sf_audio_playback_hw0.p_ctrl, p_data_type); Stores expected data type in provided pointer p_data_type.
close	g_sf_audio_playback_hw0.p_api->close(g_sf_audio_playback_hw0.p_ctrl); Close the audio module.
versionGet	g_sf_audio_playback_hw0.p_api->versionG et(&version); Return the version of the module with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successful.

Name	Description
SSP_ERR_OUT_OF_MEMORY	The number of streams open at once is limited to SF_AUDIO_PLAYBACK_CFG_MAX_STREAMS. If this number is exceeded, an out of memory error occurs.
SSP_ERR_TIMEOUT	Timeout occurred before playback finished.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.4.3 オーディオ再生 I2S フレームワークモジュールの動作の概要

I2S オーディオ再生フレームワークモジュールは、オーディオ再生をサポートするためのスレッドを内部で作成します。下図に、オーディオ再生フレームワークのスレッドとパブリックオーディオ再生フレームワーク API とのやり取りを表すフローチャートを示します。

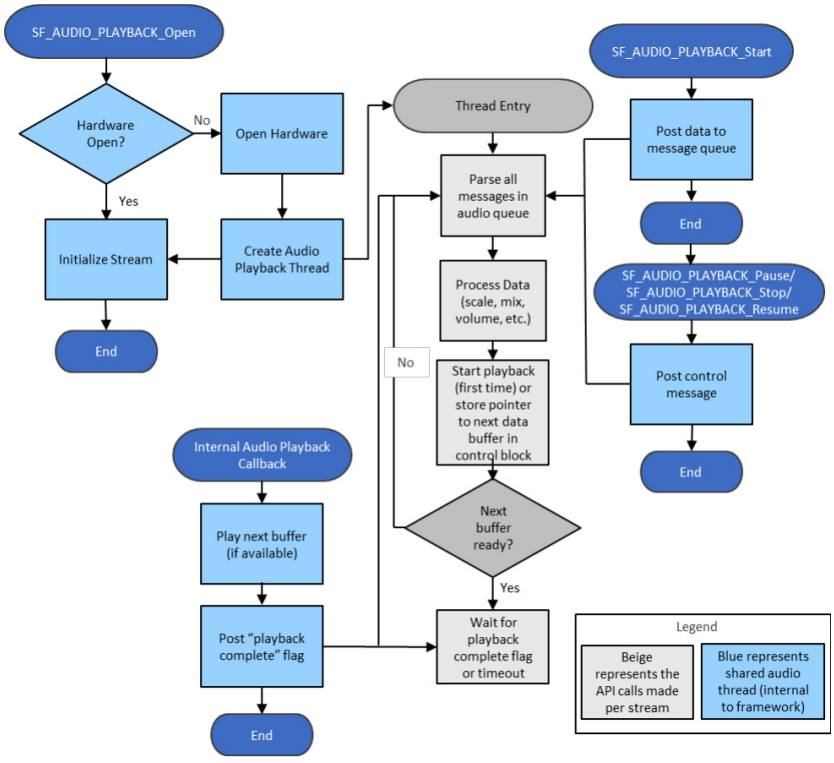


図 16:オーディオ再生 I2S フレームワークモジュールのフローチャート

オーディオ再生 I2S フレームワークの推奨される使用方法を以下に示します。

- セマフォを作成します（たとえば、g_sf_audio_playback_semaphore）.。これは [スレッド] タブで行うことができます。初期値を 2 に設定します (オーディオ再生フレームワークでは、ストリームあたり最大 2 つのデータ メッセージを保持できます)。
- コールバック関数を作成します（たとえば、sf_audio_playback_callback）.。オーディオ再生 I2S フレームワーク インスタンスにコールバック関数の名前を入力します。このコールバック関数は、オーディオ再生フレームワークがデータの処理を終了したときに呼び出されます。コールバックで、上記で作成したセマフォを配置します。
- メイン ループ内で、データを再生する前にセマフォを取得します。データを再生するには、まずメッセージング フレームワークからバッファを取得し、次にオーディオ再生データ構造をバッファ内に作成します。

オーディオ再生 I2S フレームワークは、単一のハードウェアポート上の複数のオーディオストリームをサポートします。2 つのストリームを使用する場合に必要なモジュールのブロック図を下図に示します。

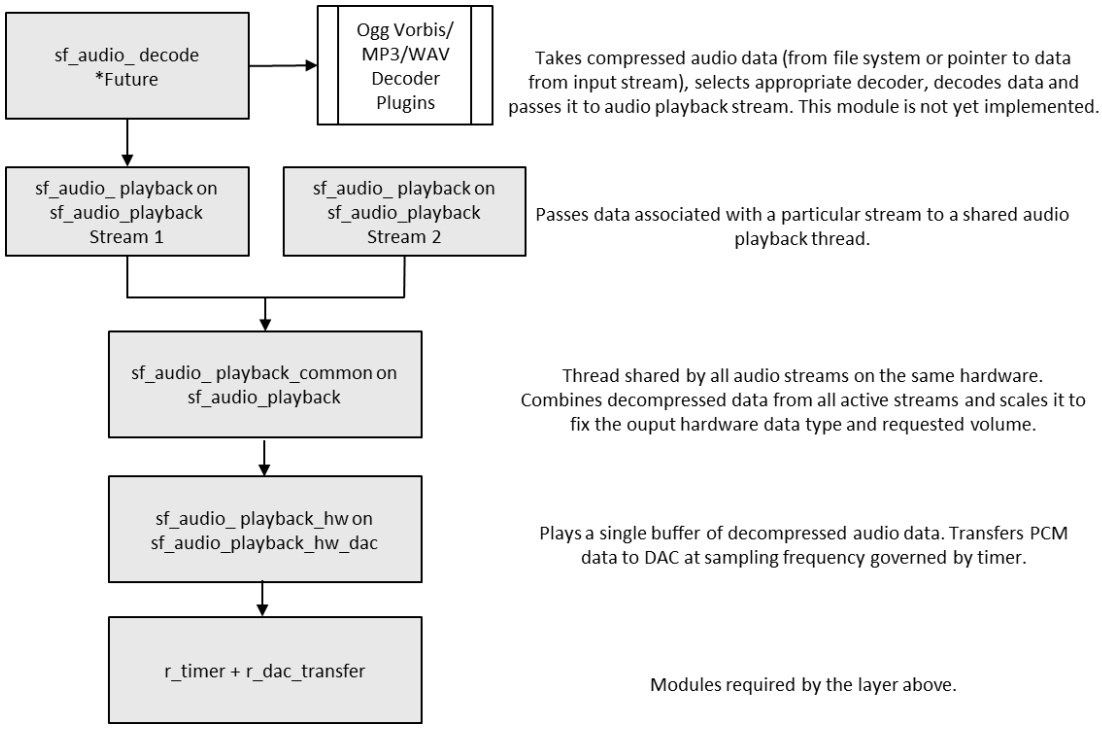


図 17:複数のオーディオストリームを実装するオーディオ再生 I2S フレームワーク

オーディオ再生 I2S フレームワークモジュールの動作に関する重要な注意事項と制限事項

- 使用されるキューは、[Audio Playback Framework Shared on sf_audio_playback] の [Properties] で指定された名前と一致する必要があります（デフォルトは g_sf_audio_playback_queue）。
- オーディオ フレームワーク I2S ハードウェア ポートには、I2S モジュールに対する依存関係があります。I2S ドライバー モジュールは DTC を使用して高速化できます (推奨)。
- I2S ドライバーモジュール。
  - オーディオ クロック周波数 (ヘルツ単位) を、使用されている入力オーディオ クロックの周波数に設定します。
  - サンプリング周波数 (ヘルツ単位) をオーディオ データのサンプリング周波数に設定します。

  - データ ビットとワード長を 16 ビットに設定します (オーディオ フレームワークでは 16 ビットのみが受け入れられます)。
  - SSIn TXI および SSIn INT 割り込みを有効にします。
- r_dtc 上の転送モジュール（推奨）。
- アクティベーション ソースを SSIn TXI 割り込みに設定します。
- オーディオ再生 I2S フレームワークは、API への変更なしに、以下の MCU ファミリをサポートするように設計されています。
  - S7G2
  - S5D9
  - S5D5
  - S3A7
  - S3A1
  - S5D3
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.4.4 アプリケーションへのオーディオ再生 I2S フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにオーディオ再生 I2S フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

オーディオ再生 I2S フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。（オーディオ再生 I2S フレームワークモジュールのデフォルト名は g_sf_audio_playback_hw0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

オーディオ再生 I2S フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_audio_playback_hw0 Audio Playback Hardware Framework on sf_audio_playback_hw_i2s	Threads	New Stack> Framework> Audio> Audio Playback Hardware Framework on sf_audio_playback_hw_i2s

次の図に示すように、sf_audio_playback_hw_i2s のオーディオ再生 I2S フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数の

スタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

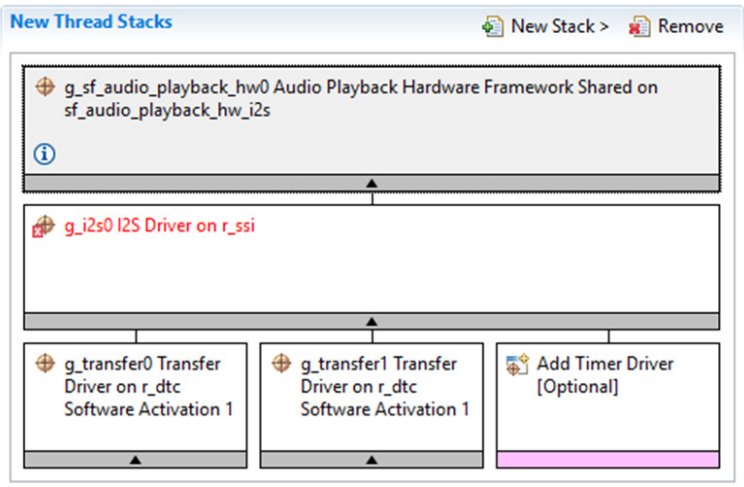


図 18:オーディオ再生 I2S フレームワークモジュールスタック

4.1.4.5 オーディオ再生 I2S フレームワークモジュールの構成

オーディオ再生 I2S フレームワークモジュールは、必要な動作に合わせてユーザーが構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます (ブロックが赤で強調表示される)。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成テーブルの設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_audio_playback_hw_i2s のオーディオ再生 I2S フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.

ISDE Property	Value	Description
Name	g_sf_audio_playback_hw0	Module name.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、ターゲットハードウェアの実装に基づいた DAC チャネルまたは I2S チャネルの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

オーディオ再生 I2S フレームワークのローレベルモジュールの構成

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

r_ssi 上の I2S HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_i2s0	Module name.
Channel	0	Physical hardware channel.
Audio Clock Frequency (Hertz)	2822400	Input audio clock frequency, used to generate the I2S clock. Must be a multiple between 1 and 128 of: (sampling_freq_hz * word_length_in_bits)
Sampling Frequecy (Hertz)	44100	Sampling frequency of audio data.
Data Bits	8 bits, 16, 18, 20, 22, 24 Default: 16 bits	Bit depth of audio data, which is the size in bits of one sample of audio data.

ISDE Property	Value	Description
Word Length	8 bits, 16, 24, 32 Default: 16 bits	Word length of audio data, must be at least the same size as the bit depth (Data Bits field).
WS Continue Mode	Enabled, Disabled Default: Disabled	Enable WS continue mode to continue to output the word select line when the peripheral is idle. Disable to stop outputting the word select line when the peripheral is idle.
Name of I2S callback function to be defined by user	NULL	A user callback function must be registered in open. The callback will be called from the interrupt service routine (ISR) when the transmission FIFO reaches the high watermark point after all data for transmission is transmitted or when reception is complete (the requested number of bytes have been received). Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Transmit interrupt priority selection
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Receive interrupt priority selection

ISDE Property	Value	Description
Idle/Error Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Idle/error interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Software Activation 1 の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection.
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Driver name.
Mode	Normal	Mode selection.
Transfer Size	4 Bytes	Transfer size selection.
Destination Address Mode	Fixed	Destination address mode selection.
Source Address Mode	Incremented	Source address mode selection.
Repeat Area (Unused in Normal Mode	Source	Repeat area selection.
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection.

ISDE Property	Value	Description
Destination Pointer	NULL	Destination pointer selection.
Source Pointer	NULL	Source pointer selection.
Number of Transfers	0	Number of transfers selection.
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection.
Activation Source (Must enable IRQ)	Software Activation 1, Software Activiation 2, Peripheral Events Default: Software Activation 1	Activation source selection.
Auto Enable	False	Auto enable selection.
Callback (Only valid with Software start)	NULL	Callback selection.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Software Activation 1 の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection.

ISDE Property	Value	Description
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer1	Driver name.
Mode	Normal	Mode selection.
Transfer Size	4 Bytes	Transfer size selection.
Destination Address Mode	Incremented	Destination address mode selection.
Source Address Mode	Fixed	Source address mode selection.
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection.
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection.
Destination Pointer	NULL	Destination pointer selection.
Source Pointer	NULL	Source pointer selection.
Number of Transfers	0	Number of transfers selection.
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection.
Activation Source (Must enable IRQ)	Software Activation 1, Software Activiation 2, Peripheral Events Default: Software Activation 1	Activation source selection.
Auto Enable	False	Auto enable selection.
Callback (Only valid with Software start)	NULL	Callback selection.

ISDE Property	Value	Description
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_agt のタイマドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection.
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer1	Driver name.
Mode	Normal	Mode selection.
Transfer Size	4 Bytes	Transfer size selection.
Destination Address Mode	Incremented	Destination address mode selection.
Source Address Mode	Fixed	Source address mode selection.
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection.
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection.

ISDE Property	Value	Description
Destination Pointer	NULL	Destination pointer selection.
Source Pointer	NULL	Source pointer selection.
Number of Transfers	0	Number of transfers selection.
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection.
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events Default: Software Activation 1	Activation source selection.
Auto Enable	False	Auto enable selection.
Callback (Only valid with Software start)	NULL	Callback selection.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_gpt のタイマドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection
Name	g_timer0	Module name
Channel	0	Channel selection

ISDE Property	Value	Description
Mode	Periodic	Mode selection
Period Value	2822400 *2	Period value selection
Period Unit	Hertz	Period unit selection
Duty Cycle Value	50	Duty cycle value selection
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	Duty cycle unit selection
Auto Start	FALSE	Auto start selection
GTIOCA Output Enabled	True, False Default: False	GTIOCA output enabled selection
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCA stop level selection
GTIOCB Output Enabled	True, False Default: False	GTIOCB output enabled selection
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCB stop level selection
Callback	NULL	Callback selection
Overflow Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

オーディオ再生 I2S フレームワークモジュールのクロック構成

オーディオ再生 I2S フレームワークハードウェアモジュールは、[Clock configuration] ウィンドウにあるペリフェラルクロックを使用します。

オーディオ再生 I2S フレームワークモジュールのピン構成

SSI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は、[SSP configuration] ウィンドウ内でのピンの選択方法と、関連するピンの選択例を示します。

オーディオ再生 I2S フレームワークモジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
I2S	Pins	Select Peripherals > Connectivity:SSI > SSI0/SSI1

注 : 選択シーケンスでは、ADC0/ADC1 または SSI0/SSI1 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_ssi の I2S ドライバーのピン構成設定

Pin Configuration Property	Value	Description
Pin Group Selection	_A only, _B only, Mixed	Pin group for I2S port
Operation Mode	Enabled, Custom, Disabled	Operation selection
SSISCK	None, P204 Default: None	SSI Serial Clock
SSIWS	None, P205 Default: None	SSI Stereo pin selection
SSIDATA	None, P206 Default: None	SSI Data pin selection

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.1.4.6 アプリケーションでのオーディオ再生 I2S フレームワークモジュールの使用

アプリケーションでオーディオ再生 I2S フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_audio_playback_hw_api_t::open API を使用してオーディオ再生 I2S フレームワークを初期化します。
2) sf_audio_playback_hw_api_t::start API を使用してオーディオ再生 I2S フレームワークのローレベルハードウェアを開始します。
3) sf_audio_playback_hw_api_t::play API を使用して、PCM オーディオサンプルを再生します。
4) ハードウェアでサポートされる PCM オーディオサンプルは sf_audio_playback_hw_api_t::dataTypeGet API によって提供されます。
5) sf_audio_playback_hw_api_t::stop API を使用して、オーディオ再生 I2S フレームワークのローレベルハードウェアを停止します。
6) sf_audio_playback_hw_api_t::close API を使用して、オーディオ再生 I2S フレームワークを閉じます。

これらの一般的な手順を、次の図の通常の動作フローに示します。

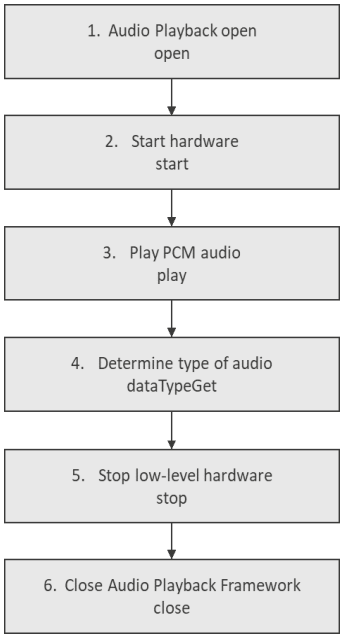


図 19:一般的なオーディオ再生 I2S フレームワークモジュールアプリケーションのフロー図

4.1.5 オーディオ記録 ADC フレームワーク

オーディオ記録 ADC フレームワークモジュールは、記録アプリケーション用のハイレベル API を提供し、Synergy MCU の sf_adc_periodic およびその下位レイヤーの ADC、GPT および DTC ペリフェラルを使用します。ユーザー定義のコールバックを作成し、サンプルカウントが完了したことを通知できます。

4.1.5.1 オーディオ記録 ADC フレームワークモジュールの特長

- 8 ビットまたは 12 ビット PCM でデータを記録します
- ADC 周期フレームワークを使用して構成と統合を単純化します
- ThreadX オブジェクト（ミューテックスなど）を使用して、不適切なアクセスからハードウェアを保護します
- 高レベル関数の API によってコーディングが簡略化されます
  - sf_audio_record_api_t::open, sf_audio_record_api_t::start
  - sf_audio_record_api_t::stop, sf_audio_record_api_t::infoGet
  - sf_audio_record_api_t::close

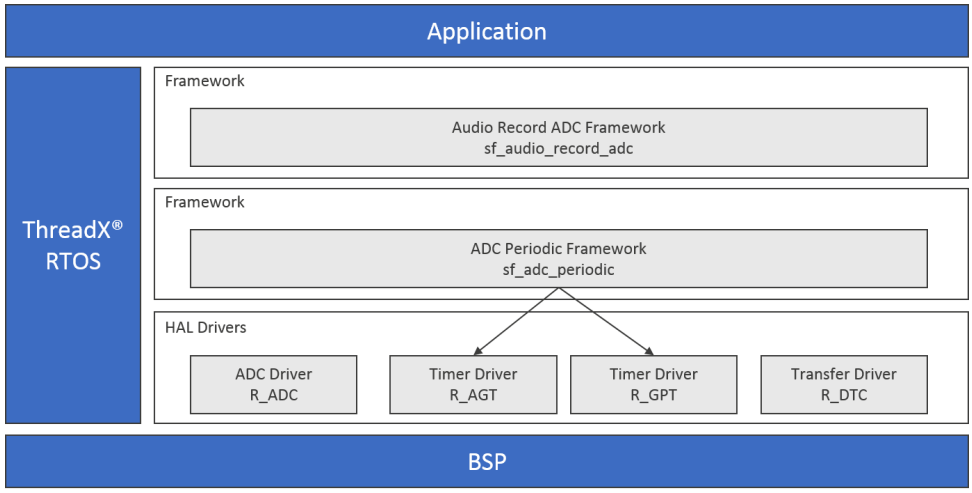


図 20: オーディオ記録 ADC フレームワークモジュールのブロック図

4.1.5.2 オーディオ記録 ADC フレームワークモジュールの API の概要

オーディオ記録 ADC フレームワークモジュールは、記録プロセスのオープン、クローズ、開始、停止などの API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

オーディオ記録 ADC フレームワークモジュールの API 要約

Function Name	Example API Call and Description
open	g_sf_audio_record_adc.p_api->open(g_sf_audio_record_adc.p_ctrl, g_sf_audio_record_adc.p_cfg); Initialize the module.
start	g_sf_audio_record_adc.p_api->start(g_sf_audio_record_adc.p_ctrl); Start audio recording.
stop	g_sf_audio_record_adc.p_api->stop(g_sf_audio_record_adc.p_ctrl); Stop audio recording.
infoGet	g_sf_audio_record_adc.p_api->infoGet(g_sf_audio_record_adc.p_api.p_ctrl); Get the channel information (mono or Stereo).
close	g_sf_audio_record_adc.p_api->close(g_sf_audio_record_adc.p_ctrl); Close the module.
versionGet	g_sf_audio_record_adc.p_api->versionGet(&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_ARGUME	Parameter has invalid value.
SSP_ERR_IN_USE	The adc periodic framework mutex may be unavailable for the unit requested. See HAL driver for other possible causes.
SSP_ERR_INTERNAL	An internal ThreadX error has occurred. This is typically a failure to create/use a mutex or to create an internal thread.
SSP_ERR_NOT_OPEN	Unit is not open
SSP_ERR_ASSERTION	The parameter p_ctrl or p_sample is NULL.
SSP_ERR_UNSUPPORTED	This function is not supported by the HAL driver (p_ctrl->p_api->close is NULL).

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.5.3 オーディオ記録 ADC フレームワークモジュールの動作の概要

オーディオ記録 ADC フレームワークモジュールは、ADC 周期フレームワークを使用してオーディオアナログデータをサンプリングします。キャプチャされたデータサンプルはユーザーバッファに保存されます。このデータは、アプリケーションの必要に応じてさらに処理することができます。オーディオレコード ADC フレームワークには、フレームワークの初期化中に初期化される構成パラメータが存在します。フレームワークの初期化では他にも、データキャプチャ用に基礎となる ADC 周期フレームワークが初期化されます。

取得されたデータはユーザー定義バッファに格納されますが、これは次に示すようにコールバック関数内で実行されます。

コールバックの名前が sf_audio_record_user_callback: と構成されているとします。

```
uint16_t * audio_record_buffer;

void sf_audio_record_user_callback (sf_audio_record_callback_args_t *p_args)

{

audio_record_buffer = ((uint16_t *)g_sf_audio_record_adc.p_cfg->

p_capture_data_buffer + (p_args->buffer_index/2)); }
```

オーディオ記録 ADC フレームワークモジュールの動作に関する重要な注意事項と制限事項

- オーディオ記録 ADC フレームワークモジュールの構成データで、データバッファの長さ、データ幅、サンプリングレート、サンプリング繰り返し回数を指定できます。

- 現在のオーディオ記録 ADC は、ローレベルフレームワークとして ADC 周期フレームワークのみをサポートしているため、I2S を使用した記録には対応していません。
- 現状では、8 ビットまたは 12 ビットの PCM データの記録をサポートしています。
- 現在、オーディオ記録 ADC ではモノチャネルしかサポートされません。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.5.4 アプリケーションへのオーディオ記録 ADC フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにオーディオ記録 ADC フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

オーディオ記録 ADC フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(ADC オーディオ記録 ADC フレームワークのデフォルト名は g_adc_record_adc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

オーディオ記録 ADC フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_audio_record_adc0 Audio Record ADC Framework on sf_audio_record_adc	Threads	New Stack> Driver> Audio> Audio Record ADC Framework on sf_audio_record_adc

次の図に示すように、sf_audio_record_adc のオーディオ記録 ADC フレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

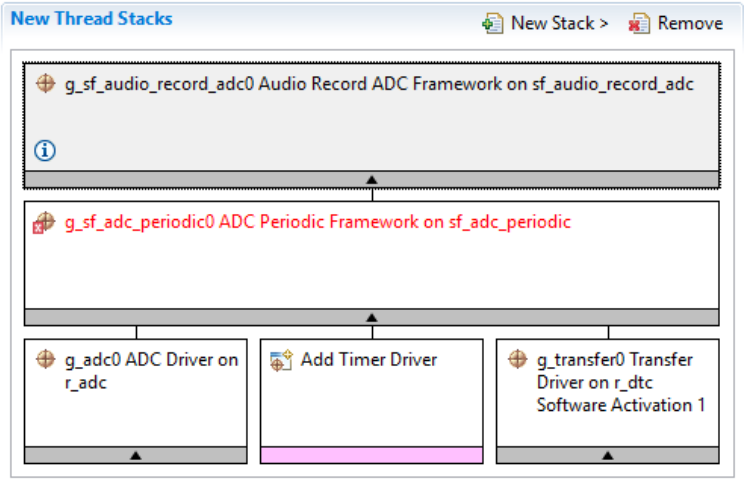


図 21:オーディオ記録 ADC フレームワークモジュールのスタック

4.1.5.5 オーディオ記録 ADC フレームワークモジュールの構成

オーディオ記録 ADC フレームワークモジュールは、必要な動作に合わせてユーザーが構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_audio_record_adc でのオーディオ記録 ADC フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_sf_audio_record_adc0	Module name.
Name of the data-buffer to store samples	p_capture_data_buffer	Name of the 16-bit data buffer to store samples.

ISDE Property	Value	Description
Length of the data-buffer	2048	Length of the buffer to which data is to be stored.
Audio Record Data Size	8-Bit, 16-Bit Default: 8-Bit	The data width of captured data 8 bit or 16 bit.
Sampling Rate in HZ	8000	Sampling rate to be used to capture data.
Number of sampling iterations	256	Samples to be captured.
Callback	g_audio_record_framework_user_callback	Callback to user after capturing the sample count.
Name of generated initialization function	sf_audio_record_adc_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

オーディオ記録 ADC フレームワークモジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

sf_adc_periodic の ADC 周期フレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.

ISDE Property	Value	Description
Name	g_sf_adc_periodic0	Module name.
Name of the data-buffer to store samples	g_user_buffer	Name of the 16-bit data buffer to store samples.
Length of the data-buffer	2048	Length of the buffer to which data is to be stored.
Number of sampling iterations	256	Priority of ADC Periodic Framework internal thread.
Callback	NULL	User function that will be called once "sample_counts" number of data has been buffered.
Name of generated initialization function	sf_adc_periodic_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_adc の ADC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: Enabled	If selected code for parameter checking is included in the build.
Name	g_adc0	Module name
Unit	0, 1 (S7G2 Only) Default: 0	Specify the ADC Unit to be used. The S7G2 has two units; 0 and 1.

ISDE Property	Value	Description
Resolution	14-Bit (S3A7/S124 Only), 12-Bit, 10-Bit (S7G2) Default: 8-Bit (S7G2 Only)	Specify the conversion resolution for this unit.
Alignment	Right, Left Default: Right	Specify the conversion result alignment.
Clear after read	Off, On Default: On	Specify if the result register must be automatically cleared after the conversion result is read. Note: If this is enabled, then watching the result register using a debugger always results in a 0.
Mode	Single Scan	The ADC Framework preconfigures and locks this field.
Channels 0-6	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 7-10 (S3A7/S124 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.

ISDE Property	Value	Description
Channels 11-15 (S3A7 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 16-20	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channel 21 (Unit 0 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channel 22 (S3A7/S124 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.

ISDE Property	Value	Description
Channels 23-27 (S3A7 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Temperature Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	Temperature sensor use selection for Channel Scan Mask
Voltage Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	Voltage sensor use selection for Channel Scan Mask
Normal/Group A Trigger	ELC Event	The ADC Framework preconfigures and locks this field.
Group B Trigger (Valid Only in Group Scan Mode)	ELC Event (The only valid trigger for either group in Group Scan Mode)	The ADC Framework preconfigures and locks this field.
Group Priority (Valid only in Group Scan Mode)	Group A cannot interrupt Group B, Group A can interrupt Group B; Group B scan restarts at next trigger, Group A can interrupt Group B; Group B scan restarts immediately, Group A can interrupt Group B; Group B scan restarts immediately and scans continuously Default: Group A cannot interrupt Group B	Do not use with ADC Framework since the mode is locked to Single Scan Mode.

ISDE Property	Value	Description
Add/Average Count	Disabled, Add two samples, Add three samples, Add four samples, Add sixteen samples, Average two samples, Average four samples Default: Disabled	Specify if addition or averaging needs to be done for any of the channels in this unit. The actual channels are specified by using a channel mask adc_channel_cfg_t::add_mask.
Channels 0-27	Disabled, Enabled Default: Disabled	This field is valid only if adc_cfg_t::add_average_count is enabled. This field determines what channels results are to be averaged or summed.
Temperature Sensor	Disabled, Enabled Default: Disabled	Temperature sensor use selection for Addition/Averaging Mask
Voltage Sensor	Disabled, Enabled Default: Disabled	Voltage sensor use selection for Addition/Averaging Mask
Channels 0-2	Disabled, Enabled Default: Disabled	Determines which of channels 0, 1 and 2 are using the updated sample-and-hold states value specified in adc_channel_cfg_t::sample_hold _states. This field must only be set if it is desired to modify the default sample and hold count value for channels 0, 1 and 2.

ISDE Property	Value	Description
Sample Hold States (Applies only to the 3 channels selected above)	24	Specifies the updated sampleand-hold count for the channel dedicated sample-and-hold circuit. This field is valid only if adc_channel_cfg_t::sample _hold _mask is not 0. Only channels 0, 1 and 2 have dedicated sample and hold circuits. Note: Use this to modify the default number of states (24) for which the value is sampled. Each state is equal to 1/ADCLK time.
Callback	NULL	The ADC Framework uses the callback internally.
Scan End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Scan End Interrupt Priority selection
Scan End Group B Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Scan End Group B Interrupt Priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_agt 上の AGT HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables parameter checking.

ISDE Property	Value	Description
Name	g_timer0	Module name.
Channel	0	Physical hardware channel.
Mode	Periodic	Warning: One Shot functionality is not available in the GPT hardware, so it is implemented in software by stopping the timer in the ISR called when the period expires. For this reason, ISR's must be enabled for one-shot mode even if the callback is unused.
Period Value	10	See Timer Period Calculation
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz Default: Microseconds	See Timer Period Calculation
Auto Start	False	Set to true to start the timer after configuring or false to leave the timer stopped until timer_api_t::start is called.
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub Default: PCLKB	The clock source for the AGT counter.
AGTO Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for AGT (AGTO pin). Set to false for no output of the timer signal.
AGTIO Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for AGT (AGTIO pin). Set to false for no output of the timer signal.

ISDE Property	Value	Description
Output Inverted	True, False Default: True	Set to false to start the output signal low. Set to true to start the output signal high.
Enable comparator A output pin	True, False Default: False	Enable comparator A output pin selection
Enable comparator B output pin	True, False Default: False	Enable comparator B output pin selection
Callback	NULL	A user callback function can be registered in timer_api_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the timer period elapses. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Underflow Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Timer interrupt priority. 0 is the highest priority.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_gpt 上の GPT HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_timer0	Module name.
Channel	0	The ADC Framework preconfigures and locks this field based on channel selected in the ADC Framework.
Mode	Periodic	The ADC Framework preconfigures and locks this field.
Period Value	10	Configure timer period to trigger ADC scans.
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz Default: Milliseconds	Configure units of the timer period set above.
Duty Cycle Value	50	Duty cycle value selection
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	Duty cycle unit selection
Auto Start	False	The ADC Framework preconfigures and locks this field.
GTIOCA Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.

ISDE Property	Value	Description
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
GTIOCB Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
Callback	NULL	The ADC Framework preconfigures and locks this field.
Overflow Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Software Activation 上の DTC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection

ISDE Property	Value	Description
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Block	Mode selection
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	1	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	1	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation 1	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

オーディオ記録 ADC フレームワークモジュールのクロック構成

ADC ペリフェラルモジュールは PCLKC をそのクロックソースとして使用します。

オーディオ記録 ADC フレームワークモジュールのピン構成

ADC 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。ADC ピンはアナログピンとして構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

注 : 一部のペリフェラルでは、動作モードの選択によって使用可能なペリフェラル信号が決まり、それに従って必要な MCU ピンが決定されます。

sf_audio_record_adc のオーディオ記録 ADC フレームワークモジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
ADC	Pins	Select Peripherals > Analog:ADC > ADC0

注 : この選択シーケンスでは、ADC0 がドライバーに必要なハードウェアターゲットであることを想定しています。

sf_audio_record_adc のオーディオ記録 ADC フレームワークモジュールのピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom Default: Custom	Select operating mode for ADC
ADTRG	None, P407, P102 Default: None	ADTRG Pin
AN00-19	None, Pnnn, Pmmm Default: None	Analog input pins
PGAVSS0	None, P003 Default: None	PGAVSS pin

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

4.1.5.6 アプリケーションでのオーディオ記録 ADC フレームワークモジュールの使用

一般的なアプリケーションの sf_audio_record_adc でオーディオ記録 ADC フレームワークモジュールを使用する手順は以下のとおりです。

1) sf_audio_record_api_t::open API を使用してモジュールを開きます。
2) sf_audio_record_api_t::start API を使用して記録を開始します。
3) コールバックを使用してユーザーバッファにデータを保存します。
4) 必要に応じてデータを処理します。
5) sf_audio_record_api_t::close API を使用して、モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

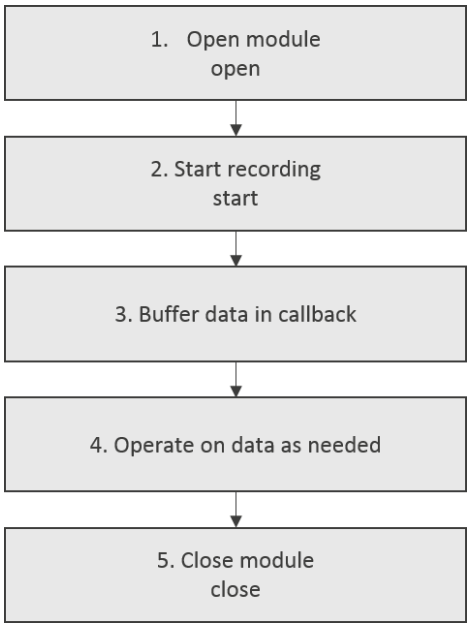


図 22:**通常のオーディオ記録 ADC フレームワークモジュールアプリケーションのフロー図**

4.1.6 オーディオ記録 I2S フレームワーク

オーディオ記録 I2S フレームワークモジュールは、オーディオ記録アプリケーション向けのハイレベル API を提供し、I2S インタフェースを使用します。オーディオ記録 I2S フレームワークモジュールは、Synergy MCU 上の SSI、GPT、DTC 周辺機能を使用します。ユーザー定義のコールバックを作成し、新しいサンプルがユーザーバッファに格納されていることを通知できます。

4.1.6.1 オーディオ記録 I2S フレームワークモジュールの特長

- スレッドセーフ

- 8 ビットまたは 16 ビット PCM でデータを記録します
- 新しいサンプルが使用可能になると呼び出される周期的コールバック関数
- 構成可能なコールバックあたりのサンプル数（サンプルカウント）

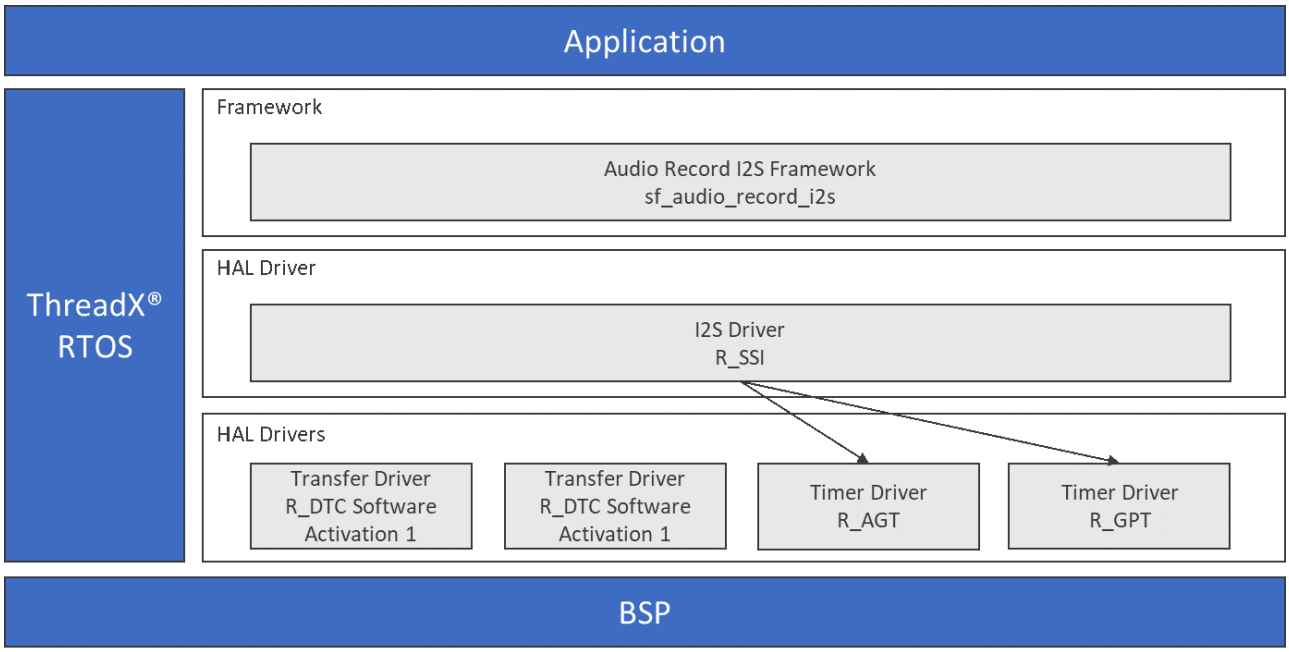


図 23: **オーディオ記録 I2S フレームワークモジュールのブロック図**

4.1.6.2 オーディオ記録 I2S フレームワークモジュールの API の概要

オーディオ記録 I2S フレームワークモジュールは、記録モジュールのオープン、開始、停止、クローズなどの API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

オーディオ記録 I2S フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_audio_record_i2s0.p_api->open (g_sf_audio_record_i2s0.p_ctrl, g_sf_audio_record_i2s0.p_cfg); Initializes audio recording framework.

Function Name	Example API Call and Description
start	g_sf_audio_record_i2s0.p_api->start (g_sf_audio_record_i2s0.p_ctrl); Starts audio recording.
stop	g_ sf_audio_record_i2s0.p_api->stop (g_sf_audio_record_i2s0.p_ctrl); Stops audio recording.
infoGet	g_sf_audio_record_i2s0.p_api->infoGet (g_sf_audio_record_i2s0.p_ctrl, p_info); Gets channel information (Mono/Stereo).
close	g_sf_audio_record_i2s0.p_api->close (g_sf_audio_record_i2s0.p_ctrl); Releases channel *mutex* and closes channel at HAL layer.
versionGet	g_ sf_audio_record_i2s0.p_api->versionGet(&version); Gets version and stores it in provided version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_INTERNAL	An internal TheadX error has occurred.

Name	Description
SSP_ERR_NOT_OPEN	Unit is not open
SSP_ERR_ASSERTION	The parameter p_ctrl or p_sample is NULL.
SSP_ERR_IN_USE	Peripheral is still running in another mode; perform Close first.
SSP_ERR_UNSUPPORTED	Command not supported.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.6.3 オーディオ記録 I2S フレームワークモジュールの動作の概要

オーディオ記録 I2S フレームワークモジュールは I2S HAL レイヤーをオーディオデータ転送の基盤となるインタフェースとして使用します。キャプチャされたデータはユーザーバッファに保存されます。このデータは、アプリケーションの必要に応じてさらに処理することができます。

取得されたデータはユーザー定義バッファに格納されますが、これは次に示すようにコールバック関数内で実行されます。

コールバックの名前が sf_audio_record_user_callback. と構成されているとします。

```
uint16_t * audio_buffer;

void audio_record_user_callback (sf_audio_record_callback_args_t * p_args)

{

    audio_buffer = ((uint16_t *)sf_audio_record_i2s.p_cfg->p_capture_data_buffer

                    + (p_args->buffer_index));

}
```

オーディオ記録 I2S フレームワークモジュールの動作に関する重要な注意事項と制限事項

オーディオ記録 I2S フレームワークモジュールの構成データで、データバッファの名前、データバッファの長さ、データサイズ(8 ビットまたは 16 ビットサンプル)、サンプリング繰り返し回数、およびコールバックの名前を指定できます。

- 現状では、このフレームワークは 8 ビットまたは 16 ビットの PCM データの記録をサポートしています。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.6.4 アプリケーションでのオーディオ記録 I2S フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにオーディオ記録 I2S フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユー

ザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。オーディオ記録 I2S フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。(オーディオ記録 I2S フレームワークモジュールのデフォルト名は g_sf_audio_record_i2s0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

オーディオ記録 I2S フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_audio_record_i2s0 Audio Record I2S Framework on sf_audio_record_i2s	Threads	New Stack> Framework> Audio> Audio Record I2S Framework on sf_audio_record_i2s

次の図に示すように、sf_audio_record_i2s のオーディオ記録 I2S フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

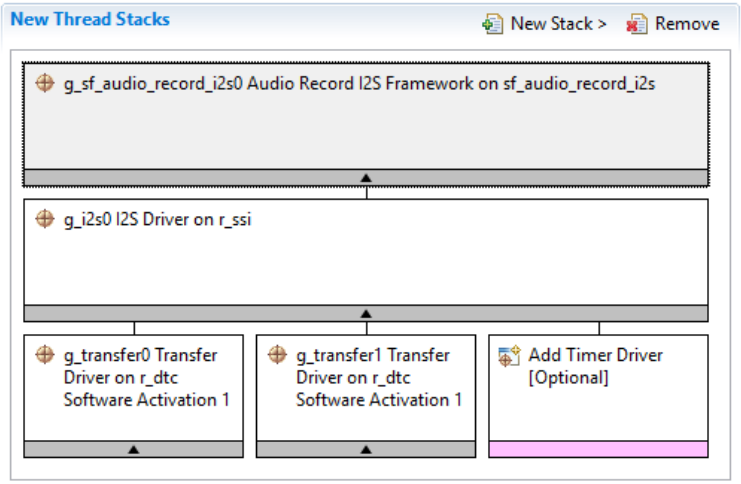


図 24:オーディオ記録 I2S フレームワークモジュールのスタック

4.1.6.5 オーディオ記録 I2S フレームワークモジュールの構成

オーディオ記録 I2S フレームワークモジュールは、必要な動作に合わせてユーザーが構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます (ブロックが赤で強調

表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成テーブルの設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_audio_record_i2s のオーディオ記録 I2S フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_sf_audio_record_i2s0	Module name.
Name of the data-buffer to store samples	p_capture_data_buffer	Data-buffer name.
Length of the data buffer	2048	Length of the data buffer
Audio Record Data Size	8-Bit, 16-Bit Default: 16-Bit	Audio record data size selection
Number of sampling iterations	256	Number of sampling iterations
Callback	g_audio_record_framework_user_callback	Callback name.
Name of generated initialization function	sf_audio_record_i2s_init0	Name of generated initialization function
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

オーディオ記録 I2S フレームワークのローレベルモジュールの構成

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

r_ssi 上の I2S HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_i2s0	Module name.
Channel	0	Physical hardware channel.
Audio Clock Frequency (Hertz)	2822400	Input audio clock frequency, used to generate the I2S clock. Must be a multiple between 1 and 128 of: (sampling_freq_hz * word_length_in_bits)
Sampling Frequency (Hertz)	44100	Sampling frequency of audio data.
Data Bits	8 bits, 16, 18, 20, 22, 24 Default: 16 bits	Bit depth of audio data, which is the size in bits of one sample of audio data.
Word Length	8 bits, 16, 24, 32 Default: 16 bits	Word length of audio data, must be at least the same size as the bit depth (Data Bits field).

ISDE Property	Value	Description
WS Continue Mode	Enabled, Disabled Default: Disabled	Enable WS continue mode to continue to output the word select line when the peripheral is idle. Disable to stop outputting the word select line when the peripheral is idle.
Audio Clock Frequency (Hertz)	External, GTIOC1A Default: External	Select External for external signal to AUDIO_CLK input pin or GTIOCA1.
Name of I2S callback function to be defined by user	NULL	A user callback function must be registered in open. The callback will be called from the interrupt service routine (ISR) when the transmission FIFO reaches the high watermark point after all data for transmission is transmitted or when reception is complete (the requested number of bytes have been received). Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Transmit interrupt priority selection
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Receive interrupt priority selection

ISDE Property	Value	Description
Idle/Error Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Idle/error interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Software Activation 1 の DTC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection.
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Driver name.
Mode	Normal	Mode selection.
Transfer Size	4 Bytes	Transfer size selection.
Destination Address Mode	Fixed	Destination address mode selection.
Source Address Mode	Incremented	Source address mode selection.
Repeat Area (Unused in Normal Mode	Source	Repeat area selection.
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection.

ISDE Property	Value	Description
Destination Pointer	NULL	Destination pointer selection.
Source Pointer	NULL	Source pointer selection.
Number of Transfers	0	Number of transfers selection.
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection.
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events Default: Software Activation 1	Activation source selection.
Auto Enable	FALSE	Auto enable selection.
Callback (Only valid with Software start)	NULL	Callback selection.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Software Activation 1 の DTC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection.

ISDE Property	Value	Description
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer1	Driver name.
Mode	Normal	Mode selection.
Transfer Size	4 Bytes	Transfer size selection.
Destination Address Mode	Incremented	Destination address mode selection.
Source Address Mode	Fixed	Source address mode selection.
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection.
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection.
Destination Pointer	NULL	Destination pointer selection.
Source Pointer	NULL	Source pointer selection.
Number of Transfers	0	Number of transfers selection.
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection.
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events Default: Software Activation 1	Activation source selection.
Auto Enable	FALSE	Auto enable selection.
Callback (Only valid with Software start)	NULL	Callback selection.

ISDE Property	Value	Description
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_agt 上の AGT HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection
Name	g_timer0	Module name
Channel	0	Channel selection
Mode	Periodic	Mode selection
Period Value	2822400 * 2	Period value selection
Period Unit	Hertz	Period unit selection
Auto Start	FALSE	Auto start selection
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub Default: PCLKB	Count source selection
AGTO Output Enabled	True, False Default: False	AGTO output selection

ISDE Property	Value	Description
AGTIO Output Enabled	True, False Default: False	AGTIO output selection
Output Inverted	True, False Default: False	Output inverted selection
Enable comparator A output pin	True, False Default: False	Enable comparator A output pin selection
Enable comparator B output pin	True, False Default: False	Enable comparator B output pin selection
Callback	NULL	Callback selection
Underflow Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_gpt 上の GPT HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection
Name	g_timer0	Module name
Channel	0	Channel selection

ISDE Property	Value	Description
Mode	Periodic	Mode selection
Period Value	2822400 *2	Period value selection
Period Unit	Hertz	Period unit selection
Duty Cycle Value	50	Duty cycle value selection
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	Duty cycle unit selection
Auto Start	FALSE	Auto start selection
GTIOCA Output Enabled	True, False Default: False	GTIOCA output enabled selection
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCA stop level selection
GTIOCB Output Enabled	True, False Default: False	GTIOCB output enabled selection
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCB stop level selection
Callback	NULL	Callback selection
Overflow Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

オーディオ記録 I2S フレームワークモジュールのクロック構成

オーディオ記録 I2S モジュールは、クロックソースとして PCLKB を使用します。

実行時にクロック周波数を変更するには、CGC インタフェースを使用します。

オーディオ記録 I2S フレームワークモジュールのピン構成

オーディオ記録 I2S フレームワークモジュールを使用するには、ペリフェラルの入力と出力のポートピンを ISDE のピンコンフィギュレータで設定する必要があります。次の表では、ISDE [Configuration] ウィンドウでのピンの選択方法を示します。

オーディオ記録 I2S フレームワークモジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SSI	Pins	Select Peripherals > Peripherals>Connectivity:SSI>SSI/SSI0/SSI1

4.1.6.6 アプリケーションでのオーディオ記録 I2S フレームワークモジュールの使用

アプリケーションでオーディオ記録 I2S フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_audio_record_api_t::open API を使用して、モジュールを初期化します。
2) sf_audio_record_api_t::start API を使用して記録を開始します。
3) 周期コールバック関数からのデータを必要に応じて操作します。
4) sf_audio_record_api_t::close API を使用して、モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フローに示します。

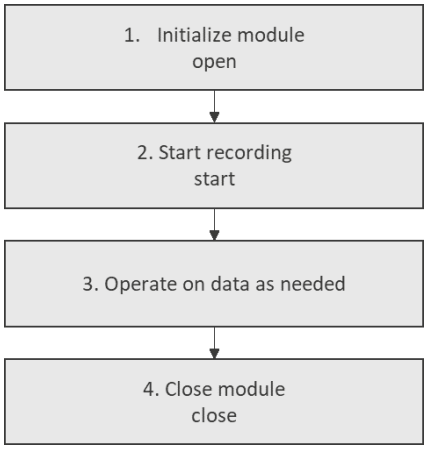


図 25:一般的なオーディオ記録 I2S フレームワークモジュールアプリケーションのフロー図

4.1.7 sf_block_media_lx_nor のブロックメディアフレームワーク

sf_block_media_lx_nor フレームワークモジュールのブロックメディアフレームワークは、QSPI NOR フラッシュメモリデバイスとのインタフェースを可能にするためにハイレベルの API を提供します。QSPI NOR フラッシュメモリに対するデータのリード、ライト、制御のための API 関数を提供します。このフレームワークには、Express Logic レベルウェアリングの X-Ware コンポーネントの LevelX NOR が組み込まれています。ウェアレベリングをサポートする LevelX 関数は、自動化されており、開発者に対してサポートします。Express Logic FileX システムを使用するファイルシステムのアクセスもサポートされているため、ファイルシステムベースのアプリケーションの実装が簡単になります。

4.1.7.1 ブロックメディアフレームワークモジュールの特長

- NOR フラッシュメモリデバイス用のブロックメディアフレームワークインタフェースをサポートします。
- NOR フラッシュメモリのファイルシステムアクセスをサポートします。
- LevelX ウェアレベリング関数を、開発者に対してサポートします。

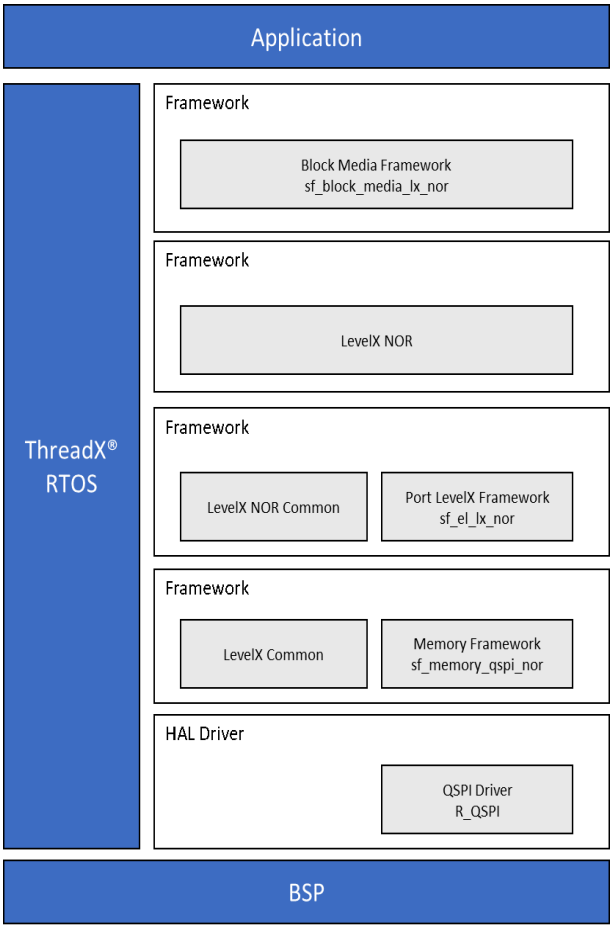


図 26: ブロックメディアフレームワークモジュールのブロック図

4.1.7.2 ブロックメディアフレームワークモジュールの API の概要

ブロックメディアフレームワークモジュールは、モジュールのオープン、リード、ライト、クローズを行う API を定義します。次の表には、使用可能なすべての API 関数のリスト、API コールの例、各 API の簡単な説明が記載されています。ステータス戻り値の表は API 要約表の後にあります。

ブロックメディアフレームワークモジュールの API の要約

Function Name	Example API Call and Description
SF_BLOCK_MEDIA_LX_NOR_Open	g_sf_block_media_lx_nor0.p_api->SF_BLOCK_MEDIA_LX_NOR_Open(g_sf_block_media_lx_nor0.p_ctrl, g_sf_block_media_lx_nor0.p_cfg); Open LevelX flash device for read/write and control. This function initializes the LevelX driver and hardware the first time it is called out of reset. The underlying flash needs to either be erased or already initialized with LevelX.
SF_BLOCK_MEDIA_LX_NOR_Read	g_sf_block_media_lx_nor0.p_api->SF_BLOCK_MEDIA_LX_NOR_Read(g_sf_block_media_lx_nor0.p_ctrl, p_dest, start_sector, sector_count); Read data from flash using LevelX.
SF_BLOCK_MEDIA_LX_NOR_Write	g_sf_block_media_lx_nor0.p_api->SF_BLOCK_MEDIA_LX_NOR_Write(g_sf_block_media_lx_nor0.p_ctrl, p_src, start_sector, sector_count); Write data to flash using LevelX.
SF_BLOCK_MEDIA_LX_NOR_Control	g_sf_block_media_lx_nor0.p_api->SF_BLOCK_MEDIA_LX_NOR_Control(g_sf_block_media_lx_nor0.p_ctrl, command, p_data); Send control commands to Block Media LevelX NOR driver.

Function Name	Example API Call and Description
SF_BLOCK_MEDIA_LX_NOR_Close	g_sf_block_media_lx_nor0.p_api->SF_BLOCK_MEDIA_LX_NOR_Close(g_sf_block_media_lx_nor0.p_ctrl); Close an open Block Media LevelX NOR driver.
SF_BLOCK_MEDIA_LX_NOR_VersionGet	g_sf_block_media_lx_nor0.p_api->SF_BLOCK_MEDIA_LX_NOR_VersionGet(&version); Return the version of the firmware and API using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	LevelX flash is available and is now open for read, write and control access.
SSP_ERR_ASSERTION	p_ctrl, p_cfg or an input pointer is NULL.
SSP_ERR_ALREADY_OPEN	The block media LevelX NOR instance has already been opened. No configurations were changed. Call the associated Close function or use associated Control commands to reconfigure the instance.
SSP_ERR_MEDIA_OPEN_FAILED	LevelX NOR or the underlying flash failed to open. The underlying flash needs to either be erased or already initialized with LevelX.
SSP_ERR_NOT_OPEN	The block media is not open.
SSP_ERR_READ_FAILED	Data read failed.
SSP_ERR_WRITE_FAILED	Data write failed.
SSP_ERR_UNSUPPORTED	This module doesn't support requested command.

Name	Description
SSP_ERR_SECTOR_RELEASE_FAILED	Sector release command failed.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.7.3 ブロックメディアフレームワークモジュールの動作の概要

sf_block_media_lx_nor フレームワークモジュールのブロックメディアフレームワークは、QSPI NOR フラッシュメモリデバイスとのインタフェース用のハイレベル API を提供すると同時に、ウェアレベリングおよびファイルシステムの動作もサポートします。ウェアレベリングは、SSP 内の Express Logic LevelX コンポーネントを使用して実装されています。ファイルシステムのサポートは、SSP に統合された Express Logic FileX コンポーネントを使用して実装されます。これらのコンポーネントは連動して、QSPI NOR Flash ファイルシステムの動作を必要とする組み込みアプリケーションのサポートを簡単にします。

ブロックメディアフレームワークは、QSPI NOR フラッシュメモリに対するデータのリード、ライト、制御のための API 関数を提供します。ブロックメディアフレームワークモジュールのオープンと初期化が成功すると、これらの関数に加えて FileX 関連のすべての API 関数が使用可能になります。これらの完全な説明については、FileX のユーザーマニュアルを参照してください。

ブロックメディアフレームワークモジュールは、他の SSP メディアモジュールに共通の標準インタフェースを使用します。たとえば、SDMMC、SPI フラッシュ、および SDRAM/RAM メモリをサポートするモジュールは同じ API コールを使用するため、プログラミングインタフェースは、すべてのメディアドライバーで同じになります。これらのモジュールは相互に簡単に交換できます。デバイス適応ドライバー（r_qspi, など）にはメモリフレームワークインタフェースを使用してアクセスします。これらのドライバーは、メディア I/O 操作の実行に必要なデバイス固有のコードを提供します。メモリインタフェース関数コールによって渡される構成構造体と制御構造体も、通常はデバイス固有です。

ブロックメディアフレームワークモジュールの動作に関する重要な注意事項と制限事項

ファイルの作成、ライト、およびリードを行うためには、メディアを消去し、フォーマットしてから sf_block_media_lx_nor を使用する必要があります。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.7.4 アプリケーションへのブロックメディアフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにブロックメディアフレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

ブロックメディアフレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(ブロックメディアフレームワークのデフォルト名は g_sf_block_media_lx_nor. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

ブロックメディアフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_block_media_lx_nor0 Block Media Framework on sf_block_media_lx_nor	Threads	New Stack> Framework> File System> Block Media Framework on sf_block_media_lx_nor

次の図に示すように、sf_block_media_lx_nor のブロックメディアフレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

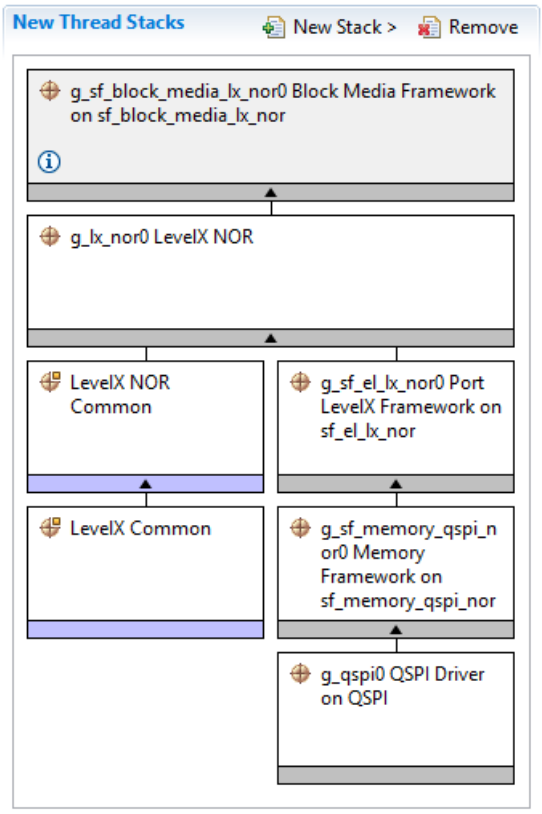


図 27: ブロックメディアフレームモジュールのスタック

4.1.7.5 ブロックメディアフレームワークモジュールの構成

ユーザーは必要な動作に合わせてブロックメディアフレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_block_media_lx_nor のブロックメディアフレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_sf_block_media_lx_nor0	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ブロックメディアフレームワークのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

LevelX NOR 共通インスタンスの構成設定

ISDE Property	Value	Description
Name	g_lx_nor0	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

LevelX 共通インスタンスの構成設定

ISDE Property	Value	Description
Thread Safety	Enabled, Disabled Default: Disabled	If Enabled, this makes LevelX thread-safe by using a ThreadX mutex object throughout the API.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_el_lx_nor の Port LevelX フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_sf_el_lx_nor0	Module name.
Event Callback	NULL	Name of the function to call when an event occurs.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_memory_qspi_nor のメモリフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_sf_memory_qspi_nor0	Module name.
Write of Erase Timeout (in ticks)	30000	Timeout ticks for waiting on write or erase to complete.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_qspi の QSPI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_qspi0	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ブロックメディアフレームワークモジュールのクロック構成

ブロックメディアフレームワークモジュールは、PCLKA をクロックソースとして使用する QSPI ペリフェラルを使用します。

実行時にクロック周波数を変更するには、CGC インタフェースを使用します。

ブロックメディアフレームワークモジュールのピン構成

ブロックメディアフレームワークモジュールを使用するには、必要に応じて QSPI ペリフェラルのポートピンを設定する必要があります。次の表では、ISDE [Configuration] ウィンドウでのピンの選択方法を示します。

sf_block_media_lx_nor のブロックメディアフレームワークモジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
QSPI	Pins	Select Peripherals > Storage:QSPI QSPI0

4.1.7.6 アプリケーションでのブロックメディアフレームワークモジュールの使用

アプリケーションでブロックメディアフレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) FileX API 関数 fx_system_initialize を使用してメディアを初期化します（sf_el_fx は自動的にこれをコールします）。
2) API 関数 lx_nor_flash_initialize を使用して LevelX NOR を初期化します（g_common_init は自動的にこれをコールします）。

3) FileX API 関数 fx_media_format を使用してメディアをフォーマットします（sf_el_fx は、[Format media during initialization] プロパティが [Enabled] に設定されている場合、自動的にこれをコールします）。メディアが既にフォーマットされている場合、このフォーマットメディアはオプションです。メディアフォーマットを行うと、メディアの内容がすべて消去されます。（オプション）
4) FileX API 関数 fx_media_open を使用してメディアを開きます（sf_el_fx は自動的にメディアを開きます）。
5) 必要に応じて、sf_block_media_api_t::read API 関数（ブロックメディアフレームワーク）、または fx_media_read(),、fx_file_read(). などの FileX API 関数のいずれかを使用してメディアを読み取ります。
6) 必要に応じて sf_block_media_api_t::write API 関数（ブロックメディアフレームワーク）、または fx_media_write(),、fx_file_write(). などの FileX API 関数のいずれかを使用してメディアに書き込みます。

注 :fx_media_open のコールが成功したら、リードおよびライトだけでなくすべての FileX API を使用することができます。

これらの共通の手順を、以下の図に示す一般的な動作フローで説明します。

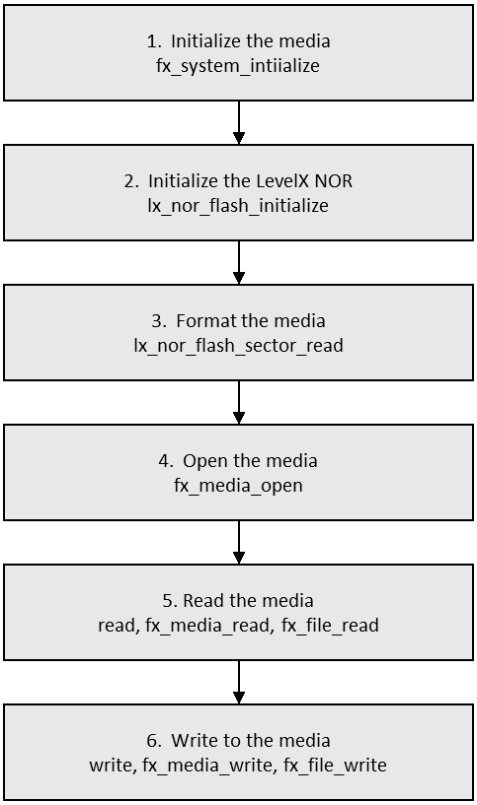


図 28:一般的なブロックメディアフレームワークモジュールアプリケーションのフロー図

4.1.8 ブロックメディア QSPI フレームワーク

ブロックメディアフレームワークモジュールは、r_qspi ドライバーを使用して QSPI フラッシュメモリペリフェラルのリード、ライト、制御を行うための QSPI チャネルを実装できます。このドライバーは、ブロック メディア インタフェースを介してファイル システムとやり取りするために必要なすべての機能を備えています。

4.1.8.1 ブロックメディア QSPI フレームワークモジュールの特長

- QSPI フラッシュメモリデバイス用の QSPI チャネルインタフェースをサポートします。
- QSPI フラッシュメモリのファイルシステムをサポートします。

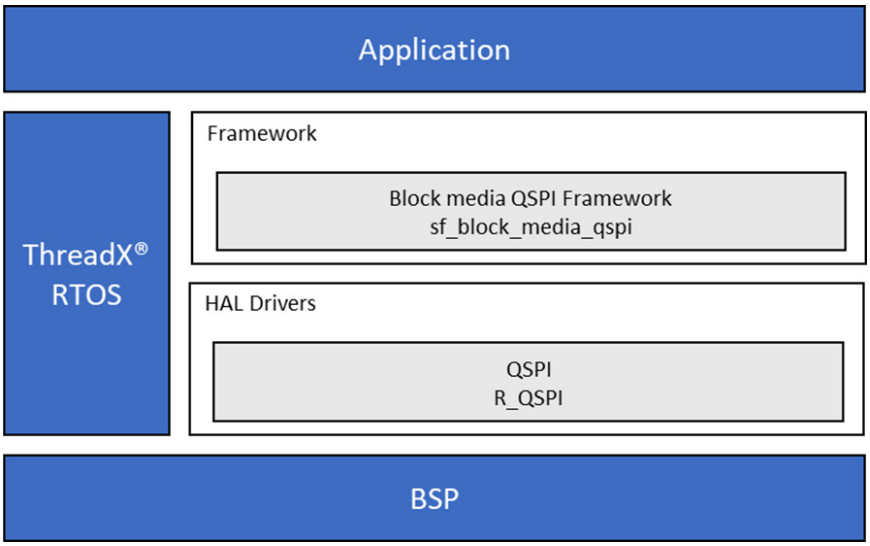


図 29: ブロックメディア QSPI フレームワークモジュールのブロック図

4.1.8.2 ブロックメディア QSPI フレームワークモジュールの API の概要

ブロックメディア QSPI フレームワークモジュールは、メディアに対してオープン、制御、リード、ライトを行う API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

ブロックメディア QSPI フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_block_media_qspi0.p_api->open (g_sf_block_media_qspi0.p_ctrl, g_sf_block_media_qspi0.p_cfg); Open a device channel for read/write and control.
read	g_sf_block_media_qspi0.p_api->read (g_sf_block_media_qspi0.p_ctrl,p_dest,start_block,block_count); Read data from a media channel.
write	g_sf_block_media_qspi0.p_api->write (g_sf_block_media_qspi0.p_ctrl,p_src,start_block,block_count); Write data to a media channel.
ioctl	g_sf_block_media_qspi0.p_api->ioctl(g_sf_block_media_qspi0.p_ctrl, command,p_data); Send control commands to and receives the status from the media port.
close	g_sf_block_media_qspi0.p_api->close (g_sf_block_media_qspi0.p_ctrl); Close the open media channel.
versionGet	g_sf_block_media_qspi0.p_api->versionGet (&version); Return the version of the driver using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_INTERNAL	An internal TheadX error has occurred.
SSP_ERR_NOT_OPEN	Unit is not open
SSP_ERR_ASSERTION	The parameter p_ctrl or p_sample is NULL.
SSP_ERR_IN_USE	Peripheral is still running in another mode; perform Close first.
SSP_ERR_UNSUPPORTED	Command not supported.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.8.3 ブロックメディア QSPI フレームワークモジュールの動作の概要

ブロック メディア フレームワーク インタフェースは単なる抽象インタフェースであり、直接関数呼び出しの代わりに関数ポインターを使用します。関数は FileX と SSP ブロックメディアドライバー（SDMMC、SPI フラッシュ、SDRAM/RAM など）の間でコールされます。このインタフェースはどのメディアドライバーに対しても変わらないため、すべてのメディアドライバーがファイル I/O レイヤーで機能的に同一に見え、コードを変更せずにメディアドライバーを相互に交換できます。デバイス適応ドライバー（sf_block_media_qspi, など）にはブロックメディアフレームワークインタフェースを使用してアクセスします。これらのドライバーは、メディア I/O 操作の実行に必要なデバイス固有のコードを提供します。ブロック メディア関数呼び出しによって渡される構成構造体と制御構造体も、通常はデバイス固有です。

ブロックメディア QSPI フレームワークモジュールの動作に関する重要な注意事項と制限事項

- ファイルの作成、書き込み、および読み取りを行うためには、メディアを少なくとも 1 回フォーマットする必要があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.8.4 アプリケーションでのブロックメディア QSPI フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにブロックメディア QSPI フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

ブロックメディア QSPI フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(ブロックメディア QSPI フレームワークのデフォルト名は g_sf_block_media_qspi0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

ブロックメディア QSPI フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_block_media_qspi0 Block Media Framework on sf_block_media_qspi	Threads	New Stack> Framework> File System> Block Media Framework on sf_block_media_qspi

次の図に示すように、sf_block_media_qspi のブロックメディア QSPI フレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

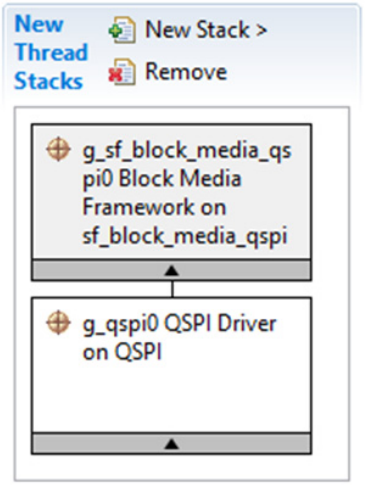


図 30: ブロックメディア QSPI フレームモジュールのスタック

4.1.8.5 ブロックメディア QSPI フレームワークモジュールの構成

ユーザーは必要な動作に合わせてブロックメディアQSPIフレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロ

パティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_block_media_qspi のブロックメディア QSPI フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_block_media_qspi	Module name.
Block size of media in bytes	4096	Block size selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ブロックメディア QSPI フレームワークモジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_qspi での QSPI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_qspi0	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ブロックメディア QSPI フレームワークモジュールのクロック構成

ブロックメディア QSPI フレームワークモジュールは、PCLKA をクロックソースとして使用します。

実行時にクロック周波数を変更するには、CGC インタフェースを使用します。

ブロックメディア QSPI フレームワークモジュールのピン構成

ブロックメディア QSPI フレームワークモジュールを使用するには、ペリフェラルの入力と出力のポートピンを ISDE のピンコンフィギュレータで設定する必要があります。次の表では、ISDE [Configuration] ウィンドウでのピンの選択方法を示します。

sf_block_media_qspi のブロックメディア QSPI フレームワークモジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
QSPI	Pins	Select Peripherals > Storage:QSPI>QSPI0

4.1.8.6 アプリケーションでのブロックメディア QSPI フレームワークモジュールの使用

一般的なアプリケーションの sf_block_media_qspi でブロックメディア QSPI フレームワークモジュールを使用する際の手順は次のとおりです。

1) FileX API fx_system_initialize を使用してメディアを初期化します（sf_el_fx は自動的にこれをコールします）。
2) FileX API fx_media_format を使用してメディアをフォーマットします（sf_el_fx は [Format media during initialization] プロパティが [Enabled] に設定されている場合、自動的にこれをコールします）。メディアが既にフォーマットされている場合、このフォーマットメディアはオプションです。メディアフォーマットを行うと、メディアの内容がすべて消去されます。（オプション）
3) FileX API fx_media_open を使用してメディアを開きます（sf_el_fx は自動的にメディアを開きます）。
4) 必要に応じて、sf_block_media_api_t::read API（ブロックメディアフレームワーク）、または fx_media_read()、fx_file_read(). などの FileX API のいずれかを使用してメディアを読み取ります。
5) 必要に応じて、sf_block_media_api_t::write API（ブロックメディアフレームワーク）、または fx_media_write()、fx_file_write(). などの FileX API のいずれかを使用してメディアに書き込みます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

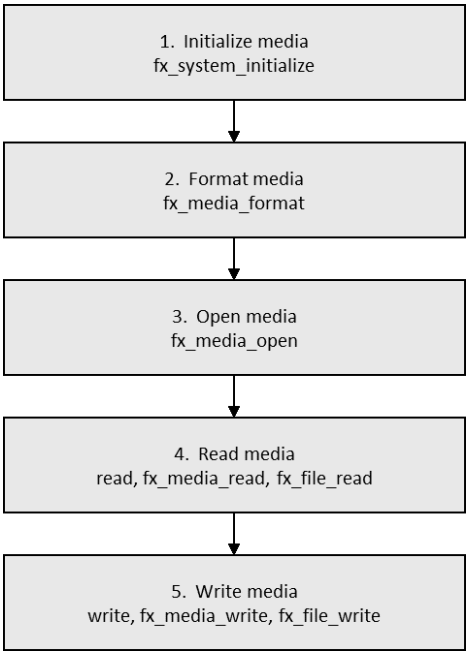


図 31:一般的なブロックメディア QSPI フレームワークモジュールアプリケーションのフロー図

4.1.9 ブロックメディア RAM フレームワーク

ブロックメディアフレームワークモジュールは、RAM メモリのリード / ライト領域を対象としたリード、ライト、および制御用に、RAM にファイルシステムを実装できます。このフレームワークは、ブロックメディアインタフェースを介してファイルシステムとやり取りするために必要なすべての機能を備えています。

4.1.9.1 ブロックメディア RAM フレームワークモジュールの特長

- 線形メモリマップデバイスでの FileX の実行を有効化します。
- RAM に対するデータの高速一時ストレージです。

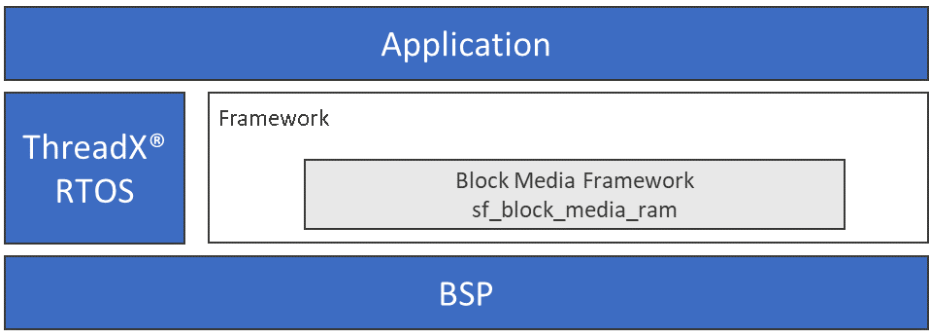


図 32:ブロックメディア RAM フレームワークモジュールのブロック図

4.1.9.2 ブロックメディア RAM フレームワークモジュールの API の概要

ブロックメディア RAM フレームワークモジュールは、モジュールのオープン、リード、ライト、クローズを行う API を実装します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

ブロックメディア RAM フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_block_media_ram0.p_api->open (g_sf_block_media_ram0.p_ctrl, g_sf_block_media_ram0.p_cfg); Open device for read, write and control.
read	g_sf_block_media_ram0.p_api->read (g_sf_block_media_ram0.p_ctrl, p_dest, start_block, block_count); Read data from RAM buffer.
write	g_sf_block_media_ram0.p_api->write (g_sf_block_media_ram0.p_ctrl, p_source, start_block, block_count); Write data to RAM buffer.
ioctl	g_sf_block_media_ram0.p_api->ioctl (g_sf_block_media_ram0.p_ctrl, command, p_data); Send control commands to and receive status of RAM buffer.
close	g_sf_block_media_ram0.p_api->close (g_sf_block_media_ram0.p_ctrl); Close the framework.

Function Name	Example API Call and Description
versionGet	g_sf_block_media_ram0.p_api->versionGet (&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_UNSUPPORTED	Command not supported.
SSP_ERR_NOT_OPEN	Unit is not open
SSP_ERR_ASSERTION	A parameter is NULL.
SSP_ERR_IN_USE	Framework is already open.
SSP_ERR_INVALID_BLOCKS	Invalid block passed.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.9.3 ブロックメディア RAM フレームワークモジュールの動作の概要

ブロック メディア フレームワーク インタフェースは単なる抽象インタフェースであり、直接関数呼び出しの代わりに関数ポインターを使用します。関数は FileX ブロックメディアドライバーと SSP ブロックメディアドライバー（SDMMC、SPI フラッシュ、SDRAM/RAM など）の間でコールされます。このインタフェースはどのメディアドライバーに対しても変わらないため、すべてのメディアドライバーがファイル I/O レイヤーで機能的に同一に見え、コードを変更せずにメディアドライバーを相互に交換できます。デバイス適応ドライバー（sf_block_media_ram, など）にはブロックメディアフレームワークインタフェースを使用してアクセスします。これらのドライバーは、メディア I/O 操作の実行に必要なデバイス固有のコードを提供します。ブロック メディア関数呼び出しによって渡される構成構造体と制御構造体も、通常はデバイス固有です。

ブロックメディア RAM フレームワークモジュールの動作に関する重要な注意事項と制限事項

メディアをオープンし、ファイルの作成、ライト、リードを行うには、メディアをフォーマットする必要があります。

使用されているメモリ領域は、コアによって直接リードおよびライトができる必要があります。内部 SRAM または SDRAM のサイズは、ブロックメディア RAM に対して十分でなければなりません。また、パワーサイクルをまたいでデータを保持することはできません。

ブロックカウントは、ブートレコード、FAT 領域、ルートディレクトリ、ディレクトリセクターを考慮して割り当てる必要があります。RAM でファイルシステムを使用している場合、最小ブロックカウントは 4 にする必要があります。これは、最初の 3 セクターがブートレコード、FAT 領域、ルートディレクトリのために予約されており、4 番目のセクターはデータセクター用に使用されるためです。

FileX 用の RAM を使用するには、[Format media during initialization] プロパティが有効になっている必要があります。

ブロックメディア RAM からブロックカウントおよびブロックサイズを構成するには、[File System is on block media] プロパティが true になっている必要があります。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.9.4 アプリケーションでのブロックメディア RAM フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにブロックメディア RAM フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。ブロックメディア RAM フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(ブロックメディア RAM フレームワークのデフォルト名は g_sf_block_media_ram0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

ブロックメディア RAM フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ sf_block_media_ram0 Block Media Framework on sf_block_media_ram	Threads	New Stack> Framework> File System> Block Media Framework on sf_block_media_ram

次の図に示すように、sf_block_media_ram のブロックメディア RAM フレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

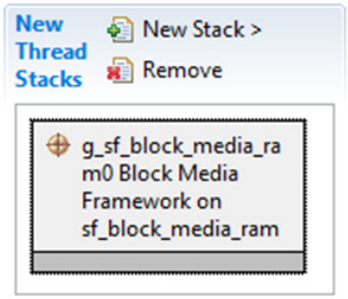


図 33: ブロックメディア RAM フレームモジュールのスタック

4.1.9.5 ブロックメディア RAM フレームワークモジュールの構成

ユーザーは必要な動作に合わせてブロックメディアRAMフレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_block_media_ram のブロックメディア RAM フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_block_media_ram0	Module name.
Block size of media in bytes	512	Block size of media in bytes selection
Number of blocks to allocate	16	To make use of file system, block count must be assigned by considering the boot record, FAT area and root directory.

ISDE Property	Value	Description
Enter the valid section for RAM buffer allocation	noinit	Enter the valid section for RAM buffer allocation

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ブロックメディア RAM フレームワークモジュールのクロック構成

ブロックメディア RAM フレームワークモジュールは、ICLK を内部 RAM のソースとして使用します。

ブロックメディア RAM フレームワークモジュールのピン構成

ブロックメディア RAM フレームワークモジュールは内部 RAM を使用するため、外部ピンは必要ありません。

4.1.9.6 アプリケーションでのブロックメディア RAM フレームワークモジュールの使用

一般的なアプリケーションの sf_block_media_ram でブロックメディア RAM フレームワークモジュールを使用する際の手順は次のとおりです。

1) FileX API fx_system_initialize を使用してメディアを初期化します（sf_el_fx は自動的にこれをコールします）。
2) FileX API fx_media_format を使用してメディアをフォーマットします（sf_el_fx は [Format media during initialization] プロパティが [Enabled] に設定されている場合、自動的にこれをコールします）。メディアが既にフォーマットされている場合、このフォーマットメディアはオプションです。メディアフォーマットを行うと、メディアの内容がすべて消去されます。（オプション）
3) FileX API fx_media_open を使用してメディアを開きます（sf_el_fx は自動的にメディアを開きます）。
4) 必要に応じて、sf_block_media_api_t::read API（ブロックメディアフレームワーク）、または fx_media_read(),fx_file_read(). などの FileX API のいずれかを使用してメディアを読み取ります。
5) 必要に応じて sf_block_media_api_t::write API（ブロックメディアフレームワーク）、または fx_media_write(),fx_file_write(). などの FileX API のいずれかを使用してメディアを読み取ります。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

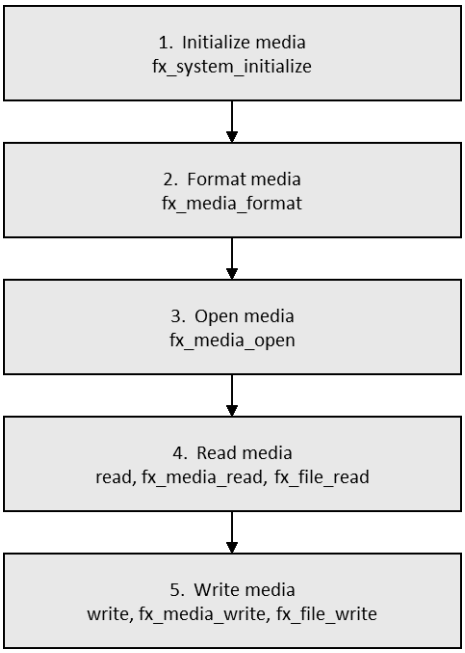


図 34:一般的なブロックメディア RAM フレームワークモジュールアプリケーションのフロー図

4.1.10 ブロックメディア SDMMC フレームワーク

ブロックメディアフレームワークモジュールは、SDHI（SD ホストインタフェース）ペリフェラルおよび SD/MMC メディアドライバーを使用して、SD カードおよび eMMC 組み込みデバイスに対するリード、ライト、および制御を実行するための SD/MMC バスプロトコルを実装できます。このドライバーは、ブロック メディア インタフェースを介してファイル システムとやり取りするために必要なすべての機能を備えています。

4.1.10.1 ブロックメディア SDMMC フレームワークモジュールの特長

- SD/MMC 向け SDHI ホストのインタフェースをサポート
- SDSC (SD 標準容量)、SDHC (SD 大容量)、および eMMC (組み込み) をサポート
- 1 ビット、4 ビット、または 8 ビット（eMMC のみ）のデータバスをサポート

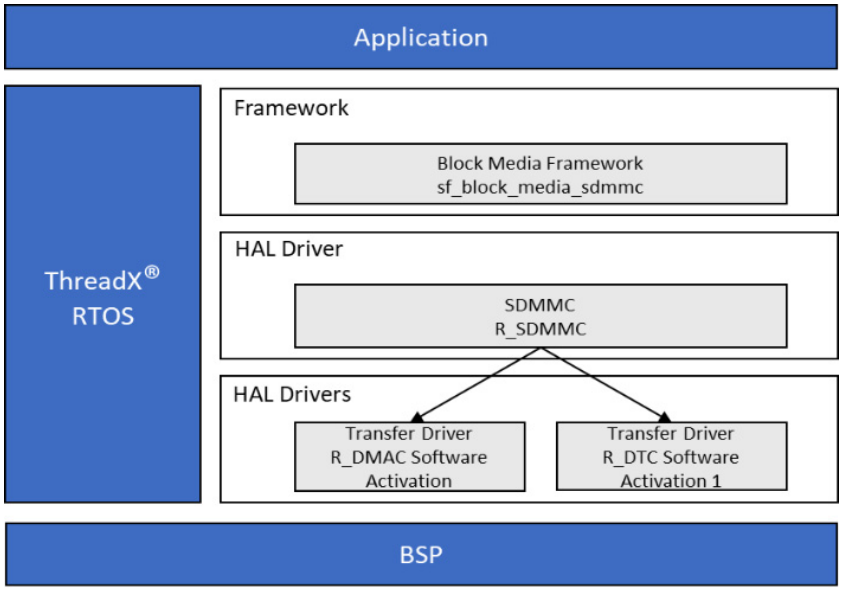


図 35: ブロックメディア SDMMC フレームワークモジュールのブロック図

4.1.10.2 ブロックメディア SDMMC フレームワークモジュールの API の概要

ブロックメディアフレームワークは、SDMMC を対象としたオープン、リード、ライト、制御、クローズのための API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

ブロックメディア SDMMC フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_block_media_sdmmc_api->open(g_sf_block_media_sdmmc.p_ctrl, g_sf_block_media_sdmmc.p_cfg); Open device for read/write and control.
read	g_sf_block_media_sdmmc_api->read(g_sf_block_media_sdmmc.p_ctrl, &destination, startsector, sectorcount); Read data from SD/MMC.

Function Name	Example API Call and Description
write	g_sf_block_media_sdmmc_api->write(g_sf_block_media_sdmmc.p_ctrl, &source, startsector, sectorcount); Write data to SDMMC channel.
ioctl	g_sf_block_media_sdmmc_api->ioctl(g_sf_block_media_sdmmc.p_ctrl, command, &data); Send control commands to the SD/MMC port and receive the status of the SD/MMC port.
close	g_sf_block_media_sdmmc_api->close(g_sf_block_media_sdmmc.p_ctrl); Close open device port.
versionGet	g_sf_block_media_sdmmc_api->versionGet(&version); Get version of Block Media SD/MMC driver.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_IN_USE	The channel specified has already been opened. No configurations were changed. Call the associated Close function or use associated control commands to reconfigure the channel.
SSP_ERR_ASSERTION	The parameter p_ctrl or p_sample is NULL.

Name	Description
SSP_ERR_WRITE_PROTECTED	SD or MMC card is Write Protected.
SF_INFO_NOT_AVAILABLE	Information not available possibly because card has been removed or is defective.
SSP_ERR_NOT_OPEN	The channel is not opened.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.10.3 ブロックメディア SDMMC フレームワークモジュールの動作の概要

ブロック メディア フレームワーク インタフェースは単なる抽象インタフェースであり、直接関数呼び出しの代わりに関数ポインターを使用します。関数は FileX と SSP ブロックメディアドライバー（SDMMC や SPI フラッシュなど）の間で呼び出されます。このインタフェースはどのメディアドライバーに対しても変わらないため、すべてのメディアドライバーがファイル I/O レイヤーで機能的に同一に見え、コードを変更せずにメディアドライバーを相互に交換できます。デバイス適応ドライバー（sf_block_media_sdmmc, など）にはブロックメディアフレームワークインタフェースを使用してアクセスします。これらのドライバーは、メディア I/O 操作の実行に必要なデバイス固有のコードを提供します。ブロック メディア関数呼び出しによって渡される構成構造体と制御構造体も、通常はデバイス固有です。

ブロックメディア SDMMC フレームワークモジュールの動作に関する重要な注意事項と制限事項

- ファイルの作成、書き込み、および読み取りを行うためには、メディアを少なくとも 1 回フォーマットする必要があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.10.4 アプリケーションでのブロックメディア SDMMC フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにブロックメディア SDMMC フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

ブロックメディア SDMMC フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（ブロックメディア SDMMC フレームワークのデフォルト名は g_sf_block_media_sdmmc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

ブロックメディア SDMMC フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_block_media_sdmmc0 Block Media Framework on sf_block_media_sdmmc	Threads	New Stack> Framework>File system>Block Media Framework

次の図に示すように、sf_block_media_sdmmc のブロックメディア SDMMC フレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

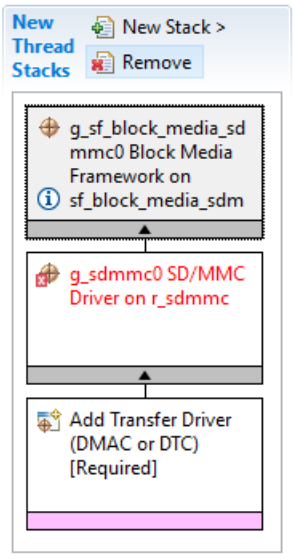


図 36: ブロックメディア SDMMC フレームワークモジュールスタック

4.1.10.5 ブロックメディア SDMMC フレームワークモジュールの構成

ユーザーは必要な動作に合わせてブロックメディア SDMMC フレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユー

ザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_block_media_sdmmc 上のブロックメディア SDMMC フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_block_media_sdmmc0	The name to be used for sf_block_media_sdmmc module control block instance.
Block size of media in bytes	512	Media Block size.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ブロックメディア SDMMC フレームワークモジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_sdmmc 上の SDMMC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.

ISDE Property	Value	Description
Name	g_sdmmc0	The name to be used for SDMMC module control block instance. This name is also used as the prefix of the other variable instances.
Channel	0, 1 Default: 1	Channel of SD/MMC peripheral, channel 0 or 1
Media Type	Embedded, Card Default: Embedded	Media is a card or an embedded device. This allows to firmware to know whether to look for card insertion/removal and write protect pins.
Bus Width	1 Bit, 4 Bits, 8 Bits Default: 4 Bits	Data bus with as defined by hardware interface. (8 Bits for eMMC only)
Block Size	512	Block size selection
Card Detection	Not Used, CD Pin Default: CD Pin	Card detection selection
Callback	NULL	(Not required if using Filex) Set to name of user callback function. Provides event that caused interrupt: SDMMC_EVENT_CARD_REMOVED, SDMMC_EVENT_CARD_INSERTED, SDMMC_EVENT_ACCESS, SDMMC_EVENT_SDIO, SDMMC_EVENT_TRANSFER_COMPLETE, SDMMC_EVENT_TRANSFER_ERROR

ISDE Property	Value	Description
Access Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Access interrupt priority selection
Card Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Card interrupt priority selection
DMA Request Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	DMA request interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dmac Software Activation の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Selects if code for parameter checking is to be included in the build
Name	g_transfer0	Module name
Channel	0	
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection

ISDE Property	Value	Description
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Software Activation 1 の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection

ISDE Property	Value	Description
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation 1	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ブロックメディア SDMMC フレームワークモジュールのクロック構成

SDHI ブロック（SDMMC 関数および SDIO 関数の実装に使用）は、クロックのソースとして PCLKA を使用します。データレートを最適化する必要がない限り、SDMMC ペリフェラルに対して固有のクロックを設定する必要はありません。SDMMC ドライバーは、PCLKA 周波数と、SD、SDIO または eMMC デバイスで許可される最大クロックレート（これはメディアデバイスの初期化時に取得されます）に基づいて、適切な内蔵分周器を選択します。

ブロックメディア SDMMC フレームワークモジュールのピン構成

e^2 studio ピンコンフィギュレータを使用して、SDMMC ペリフェラルの I/O ピンを設定します。ほとんどのボード、高速メモリ、SDIO デバイスにおいて、各品のドライブ能力は「中」または「高」に設定する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

注 : 一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号が決まり、それに従って必要な MCU ピンが決定されます。

sf_block_media_sdmmc 上のブロックメディア SDMMC フレームワークモジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
SDHI	Pins	Select Peripherals > Storage:SHDI > SDHI0

注 : 選択シーケンスでは、SCI1 がドライバーに必要なハードウェアターゲットであることを想定しています。

sf_block_media_sdmmc 上のブロックメディア SDMMC フレームワークモジュールのピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, SD_MMC 1 bit SD_MMC 4 bit MMC 8 bit	Select mode as per application
CLK	None, P413 Default: None	Clock Pin
CMD	None, P412 Default: None	Command Pin

Property	Value	Description
DAT0-7	None, PXXX Default: None	Data Pin

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

ブロックメディア SDMMC のその他の設定

メディアのリード / ライト関数および SDIO の拡張リード / ライト関数は非ブロッキング関数なので、DMAC または DTC で割り込みと転送関数が必要です。リード / ライト関数は、初期動作が正常に開始したことを示す SSP_SUCCESS を返します。ただし、ユーザーアプリケーションはユーザーコールバックを待ち、リード / ライトの完了を示す SDMMC_EVENT_TRANSFER_COMPLETE または SDMMC_EVENT_TRANSFER_ERROR イベントを確認する必要があります。

4.1.10.6 アプリケーションでのブロックメディア SDMMC フレームワークモジュールの使用

一般的なアプリケーションで sf_block_media_sdmmc 上のブロックメディア SDMMC フレームワークモジュールを使用する際の手順は次のとおりです。

1) sf_block_media_api_t::open API を使用して、メディアを初期化します。
2) 必要に応じて、sf_block_media_api_t::read API を使用して、メディアを読み取ります。
3) 必要に応じて、sf_block_media_api_t::write API を使用して、メディアに書き込みます。
4) 必要に応じてデータを処理します。
5) 必要に応じて、sf_block_media_api_t::close API を使用して、メディアを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

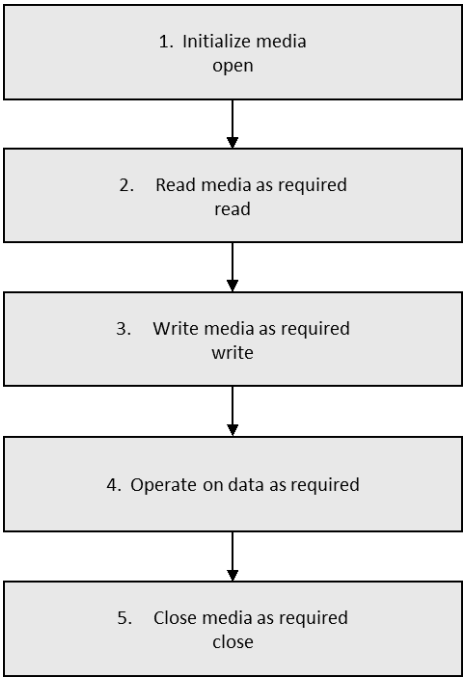


図 37:通常のブロックメディア SDMMC フレームワークモジュールアプリケーションのフロー図

アプリケーションで sf_el_fx を使用して sf_block_media_sdmmc を使用する一般的な手順は次のとおりです。

1) FileX API fx_system_initialize を使用してメディアを初期化します（sf_el_fx は自動的にこれをコールします）。
2) FileX API fx_media_format を使用してメディアをフォーマットします（sf_el_fx は [Format media during initialization] プロパティが [Enabled] に設定されている場合、自動的にこれをコールします）。
3) FileX API fx_media_open を使用してメディアを開きます（sf_el_fx は自動的にメディアを開きます）。
4) 必要に応じて、sf_block_media_api_t::read API（ブロックメディアフレームワーク）、または fx_media_read(), fx_file_read() などの FileX API のいずれかを使用してメディアを読み取ります。
5) 必要に応じて、sf_block_media_api_t::write API（ブロックメディアフレームワーク）、または fx_media_write(), fx_file_write() などの FileX API のいずれかを使用してメディアに書き込みます。

注 :fx_media_open コールが成功した後は、すべての FileX API（read と write だけでなく）を使用できます。

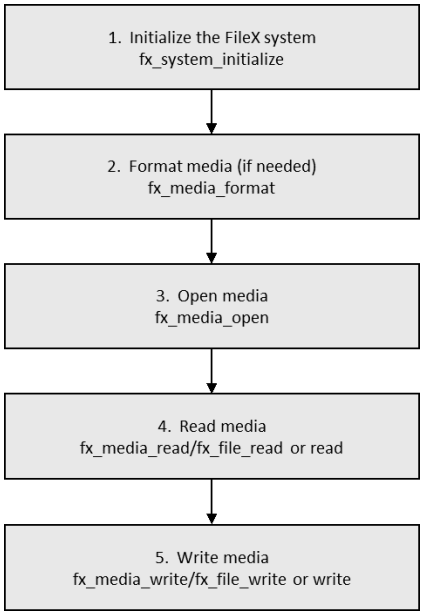


図 38:通常のブロックメディア SDMMC フレームワークモジュールアプリケーションのフロー図

4.1.11 BLE フレームワーク

Bluetooth Smart とも呼ばれる Bluetooth® Low Energy（BLE）はクラシック Bluetooth のサブセットであり、Bluetooth 4.0 コア仕様に導入されました。BLE はクラシック Bluetooth とは対照的に、消費電力を非常に低く抑えるように設計されています。これにより、電力容量が限られている Internet of Thing（IoT）デバイスが、近接するデバイス間で小容量のデータを送信できるようになります。

アプリケーション開発者は、BLE スタックによって提供される機能を利用するためにその API を使用します。さまざまなベンダによって提供される BLE スタック API は標準化されていません。アプリケーション開発者は異なる BLE スタックにコードを移植する際にコードを更新する必要があります。

Synergy BLEフレームワークは、さまざまなベンダによって提供される基礎となるBLEスタック用の汎用インタフェースを提供することでこの問題に対処し、アプリケーションとベンダ固有 BLE スタックコードが対になることを防ぎます。汎用 API の使用により、アプリケーション開発が単純かつ移植可能になります。

BLE フレームワークは BLE アプリケーションにハイレベルの API を提供し、Synergy Software Package（SSP）通信フレームワークを使用します。このフレームワークが、基礎となる BLE モジュールと通信するための UART ドライバーを有効にします。これによって汎用 BLE プロファイルフレームワーク (g_sf_ble_onboard_profile), も生成され、BLE プロファイルに対して統一されたインタフェースが提供されます。RL78G1D BLE ハードウェアモジュールの場合、汎用 BLE プロファイルは BLE モジュールファームウェアによって実装されます。

4.1.11.1 BLE フレームワークモジュールの特長

- ThreadX® RTOS 対応およびスレッドセーフ
- Bluetooth v4.2 準拠フレームワーク
- Generic Access Profile（GAP）機能

  - ユーザー定義アドバタイジングデータ
  - セキュリティモード 1 および 2
  - ペリフェラルロールおよびセントラルロール
  - 最大 6 デバイスに対応するホワイトリスト
  - ボンディングサポート
- Generic Attribute Profile（GATT）機能
  - GATT クライアントおよびサーバー
- Generic Attribute Profile（GATT）API
- Generic Access Profile（GAP）API
- 汎用オンボードプロファイル API

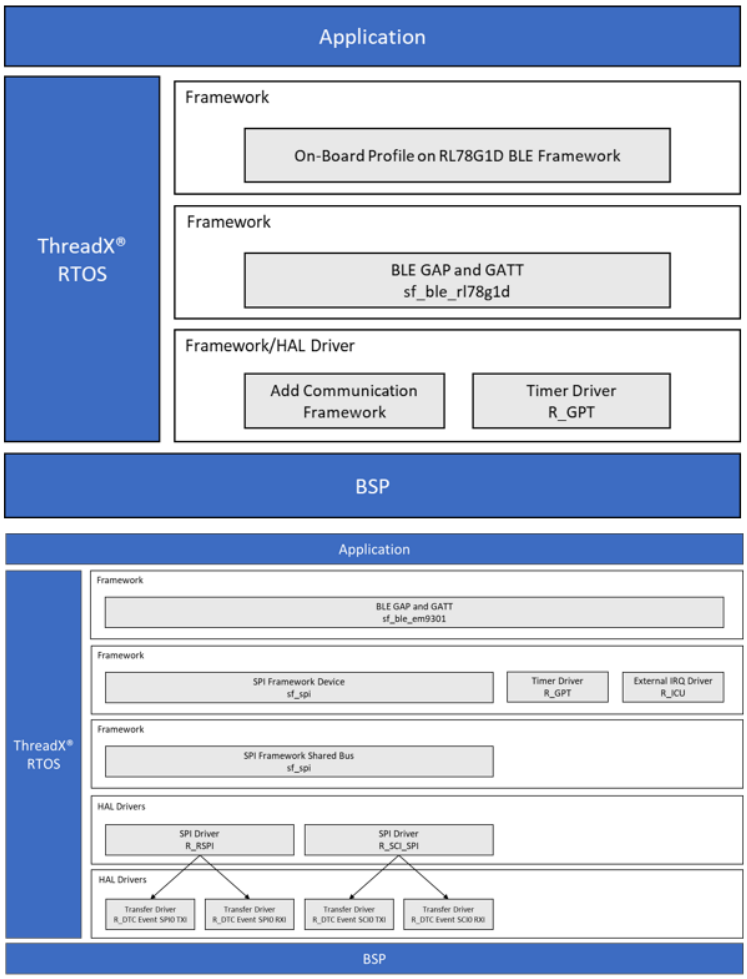


図 39:BLE フレームワークモジュールのブロック図

注 :sf_ble_rl78g1d 上の BLE および GATT は、RL789G1D BLE フレームワークまたは固有のフレームワーク上で、オンボードプロファイルのローレベル実装として使用できます。

4.1.11.2 BLE フレームワークモジュールの API の概要

BLE フレームワークは、初期化、値の設定と取得、モジュールの停止の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

BLE フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_ble0.p_api->open(g_ sf_ble_0.p_cfg); This function initializes and enables the BLE module. It accepts the BLE module configuration as an argument, which has the following parameters: - 48-bit Bluetooth Address - Device scan interval - Device scan window - Device discoverable time - Device connection interval - Slave latency - Supervision timeout - Own address type - Maximum slaves allowed to be connected
close	g_sf_ble0.p_api->close (g_ sf_ble0.p_cfg); This API de-initialize the interface and may put the BLE module in low power mode or power it off. It also closes the driver, disables the driver link, disable the interrupt in the BLE module driver.

Function Name	Example API Call and Description
infoGet	g_sf_ble0.p_api->infoGet (g_sf_ble_0.p_cfg, p_handle, p_ble_info); The infoGet API takes the BLE control structure as an argument. It returns the following information obtained from the BLE module: - Chipset/driver information string - RSSI value (unsigned integer 16 bits)
provisionGet	g_sf_ble0.p_api->provisionGet (g_sf_ble_0.p_cfg, p_ble_provisioning); The provisionGet API gets the BLE GAP provisioning information and takes the BLE control structure as an argument. It returns the following parameters: -Bonding Mode -Security Mode -GAP Role (Central/Master or Peripheral/Slave) - Security Keys
provisionSet	g_sf_ble0.p_api->provisionSet (g_sf_ble_0.p_cfg, p_ble_provisioning); The provisionSet() function provisions BLE module. It takes BLE control structure and provisioning structure as an argument. - Bonding Mode - Security Mode - GAP Role (Central/Master or Peripheral/Slave) - Security Keys - GAP user event callback

Function Name	Example API Call and Description
scan	g_sf_ble0.p_api->scan (g_sf_ble_0.p_cfg, p_scan, p_cnt, p_scan_info); The scan() function takes BLE control structure as an argument. Scan() function returns a list of BLE devices scanned by the BLE module with below parameters. -Bluetooth address (48-bits) -RSSI -Scan data -Advertising Event type The scan() function takes device count as an argument which acts as an in/out parameter. It specifies size of scan result array and BLE framework sets it to count indicating number of scan results stored in array. The function takes scan type as argument(Active/Passive).
advertisementStart	g_sf_ble0.p_api->advertisementStart (g_sf_ble_0.p_cfg, p_advt_info); The advertisementStart() function takes following parameters: - Discovery mode (General/Limited) - Filter policy – Support for scan/connect request filtering combinations - Advertisement data - Connection mode - Advertisement intervals - Channel map - Address type - Advertising type - Scan response data
advertisementStop	g_sf_ble0.p_api->advertisementStop (g_sf_ble_0.p_cfg); Stops advertisement.

Function Name	Example API Call and Description
whitelistAdd	g_sf_ble0.p_api->whitelistAdd (g_sf_ble_0.p_cfg, p_bp_addr); The whitelistAdd() function add devices to the whitelist for advertisements, scan requests and connection requests.
whitelistDel	g_sf_ble0.p_api->whitelistDel (g_sf_ble_0.p_cfg, p_bp_addr); The whitelistDel() function deletes devices from the whitelist for advertisements, scan requests and connection requests.
bondingStart	g_sf_ble0.p_api->bondingStart (g_sf_ble_0.p_cfg, p_handle, p_bp_addr, p_bonding_start); The bondingStart() function start bonding with a remote device.
bondingResponse	g_sf_ble0.p_api->bondingResponse (g_sf_ble_0.p_cfg, p_handle, p_bp_addr, p_bonding_resp); The bondingResponse () function responds to a bonding request.
startEncryption	g_sf_ble0.p_api->startEncryption (g_sf_ble_0.p_cfg, p_enc_info); The startEncryption() function begins an encryption operation.
connect	g_sf_ble0.p_api->connect (g_sf_ble_0.p_cfg, p_handle, p_conn); The function connects to a remote device.

Function Name	Example API Call and Description
disconnect	g_sf_ble0.p_api->disconnect (g_sf_ble_0.p_cfg, p_handle); The disconnect() function disconnects from a remote device.
listen	g_sf_ble0.p_api->listen (g_sf_ble_0.p_cfg); The function listens for an incoming connect request from a remote device.
authorization	g_sf_ble0.p_api->authorization (g_sf_ble_0.p_cfg, &conhandle); The authorization() function authorizes a remote device after connection.
setTxPower	g_sf_ble0.p_api->setTxPower(g_sf_ble_0.p_cfg, &con_handle, &tx_power_info); The setTxPower() function sets the transmit power for the procedure specified by the connection handle.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

BLE フレームワークのオンボードプロファイルモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_ble_onboard_profile0.p_api->open (g_sf_ble_onboard_profile0.p_cfg); This API initializes the interface for data transfers.

Function Name	Example API Call and Description
close	g_sf_ble_onboard_profile0.p_api->close (g_sf_ble_onboard_profile0.p_cfg); This API de-initializes the interface and may put it in low power mode or power it off. The API closes the driver, and disables the driver link and interrupt.
onbpEnable	g_sf_ble_onboard_profile0.p_api->onbpEnable (sf_ble_onboard_profile0.p_cfg, p_handle, profile, p_prf_cb, sec); Enables the profile in server mode or client mode.
onbpServerWriteData	g_sf_ble_onboard_profile0.p_api->onbpServerWriteData (sf_ble_onboard_profile0.p_cfg, p_handle, profile, characteristics, p_data); Updates the value of the characteristic in the local database.
onbpServerSendNotification	g_sf_ble_onboard_profile0.p_api->onbpServerSendNotification (sf_ble_onboard_profile0.p_cfg, p_handle, profile, characteristics, p_data); Sends notifications.
onbpServerSendIndication	g_sf_ble_onboard_profile0.p_api->onbpServerSendIndication (sf_ble_onboard_profile0.p_cfg, p_handle, profile, characteristics, p_data); Sends indications.
onbpClientWriteCCCD	g_sf_ble_onboard_profile0.p_api->onbpClientWriteCCCD (sf_ble_onboard_profile0.p_cfg, p_handle, profile, cccd_char, cccd_val); Sets the Client Configuration Control Descriptor on the remote device.

Function Name	Example API Call and Description
onbpDisable	g_sf_ble_onboard_profile0.p_api->onbpDisable (sf_ble_onboard_profile0.p_cfg, p_handle, profile); Disables the profile in server mode and client mode.
onbpClientReadChar	g_sf_ble_onboard_profile0.p_api->onbpClientReadChar (sf_ble_onboard_profile0.p_cfg, p_handle, profile, characteristics); Reads a GATT characteristic associated with the profile or service.
onbpClientWriteChar	g_sf_ble_onboard_profile0.p_api->onbpClientWriteChar (sf_ble_onboard_profile0.p_cfg, p_handle, profile, characteristics); Writes a GATT characteristic associated with the profile or service.
versionGet	g_sf_ble_onboard_profile0.p_api->versionGet(&version); Retrieve the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

注 :BLE 標準プロファイルに関連するすべての詳細は、BLE プロファイル仕様に示されています。

注 :BLE API は、モジュールが機能をサポートしていない場合、SSP_ERR_UNSUPPORTED を返します。

BLE フレームワーク GATT API の要約

Function Name	Example API Call and Description
gattAddCustomProfiles	g_sf_ble0.p_api->gattAddCustomProfiles (g_sf_ble_0.p_cfg, p_handle, p_sf_ble_svc_dscv_req, p_sf_ble_svc_dscv_rsp, p_rsp_cnt); This function adds custom profiles to the GATT database.
gattServiceDiscovery	g_sf_ble0.p_api->gattServiceDiscovery (g_sf_ble_0.p_cfg, p_handle, p_sf_ble_svc_dscv_req, p_sf_ble_svc_dscv_rsp, p_rsp_cnt); The gattServiceDiscovery() function performs service discovery.
gattCharDiscovery	g_sf_ble0.p_api->gattCharDiscovery (g_sf_ble_0.p_cfg, p_handle, p_sf_ble_svc_dscv_req, p_sf_ble_svc_dscv_rsp, p_rsp_cnt); The gattCharDiscovery() function performs the Char discovery operation.
gattCharDescDiscovery	g_sf_ble0.p_api->gattCharDescDiscovery (g_sf_ble_0.p_cfg, p_handle, start_handle, end_handle, p_sf_ble_svc_dscv_rsp, p_rsp_cnt); Discovers GATT characteristics descriptor on a remote device.
gattCharRead	g_sf_ble0.p_api->gattCharRead (g_sf_ble_0.p_cfg, p_handle, start_handle, p_char_read_req, p_char_read_rsp); Reads GATT characteristics on a remote device.

Function Name	Example API Call and Description
gattCharWrite	g_sf_ble0.p_api->gattCharWrite (g_sf_ble_0.p_cfg, p_handle, p_char_read_req); Writes GATT characteristics on a remote device.
gattCharExecuteWrite	g_sf_ble0.p_api->gattCharExecuteWrite (g_sf_ble_0.p_cfg, p_handle, execute_flag); Executes a write (commit) on GATT characteristics on a remote device.
gattCharWriteLocal	g_sf_ble0.p_api->gattCharWriteLocal (g_sf_ble_0.p_cfg, char_handle, data_length); Update the local GATT database.
gattSendNotify	g_sf_ble0.p_api->gattSendNotify (g_sf_ble_0.p_cfg, p_handle, char_handle); Sends notifications from local GATT server to remote GATT client.
gattSendIndicate	g_sf_ble0.p_api->gattSendIndicate (g_sf_ble_0.p_cfg, p_handle, char_handle); Sends indications from local GATT server to remote GATT client.
gattWriteResponse	g_sf_ble0.p_api->gattWriteResponse (g_sf_ble_0.p_cfg, p_handle, char_handle); Responds to the write characteristic value request from the remote GATT client.
versionGet	g_sf_ble0.p_api->versionGet(&version); Retrieve the API version using the version pointer.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_ASSERTION	Parameter has invalid value.
SSP_ERR_INVALID_PTR	p_version is NULL.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.11.3 BLE フレームワークモジュールの動作の概要

このセクションでは、Synergy BLE フレームワークソフトウェアアーキテクチャの概要を示し、ユーザーのアプリケーションレベルの動作フローシーケンスに沿って、BLE フレームワークの一部として使用される主な SSP モジュールについて詳しく説明します。

注 :BLE フレームワークモジュールの総合的な説明は、『BLE Framework Application Project』にあります。プロジェクトと関連するアプリケーションに関する完全な説明は、*http://www.renesas.com* ホームページの上部の検索バーで「r30qan0309eu」を検索すると見つかります。

BLE フレームワークモジュールの動作に関する重要な注意事項と制限事項

BLE フレームワークはアプリケーションの共通インタフェースを提供します。このインタフェースの実装はモジュールごとに固有です。現在、Synergy BLE フレームワークは、RL78G1D BLE モジュールに対して実装されるインタフェースを定義します。各実装は対応する BLE デバイスドライバーとやりとりします。BLE デバイスドライバーは基礎となる SSP 通信フレームワーク (g_sf_comms) を使用し、このフレームワークが汎用非同期送受信機 (UART)、データトランスファコントローラ (DTC)、汎用 PWM タイマ (GPT) ドライバーなど SSP HAL コンポーネントとやりとりして、BLE モジュールと通信します。次の図に、BLE フレームワークモジュールのアーキテクチャ概要を示します。

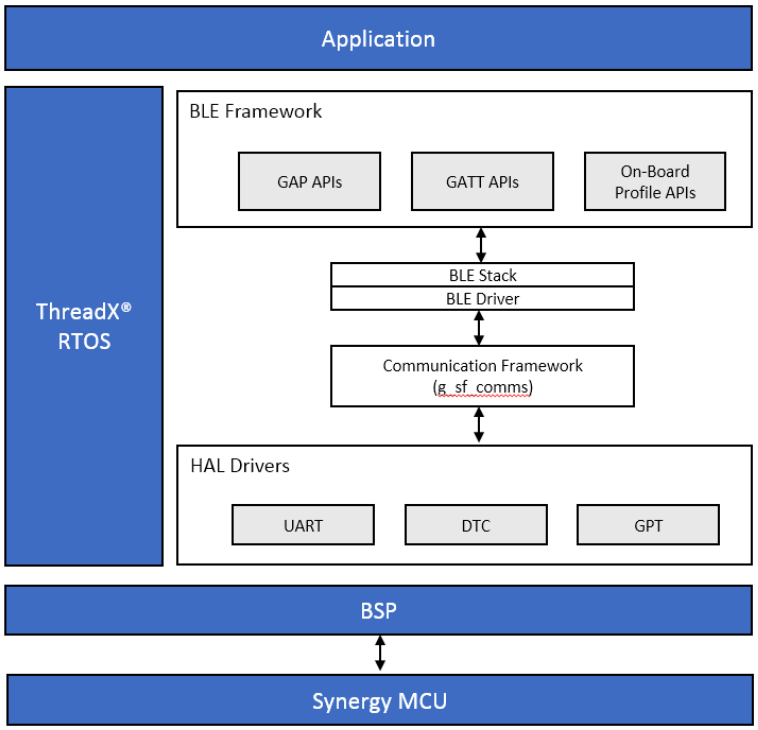


図 40:一般的な BLE モジュールのアーキテクチャタイプ

GAP API および GATT API

BLE フレームワークは、BLE モジュールの構成とプロビジョニングを行うアプリケーションのために汎用インタフェースを提供します。BLE モジュールには、一連の Bluetooth Smart 規格によって指定されたさまざまな構成パラメータがあります。個別のデバイスドライバーまたは BLE モジュール（あるいは両方）ですべての構成パラメータがサポートされるとは限りません。プロビジョニング API で最小限提供されるのは、BLE インタフェースの動作モード、セキュリティモード、セキュリティキー、ボンディングモードを設定するメカニズムです。GAP および GATT レイヤーの API も提供されます。

オンボードプロファイル API

オンボードプロファイル API では、BLE モジュールファームウェアによって実装される BLE プロファイルに統一インタフェースが提供されます。

BLE スタック

通常、BLE モジュールホストスタックは BLE モジュールベンダによって提供されます。一般的に BLE モジュールは、ホスト MCU と BLE モジュールの間の HW/SW パーティションに応じて 3 種類あります。RL78G1D BLE モジュールはネットワークコントローラ実装アーキテクチャに含まれ、このアーキテクチャでは、BLE チップセットに BLE リンクレイヤー、GAP、GATT、オンボードプロファイルのすべての実装が含まれます。このモジュールは、SSP によって提供される sf_comms フレームワークを介して MCU とのインタフェースになります。

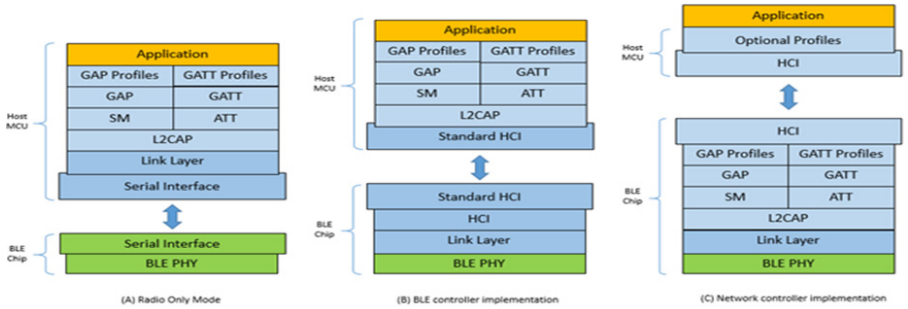


図 41:BLE モジュールのアーキテクチャタイプ

A: BLE 無線専用モード

リンクレイヤー、L2CAP、GATT、GAP レイヤー、プロファイル、アプリケーションはホスト MCU で実行します。物理レイヤーは BLE チップセットで実行します。

B: BLE コントローラ実装

リンクレイヤーは BLE チップセットで実行し、L2CAP レイヤーと上位 BLE プロトコル（GATT、GAP）レイヤー、プロファイル、アプリケーションはホスト MCU で実行します。

C: ネットワークコントローラ実装

リンクレイヤー、L2CAP、GATT、GAP レイヤー、汎用プロファイルは、BLE チップセットで実行します。オプションのプロファイルとアプリケーションはホストプロセッサで実行します。

BLE フレームワークインスタンス

アプリケーションは、BLE フレームワークインスタンスを使用する前に定義する必要があります。インスタンスは次のいずれかへのポインタを含む構造体です。

BLE フレームワーク制御構造体

この構造体はすべての BLE フレームワーク API で使用されます。この構造体にはドライバーハンドルへのポインタが含まれます。ドライバーハンドルは、BLE デバイスドライバーで必要な情報を格納するためにフレームワークが使用します。

BLE フレームワーク構成構造体

この構造体は sf_ble_api_t::open API に渡されます。この構造体を使用して BLE モジュールを構成できます。この構成は、初期化中（open など）にもプロビジョニング中（sf_ble_api_t::provisionSet API など）にも適用されます。BLE モジュールでサポートされない構成パラメータはフレームワークによって無視されます。

BLE フレームワーク API 構造体

この構造体には、所定のモジュール固有の BLE フレームワーク API へのポインタが含まれます。これらの API の詳細については、「BLE フレームワーク モジュールの設定」を参照してください。

アプリケーションで BLE フレームワークモジュールを使用する際の手順は次のとおりです。

1) BLE ハードウェアモジュールを初期化します。
2) GATT レイヤーのロール（GATT クライアントまたは GATT サーバー）を選択します。スレーブ（ペリフェラル）デバイスを GATT サーバーに、マスタ（セントラル）デバイスを GATT クライアントにするのが最も一般的です。
3) アプリケーションが汎用（オンボード）プロファイル API または GAP API か GATT API を使用して動作を制御します。

注 :GAP プロビジョニング構造体では、BLE ユーザーコールバックがドライバーのスレッドコンテキストで実行されます。アプリケーションは、コールバックロジックを最小限にし、ブロッキングコールが一切ないようにする必要があります。プリント文やブロッキングコールは、BLE ドライバーの実行に遅延をもたらす可能性があります。BLE API が決してユーザーコールバックの中でコールされないようにする必要があります。コールされると、コードの失敗につながる可能性があります。

次の BLE モジュール初期化シーケンスは Synergy 自動生成コードの一部です。

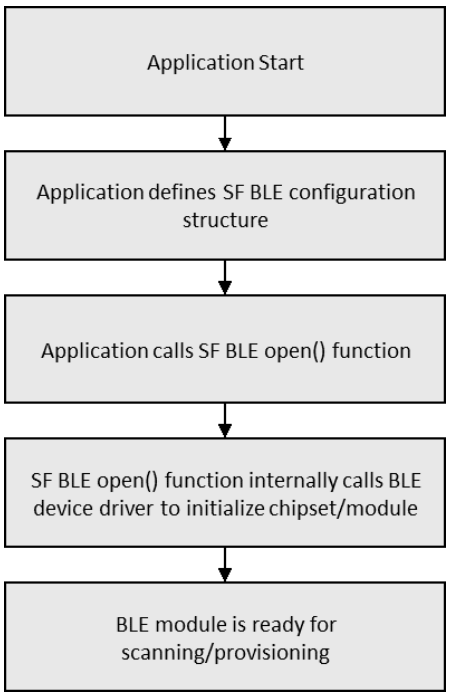


図 42: **自動生成初期化シーケンスフローチャート**

オンボードプロファイルベースのクライアントアプリケーションフローシーケンス

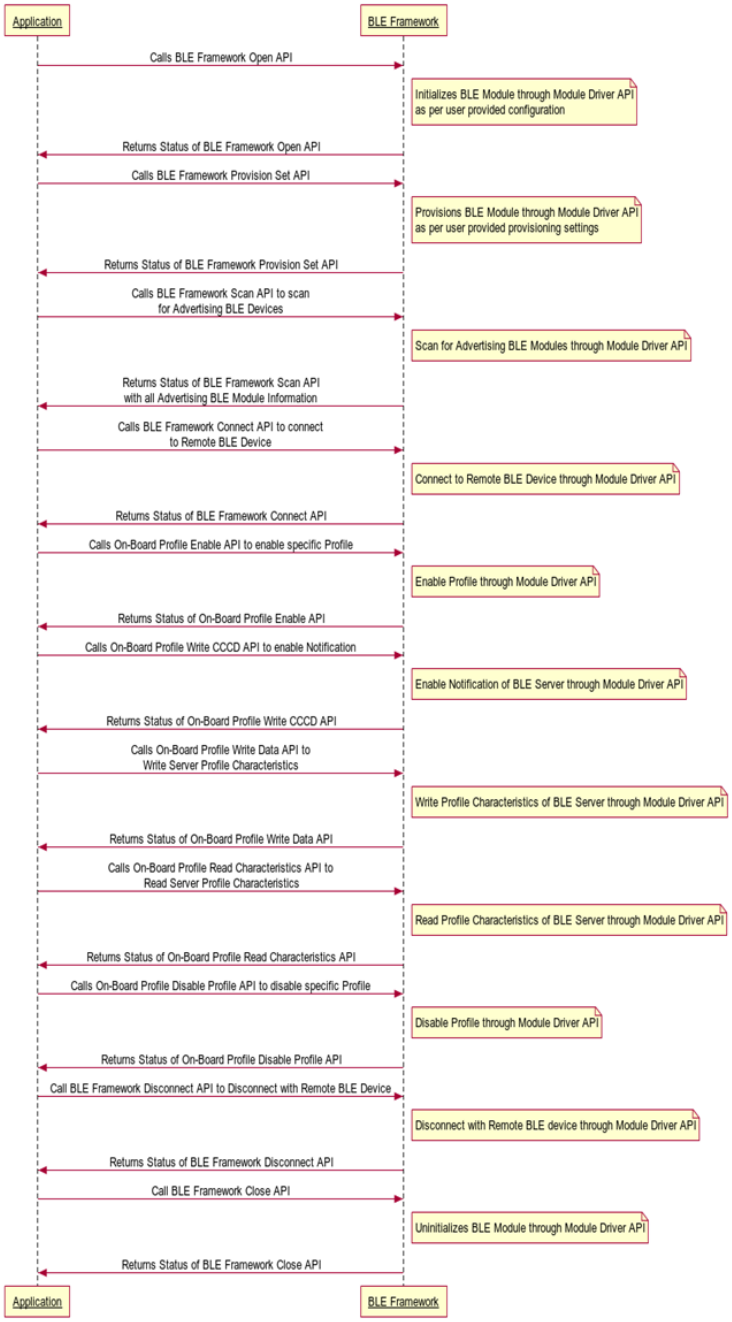


図 43:オンボードプロファイルのクライアントアプリケーションフロー

オンボードプロファイルベースのサーバーアプリケーションフローシーケンス

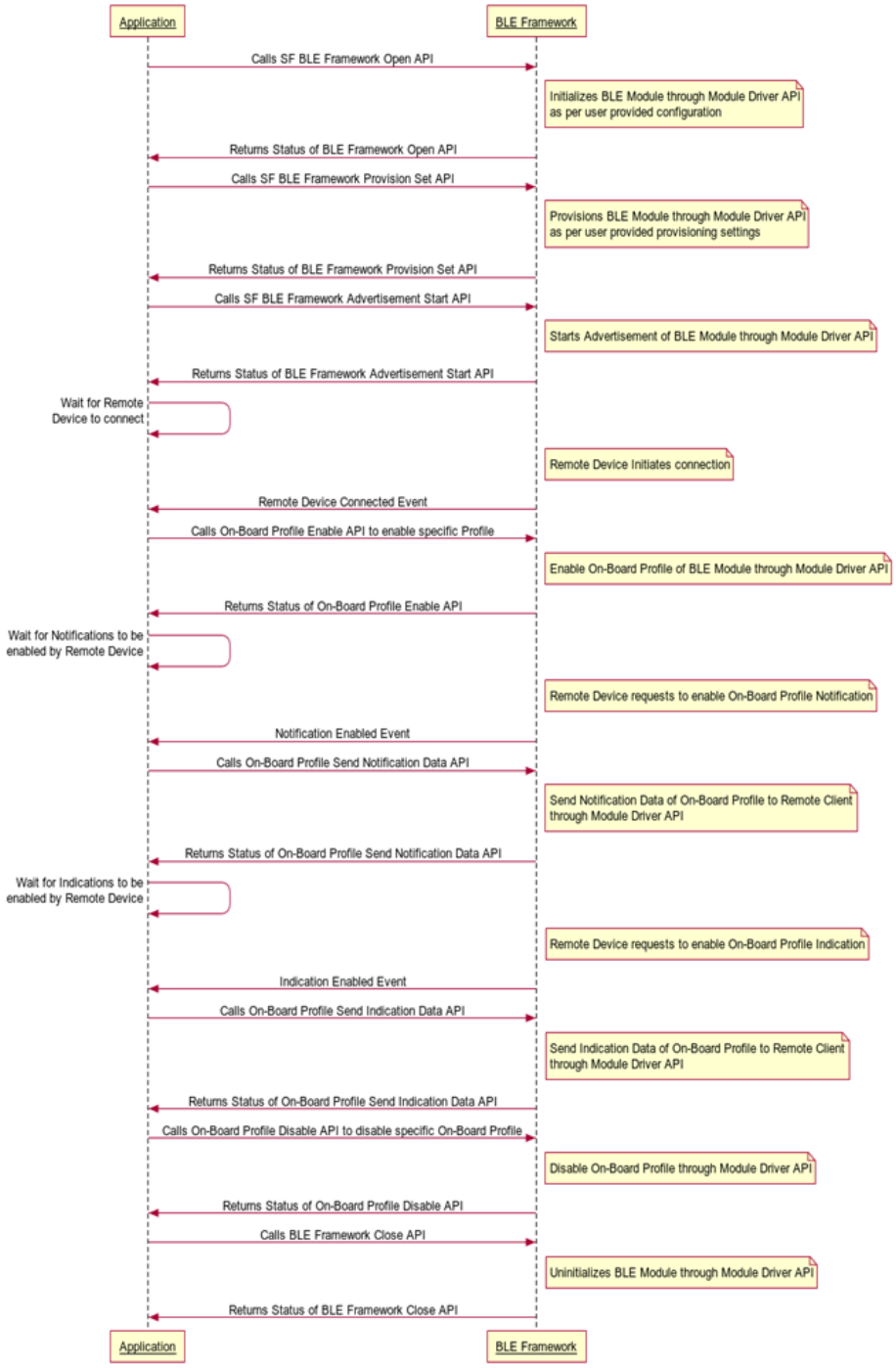


図 44:オンボードプロファイルのサーバーアプリケーションフロー

GAP/GATT ベースのクライアントアプリケーションフローシーケンス

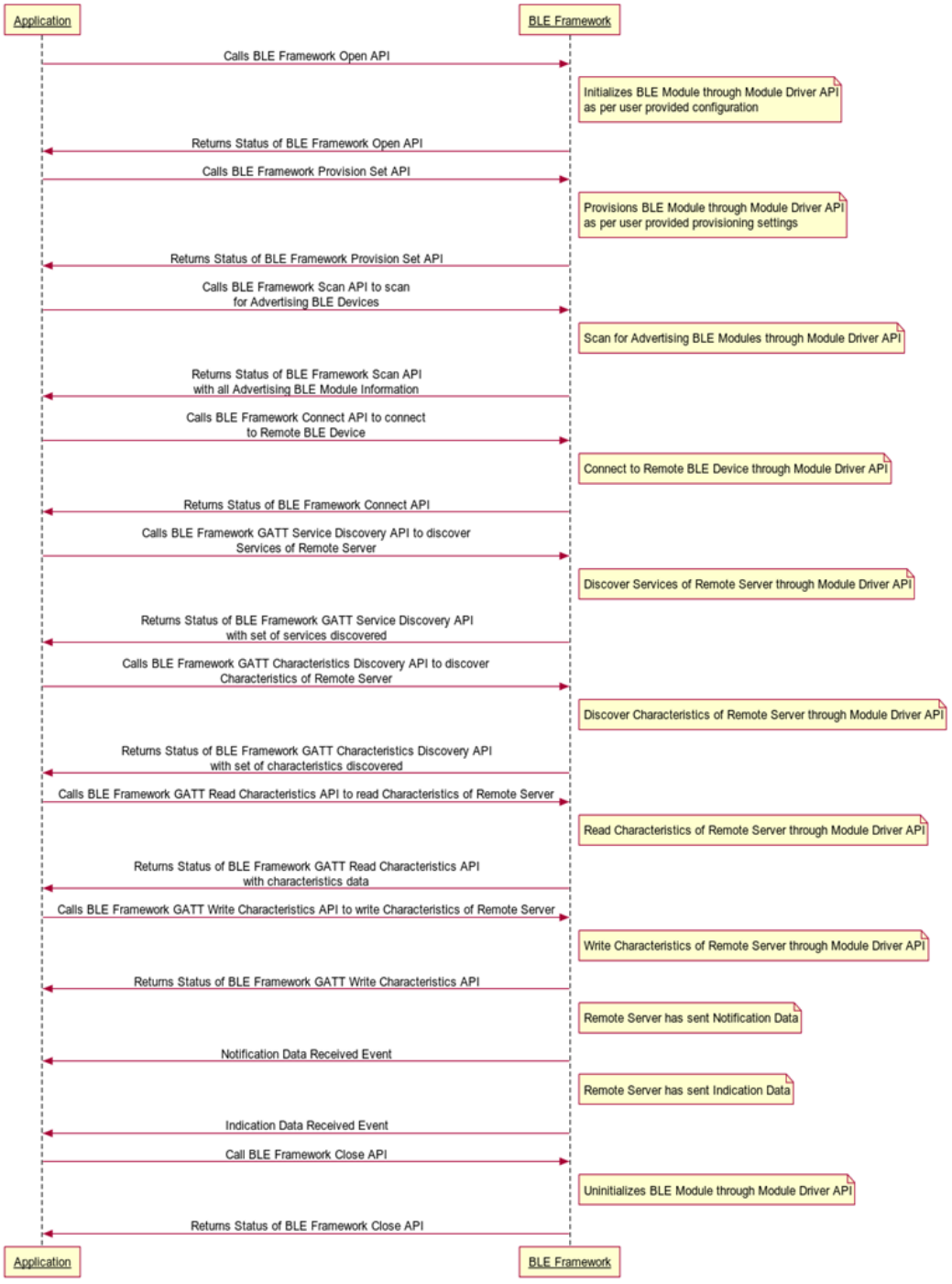


図 45:**GAP および GATT のクライアントアプリケーションフロー**

GAP/GATT ベースのサーバーアプリケーションフローシーケンス

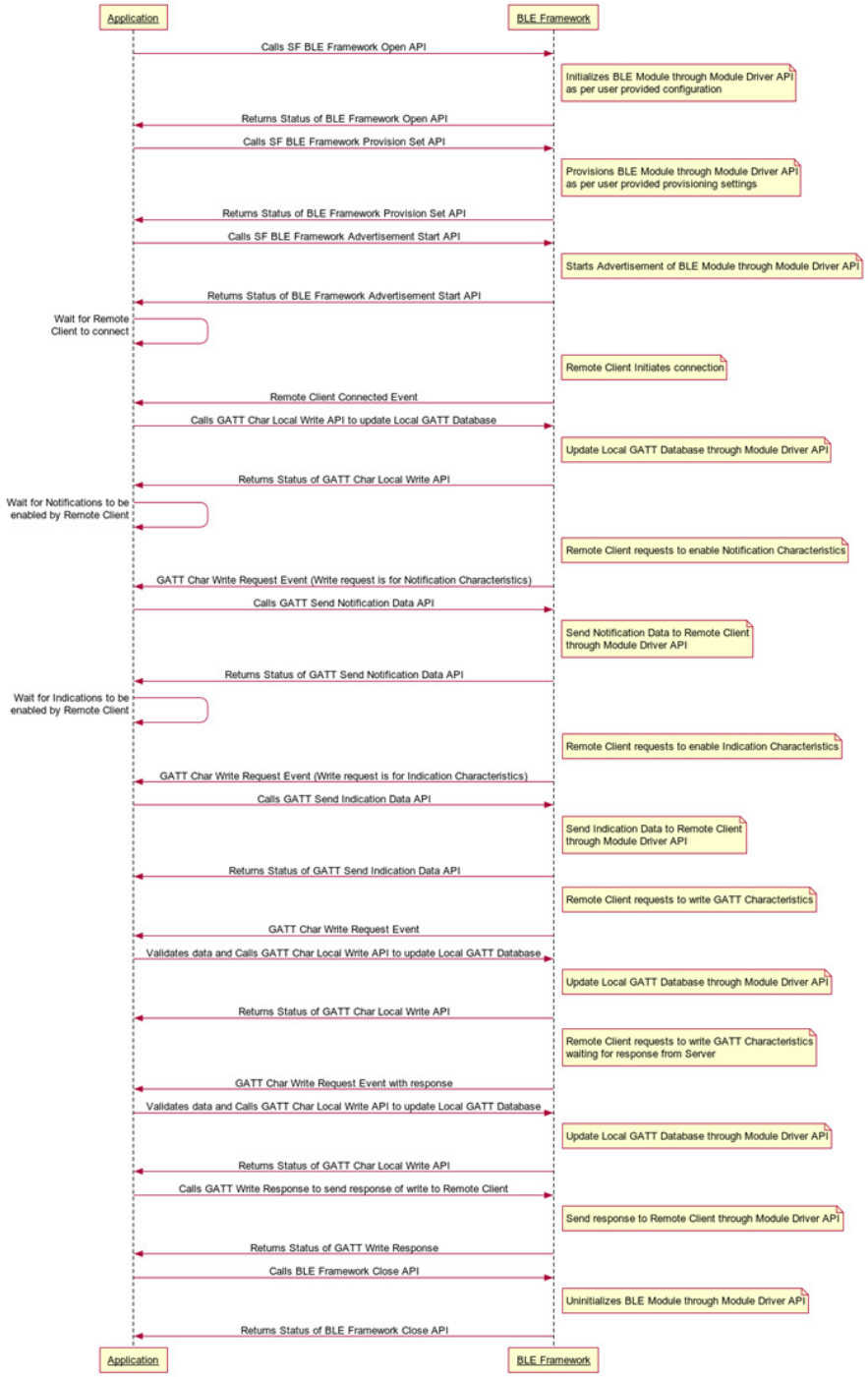


図 46:GAP および GATT のクライアントアプリケーションフロー

セキュリティマネージャは、通信リンクの暗号化に使用されるセキュリティキーの生成と交換の機能を BLE プロトコルスタックに提供します。セキュリティマネージャには次の 2 つの機能があります。

イニシエータ：これは GAP マスタ（セントラル）デバイスです。

レスポンダ：これは GAP スレーブ（ペリフェラル）デバイスです。

イニシエータはセキュリティ手順を開始するマスタデバイスです。ただし、スレーブデバイスは、イニシエータがセキュリティ手順を開始するように非同期で要求できます。

BLE セキュリティで提供されるモードには、各モードに関連付けられたレベルがあります。セキュリティのモードおよびレベルは、認証または未認証のペアリング、暗号化、データ署名のサポートの組み合わせです。ペアリングはさまざまなセキュリティ要件を満たすために必要です。2 種類のペアリングがあります。

認証されたペアリング : デバイスが MITM（中間者）攻撃から保護されます。

認証されていないペアリング : デバイスが MITM から保護されません。

セキュリティモード 1

- セキュリティレベル 1: セキュリティなし
- セキュリティレベル 2: 未認証ペアリングと暗号化
- セキュリティレベル 3: 認証ペアリングと暗号化
- セキュリティレベル 4: 認証 LE セキュア接続ペアリングと暗号化

セキュリティモード 2

- セキュリティレベル 1: 未認証ペアリングとデータ署名
- セキュリティレベル 2: 認証ペアリングとデータ署名

注 :RL78G1D BLE モジュールでは、セキュリティレベル 4 のセキュリティモード 1 はサポートされません。

セキュリティなし：オンボードプロファイルセキュリティが「セキュリティなし」に設定されると、プロファイル通信は BLE GAP のセキュリティモードおよびレベルに関係なくセキュリティなしモードで動作します。

未認証 : このプロファイルセキュリティ方式が動作するためには、セキュリティモード 1 のレベル 1 以外の BLE GAP セキュリティ方式のいずれかを使用する必要があります。モジュールは、リモートデバイスとのペアリングが完了している必要があります。

認証完了：このプロファイルセキュリティ方式が動作するためには、以下の BLE GAP セキュリティ方式のいずれかを使用する必要があります。モジュールは、リモートデバイスとのペアリングが完了している必要があります。

- セキュリティモード 1 のレベル 3
- セキュリティモード 2 のレベル 2

承認：このプロファイルセキュリティ方式が動作するためには、リモートデバイスが GAP 接続の間に BLE フレームワーク承認 API を使用して承認済みである必要があります。このオンボードプロファイルのセキュリティパラメータは、サーバー API のみに固有です。

暗号化：このセキュリティ手順では、プロファイルは暗号化された通信を使用します。このプロファイルセキュリティ方式が動作するためには、セキュリティモード 1 のレベル 1 以外の BLE GAP セキュリティ方式のいずれかを使用する必要があります。暗号化を使用しないセキュリティ方式が使用されている場合、プロファイルは非暗号化モードで動作します。

1) オンボードプロファイルセキュリティは、sf_ble_onboard_profile_api_t::onbpEnable 関数で設定できます。sf_ble_onboard_profile_api_t::onbpEnable API は、サーバーモードまたはクライアントモードでプロファイルを有効にします。これは、クライアントモードとサーバーモードを有効にする汎用 API です。しかし、プロファイルのセキュリティパラメータはサーバー API のみに固有です。セキュ

リティパラメータは、ドライバー API でサーバーのみに設定できます。そのため、ユーザーはサーバー enable API にプロファイルセキュリティを設定する必要があります。

2) プロファイルセキュリティ enum はビットパターンなので、ビット演算子を使用することで複数のプロファイルセキュリティを設定できます。プロファイルセキュリティを設定するときは、以下の点に注意する必要があります。

- プロファイルセキュリティがない場合は、他のセキュリティビットも設定してはいけません。
- 認証完了と未認証の両方にセキュリティビットを設定してはいけません。

BLE セキュリティには以下の手順があります。

ペアリング：この手順は、通信リンクを暗号化する一時暗号化キーを生成するために使用されます。永続暗号化キーはこの暗号化済みの通信リンクで共有して、追加の通信で使用できます。

ボンディング：これは、ペアリングと永続キーの格納を組み合わせたものです。ペアリングの後で永続キーが不揮発性メモリに格納されると、それによって 2 つのデバイス間の永続的なボンディングが形成されます。その後の通信では、デバイスがボンディング手順を実行する必要はありません。

暗号化の確立：通信は永続キーを使用して暗号化されます。

ペアリングによって作成されたセキュアなリンクは接続の間のみ継続しますが、ボンディングではボンドと呼ばれる永続的な関係が形成されます。

BLE セキュリティには 3 つのフェーズがあります。2 つのデバイスが GAP 接続手順を使用して接続を確立した後で、3 つのフェーズによりセキュアな通信リンクを確立します。

- フェーズ 1（ペアリングフェーズ、情報共有）最初のフェーズ 1 では、一時キーの生成に必要なすべての情報が 2 つのデバイス間で共有されます。
- フェーズ 2（ペアリングフェーズ、一時キー共有）このフェーズでは、一時暗号化キー（Short Term Key すなわち STK）が両方のデバイス上で生成されます。これは接続の暗号化に使用されます。こうして暗号化されたリンクをさらに他の通信に使用できます。ピアデバイスが接続されている間、この通信リンクは暗号化されたままになります。
- フェーズ 3（ボンディング、永続キーの共有と格納）ボンディングが必要な場合、デバイスはこのフェーズに進みます。このフェーズでは、暗号化されたリンク（フェーズ 2 で一時キーを使用して確立）を使用して、永続キー（Long Term Key すなわち LTK）が 2 つのデバイス間で交換されます。これらの永続キーは不揮発性メモリに格納され、各接続のためにデバイスが使用できるようになります。

1) BLE フレームワークのテストは RL78G1D BLE ハードウェアモジュールでしか行われていません。さまざまな BLE モジュールのサポートは今後のバージョンで追加される予定です。

2) RL78G1D を使用する BLE フレームワークではコンパイル時に警告が表示されます。すべての警告はサードパーティの RL78G1D ドライバーコードに対するものです。BLE フレームワークファイルには警告はありません。これらの警告はユーザーアプリケーションには影響しません。

3) BLE フレームワークでのカスタムプロファイルサポートは、RL78G1D タイプの BLE ハードウェアモジュールのみに制限されています。

4) HID プロファイルクライアントモードは RL78G1D BLE ハードウェアモジュールによってサポートされていません。この結果、HID プロファイルの BLE フレームワーク実装でも HID プロファイルクライアントモードはサポートされません。RL78G1D 対応の BLE フレームワークを使用するアプリケーションは、クライアントモードで HID プロファイルを使用できません。

5) 複数のスレーブ BLE デバイスを RL78G1D BLE モジュールに接続することはできません。

BLE フレームワークは、以下のボードでのみテストされます。

- DK-S7G2 バージョン 3.1
- DK-S3A7 バージョン 2.0

- PK-S5D9 バージョン 1.0
- TB-S5D5 バージョン 1.1
- TB-S3A6 バージョン 1.0
- DK-S128 バージョン 1.1
- DK-S124 バージョン 3.1

モジュールの制限事項については、SSP の最新のリリースノートを参照してください。

4.1.11.4 アプリケーションへの BLE フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して BLE フレームワークモジュールをアプリケーションに組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。BLE フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。(BLE フレームワークモジュールのデフォルト名は g_sf_ble0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

BLE フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_ble0 RL78G1D BLE GAP and GATT on sf_ble_rl78g1d	Threads	New Stack> Framework> Networking> BLE > RL78G1D BLE GAP and GATT on sf_ble_rl78g1d
g_sf_ble_onboard_profile0 On-Board Profile on RL78G1D BLE Framework	Threads	New Stack> Framework> Networking> BLE > On-Board Profile on RL78G1D BLE Framework
g_sf_ble0 EM9301 BLE GAP and GATT on sf_ble_em9301	Threads	New Stack> Framework> Networking> BLE> EM9301 BLE GAP and GATT on sf_ble_em9301

次の図に示すように、sf_ble の BLE フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

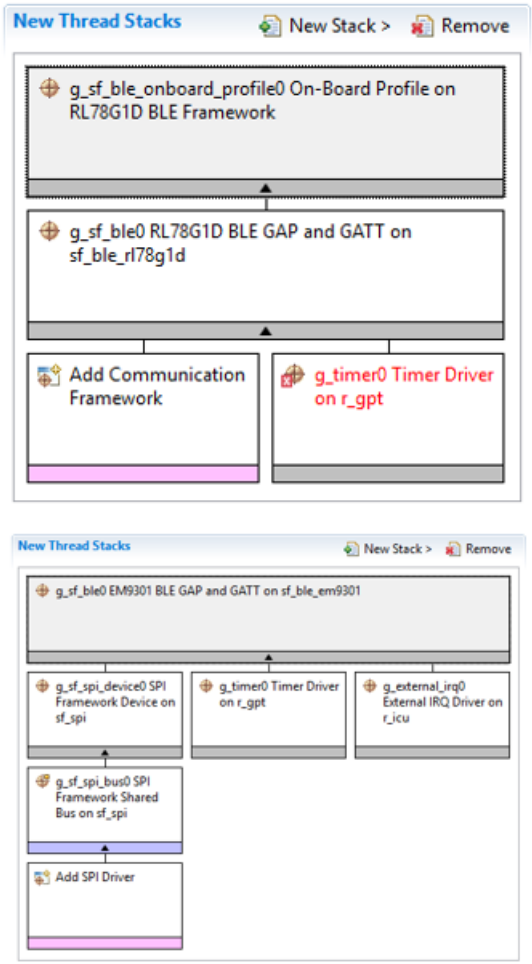


図 47:BLE フレームワークモジュールのスタック

上記のスタックで、[Add Communication Framework] ブロックはまだ設定されていません。通信フレームワークには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。一般的なオプションは以下のとおりです。

- sf_comms_telnet 通信フレームワーク
- sf_el_ux_comms_v2 通信フレームワーク
- sf_uart_comms 通信フレームワーク
- sf_el_nx_comms 通信フレームワーク [非推奨]
- sf_el_ux_comms 通信フレームワーク [非推奨]

4.1.11.5 BLE フレームワーク モジュールの設定

ユーザーは必要な動作に合わせて BLE フレームワークモジュールを構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます（ブロックが赤で強調表示される）。こ

れらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成表の設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

RL78G1D BLE フレームワーク上のオンボードプロファイルの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Heart Rate Profile	Enabled, Disabled Default: Enabled	Heart rate profile selection
Alert Notification Profile	Enabled, Disabled Default: Disabled	Alert notification profile selection
Blood Pressure Profile	Enabled, Disabled Default: Disabled	Blood pressure profile selection
Find Me Profile	Enabled, Disabled Default: Disabled	Find me profile selection
HID Over GATT Profile	Enabled, Disabled Default: Disabled	HID gatt profile selection
Health Thermometer Profile	Enabled, Disabled Default: Disabled	Health thermometer profile selection

ISDE Property	Value	Description
Phone Status Alert Profile	Enabled, Disabled Default: Disabled	Phone alert profile selection
Proximity Profile	Enabled, Disabled Default: Disabled	Proximity proximity profile selection
Scan Parameter Profile	Enabled, Disabled Default: Disabled	Scan parameter profile selection
Time Profile	Enabled, Disabled Default: Disabled	Time time profile selection
Name	g_sf_ble_onboard_profile0	Module name
Name of generated initialization function	sf_ble_rl78g1d_onboard_profile_init0	Name of generated initialization function
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_ble_rl78g1d 上の BLE GAP および GATT の構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_sf_ble0	Module name

ISDE Property	Value	Description
Bluetooth Device Address (Restart Board after first run to see changed Address)	{0x0, 0x0, 0x0, 0x0, 0x0}	Bluetooth device address selection
Address Type	Public Address, Random Address Default: Public Address	Address type selection
Scan Interval	48	Scan interval selection
Scan Window	48	Scan window selection
Maximum Connection Interval	40	Maximum connection interval selection
Connection Slave Latency	0	Connection slave latency selection
Supervision Timeout	80	Supervision timeout selection
BLE Driver Thread Priority	1	BLE Driver thread priority selection
BLE Serial Thread Priority	1	BLE Serial thread priority selection
Name of generated initialization function	sf_ble_rl78g1d_init0	Name of generated initialization function
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、ローレベルモジュールの [properties] セクションのすべての設定を示します。

r_gpt 上のタイマドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_timer0	Module name.
Channel	0	Channel selection.
Mode	Periodic	Warning: One Shot functionality is not available in the GPT hardware, so it is implemented in software by stopping the timer in the ISR called when the period expires. For this reason, ISR's must be enabled for one-shot mode even if the callback is unused.
Duty Cycle Range (only applicable in PWM mode)	Shortest: 2 PCLK, Longest (Period-1) PCLK/Shortest: 1 PCLK, Longest: (Period-2) PCLK Default: Shortest 2 PCLK, Longest: (Period-1) PCLK	Select the duty cycle range. Due to hardware limitations, one PCLK is added before the output pin toggles after the duty cycle is reached. This extra clock cycle is added to the ON time (if Shortest: 2 PCLK is selected) or the OFF time (if Shortest: 1 PCLK is selected) based on this configuration.
Period Value	10	See Timer Period Calculation
Period Unit	Milliseconds	See Timer Period Calculation
Duty Cycle Value	50	Duty cycle value selection
Duty Cycle Unit	Unit Raw Counts	Duty cycle unit selection
Auto Start	True	Set to true to start the timer after configuring or false to leave the timer stopped until timer_api_t::start is called.

ISDE Property	Value	Description
GTIOCA Output Enabled	False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCA Stop Level	Pin Level Low	Controls output pin level when the timer is stopped.
GTIOCB Output Enabled	False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCB Stop Level	Pin Level Low	Controls output pin level when the timer is stopped.
Callback	RBLE_Timer_cb	A user callback function can be registered in timer_api_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the timer period elapses. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Overflow Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Overflow interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

EM9301 BLE GAP および GATT の構成

sf_ble_em9301 上の EM9301 BLE GAP および GATT の構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
No of Services in Custom Profile	1	Select the number of services in custom profile.
No of Characteristics in Custom Profile	1	Select the number of characteristics in custom profile.
Name	g_sf_ble0	Module name.
Bluetooth Device Address	{0x0,0x0,0x0,0x0,0x0,0x0}	Select the bluetooth device address.
Address Type	Public Address, Random Address Default: Public Address	Select the address type.
Scan Interval	48	Select the scan interval.
Scan Window	48	Select the scan window.
Maximum Connection Interval	40	Select the maximum connection interval.
BLE Driver Stack Thread Priority	1	Select the BLE driver stack thread priority.
BLE Driver Receive Thread Priority	1	Select the BLE driver receive thread priority.
Chip Select Pin (CS)	IOPORT_PORT_05_PIN_07	Select the chip select pin.
MISO Pin	IOPORT_PORT_05_PIN_10	Select the MISO pin.

ISDE Property	Value	Description
MOSI Pin	IOPORT_PORT_05_PIN_09	Select the MOSI pin.
Interrupt Pin	IOPORT_PORT_00_PIN_05	Select the interrupt pin.
Reset Pin	IOPORT_PORT_10_PIN_05	Select the reset pin.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。デフォルト以外の設定が望ましい場合もあります。たとえば、異なる画面サイズの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、ローレベルモジュールの [properties] セクションのすべての設定を示します。

sf_spi 上の SPI フレームワークデバイスの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_sf_spi_device0	Module name.
Clock Phase	Data sampling on odd edge, data variation on even edge/Data sampling on even edge, data variation on odd edge Default: Data sampling on odd edge, data variation on even edge	Select the clock phase.

ISDE Property	Value	Description
Clock Polarity	Low when idle, High when idle Default: Low when idle	Select the clock polarity.
Chip Select Port	0	Select GPIO port used for the chip select.
Chip Select Pin	0	Select GPIO pin used for the chip select.
Chip Select Active Level	Low	Polarity of the Chip Select signal, active High or Low

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_spi 上の SPI フレームワーク共有バスの構成設定

ISDE Property	Value	Description
Name	g_sf_spi_bus0	Give a name to identify the bus. This bus name is used in the Framework configuration to link the SPI peripheral to a bus.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_rspi 上の SPI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_spi0	Module name.

ISDE Property	Value	Description
Channel	0	SCI or SPI Channel number to which the device has been connected.
Operating Mode	Master, Slave Default: Master	Configure as a Master or Slave device. Note: Current version of SSP supports only SPI Master mode.
Clock Phase	Data sampling on odd edge, data variation on even edge	Data sampling on odd or even clock edge.
Clock Polarity	Low when idle	Clock level when idle.
Mode Fault Error	Enable, Disable Default: Disable	Indicates Mode fault error (master/slave conflict) flag.
Bit Order	MSB First, LSB First Default: MSB First	Select transmit order MSB/LSB first.
Bitrate	500000	Transmission or reception rate. Bits per second.
Callback	NULL	Optional Callback function pointer.
SPI Mode	SPI Operation, Clock Synchronous operation Default: SPI Operation	Select spi or clock syn mode operation.
Slave Select Polarity(SSL0)	Active Low, Active High Default: Active Low	Select SSL0 signal polarity.
Slave Select Polarity(SSL1)	Active Low, Active High Default: Active Low	Select SSL1 signal polarity.

ISDE Property	Value	Description
Slave Select Polarity(SSL2)	Active Low, Active High Default: Active Low	Select SSL2 signal polarity.
Slave Select Polarity(SSL3)	Active Low, Active High Default: Active Low	Select SSL3 signal polarity.
Select Loopback1	Normal, Inverted Default: Normal	Select loopback1.
Select Loopback2	Normal, Inverted Default: Normal	Select loopback2.
Enable MOSI Idle State	Enable, Disable Default: Disable	Select MOSI idle fixed value and selection.
MOSI Idle State	MOSI Low, MOSI High Default: MOSI Low	Select mosi idle fixed value and selection.
Enable Parity	Enable, Disable Default: Disable	Enable/disable parity.
Parity Mode	Parity Odd, Parity Even Default: Parity Odd	Select parity.
Select SSL(Slave Select)	SSL0, SSL1, SSL2, SSL3 Default: SSL0	Select which slave to use; 0-SSL0; 1-SSL1; 2-SSL2; 3-SSL3.

ISDE Property	Value	Description
Select SSL Level After Transfer	SSL Level Keep, SSL Level Do Not Keep Default: SSL Level Do Not Keep	Select SSL level after transfer completion; 0-negate; 1-keep.
Clock Delay Enable	Clock Delay Enable, Clock Delay Disable Default: Clock Delay Disable	Clock delay enable selection.
Clock Delay Count	Clock Delay (1:8) RSPCK Default: Clock Delay 1 RSPCK	Clock delay count selection.
SSL Negation Delay Enable	Negation Delay Enable, Negation Delay Disable Default: Negation Delay Disable	SSL negation delay enable selection.
Negation Delay Count	Negation Delay (1:8) RSPCK Default: Negation Delay 1 RSPCK	Negation delay count selection.
Next Access Delay Enable	Next Access Delay Enable, Next Access Delay Disable Default: Next Access Delay Disable	Next access delay enable selection.
Next Access Delay Count	Next Access Delay (1:8) RSPCK Default: Next Access Delay 1 RSPCK	Next access delay count selection.

ISDE Property	Value	Description
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Receive interrupt priority selection.
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit interrupt priority selection.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit end interrupt priority selection.
Error Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Error interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SPI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection

ISDE Property	Value	Description
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte, 2 Bytes, 4 Bytes Default: 2 Bytes	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode)	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SPI0 TXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SPI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte, 2 Bytes, 4 Bytes Default: 2 Bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode)	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SPI0 RXI	Activation source selection
Auto Enable	False	Auto enable selection

ISDE Property	Value	Description
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sci_spi 上の SPI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_spi0	Module name.
Channel	0	SCI or SPI Channel number to which the device has been connected.
Operating Mode	Master, Slave Default: Master	Configure as a Master or Slave device. Note: Current version of SSP supports only SPI Master mode.
Clock Phase	Data sampling on odd edge, data variation on even edge	Data sampling on odd or even clock edge.
Clock Polarity	Low when idle	Clock level when idle.

ISDE Property	Value	Description
Mode Fault Error	Enable, Disable Default: Disable	Indicates Mode fault error (master/slave conflict) flag.
Bit Order	MSB First, LSB First Default: MSB First	Select transmit order MSB/LSB first
Bitrate	100000	Transmission or reception rate. Bits per second.
Bit Rate Modulation Enable	Enable, Disable Default: Enable	Bitrate Modulation Function enable or disable
Callback	NULL	Optional Call back function pointer.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Bitrate Modulation Function enable or disable. Note: This is applicable only for SCI SPI.
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit interrupt priority selection.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit end interrupt priority selection
Error Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Error interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

Event SCI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency seleciton
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection

ISDE Property	Value	Description
Activation Source (Must enable IRQ)	Event SCI0 TXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection

ISDE Property	Value	Description
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SCI0 RXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_gpt のタイマドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.

ISDE Property	Value	Description
Name	g_timer0	Module name.
Channel	0	Channel selection.
Mode	Periodic	Warning: One Shot functionality is not available in the GPT hardware, so it is implemented in software by stopping the timer in the ISR called when the period expires. For this reason, ISR's must be enabled for one-shot mode even if the callback is unused.
Duty Cycle Range (only applicable in PWM mode)	Shortest: 2 PCLK, Longest (Period-1) PCLK/Shortest: 1 PCLK, Longest: (Period-2) PCLK Default: Shortest 2 PCLK, Longest: (Period-1) PCLK	Select the duty cycle range. Due to hardware limitations, one PCLK is added before the output pin toggles after the duty cycle is reached. This extra clock cycle is added to the ON time (if Shortest: 2 PCLK is selected) or the OFF time (if Shortest: 1 PCLK is selected) based on this configuration.
Period Value	10	See Timer Period Calculation
Period Unit	Milliseconds	See Timer Period Calculation
Duty Cycle Value	50	Duty cycle value selection
Duty Cycle Unit	Unit Raw Counts	Duty cycle unit selection
Auto Start	True	Set to true to start the timer after configuring or false to leave the timer stopped until is called.
GTIOCA Output Enabled	False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.

ISDE Property	Value	Description
GTIOCA Stop Level	Pin Level Low	Controls output pin level when the timer is stopped.
GTIOCB Output Enabled	False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCB Stop Level	Pin Level Low	Controls output pin level when the timer is stopped.
Callback	NULL	A user callback function can be registered in . If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the timer period elapses. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Overflow Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Overflow interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_icu 上の外部 IRQ ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter checking setting enables or disables the addition of parameter checking code.
Name	g_external_irq0	Module name.
Channel	0	Specifies the hardware IRQ channel used.
Trigger	Rising	Selection for trigger event mode
Digital Filtering	Disabled	Digital filter enable/disable.
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCKL/64	Sets noise filter sampling period.
Interrupt enabled after initialization	True	Determines if the interrupt is enabled immediately after initialization.
Callback	BLE0_IRQ_CALLBACK	A user callback function can be registered in . If this callback function is provided, it is called from the interrupt service routine (ISR) each time the IRQn triggers. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.

ISDE Property	Value	Description
Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注 : モジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

BLE フレームワークモジュールのクロック構成

BLE フレームワークモジュールは、ローレベルモジュールによって指定されるクロックを使用します。

BLE フレームワークモジュールのピン構成

BLE フレームワークモジュールは、ローレベルモジュールによって指定されるピンを使用します。

4.1.11.6 アプリケーションでの BLE フレームワークモジュールの使用

一般的なアプリケーションで Bluetooth Low Energy フレームワークモジュールを使用する際の手順は次のとおりです。

1) 初期化（open API を使用した、コールバック関数の登録、プロビジョニング、アドバタイズメント）
2) sf_ble_api_t::provisionSet API を使用した BLE モジュールのプロビジョニング
3) sf_ble_api_t::scan API を使用したアドバタイズメントのスキャン
4) sf_ble_api_t::connect API を使用したリモート BLE デバイスへの接続
5) sf_ble_onboard_profile_api_t::onbpEnable API を使用したオンボードプロファイルの有効化
6) sf_ble_onboard_profile_api_t::onbpClientWriteCCCD API を使用した通知の有効化
7) sf_ble_onboard_profile_api_t::onbpServerWriteData API を使用したプロファイルの特性のライト
8) sf_ble_onboard_profile_api_t::onbpClientReadChar API を使用したプロファイルの特性のリード
9) sf_ble_onboard_profile_api_t::onbpDisable API を使用したプロファイルの無効化
10) sf_ble_api_t::disconnect API を使用したリモート BLE デバイスからの切断
11) sf_ble_api_t::close API を使用した BLE モジュールのクローズ

次の図は、一般的な手順を示した通常の動作フロー図です。

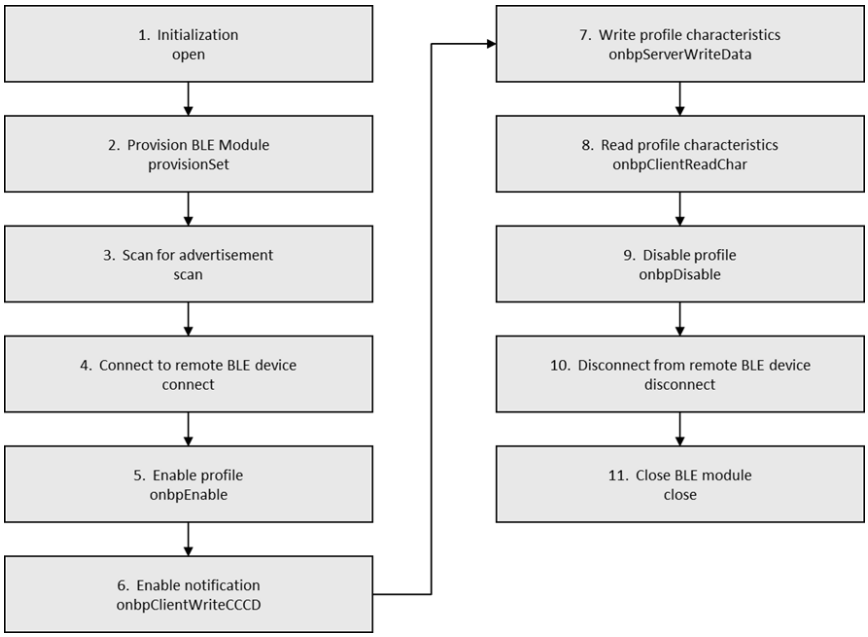


図 48:BLE モジュールの初期化を使用したアプリケーションの制御フロー

4.1.12 セルラーフレームワーク

セルラーフレームワークモジュールによって、SSP でのセルラーモデム統合のためにハイレベルのアプリケーションレイヤーインタフェースが提供されます。セルラーフレームワークによって、さまざまなベンダのセルラーモデムをアプリケーションで使用するための共通インタフェースが提供されます。

セルラーフレームワークは、データ通信のためにセルラーネットワークのプロビジョニング、構成、通信を行う一連の API を提供します。セルラーフレームワークはコンソールフレームワークを使用し、AT コマンドを使用してシリアルインタフェースを介してセルラーモデムと通信します。セルラーフレームワークは、NetX ™によって提供される PPP WAN プロトコルを利用して、データ通信用のシリアルインタフェース上でシリアルデータパイプを作成します。TCP/IP を使用するデータ通信は、NetX アプリケーションプロトコル、ソケット、あるいは IoT プロトコル（MQTT など）を使用して、この Wide Area Network（WAN）リンク上で確立できます。

また、セルラーフレームワークではフレームワークレベルのソケット API も提供されます。これは、特定のセルラーハードウェアモジュール内のオンチップ（セルラーハードウェアモジュール内）の TCP/IP スタックや、さらにはソケット API を使用するネットワークの TCP/IP リンクと通信するためのものです。

4.1.12.1 セルラーフレームワークモジュールの特長

- 以下を使用する接続がサポートされます。
  - セルラーモジュールに存在するオンチップスタック対応の BSD ソケットインタフェース
  - NSAL インタフェースを使用する Synergy MCU（ホスト）上の NetX スタック
- ネットワークスタックに対応する API の共通セットと、さまざまなセルラーハードウェアモジュール対応の汎用インタフェースがサポートされます。

- 汎用 API と抽象化を利用して、セルラーハードウェアモジュール用に開発されたアプリケーションを、別のセルラーハードウェアモジュールと連携するために簡単に移行できます。
- サポートされるセルラーモデム :
  - NimbeLink CAT3（NL-SW-LTE-TSVG）Verizon-US
  - NimbeLink CAT3（NL-SW-LTE-TEUG）India and Europe
  - NimbeLink CAT1（NL-SW-LTE-GELS3-B and NL-SW-LTE-GELS3-C）Verizon-US
  - Quectel BG96（CAT M1、NB-IoT、および GPRS）

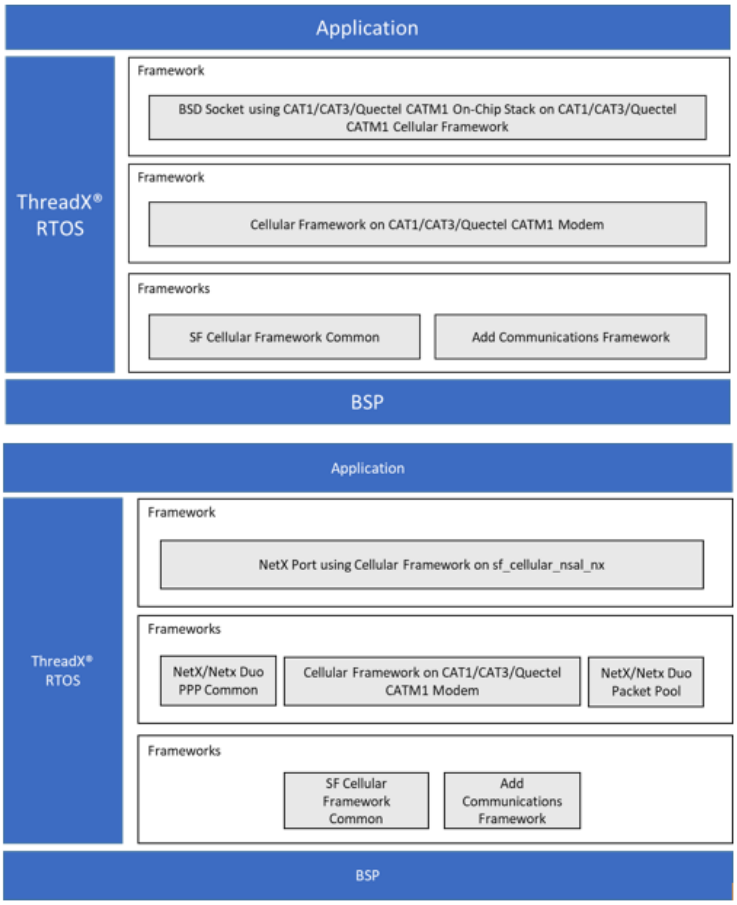


図 49: セルラーフレームワークモジュールのブロック図

注 :3 つの BSD ソケットオンチップスタックのいずれも、その下に 3 つのセルラーフレームワークモジュールのいずれかを実装できます。同様に、sf_cellular_nsal_nx 上の NetX ポートは、その下に 3 つのセルラーフレームワークモジュールのいずれかを実装できます。

4.1.12.2 セルラーフレームワークモジュールの API の概要

セルラーフレームワークモジュールは、関連する各モジュールの API を定義します。この後で各 API の動作について説明します。

注：セルラーフレームワークモジュールの API の詳しい説明が、Renesas Web サイトの「Cellular Application Note」にあります。上部の検索バーで「R30AN0311」を検索すると、アプリケーションの注意事項とアプリケーションプロジェクトが検索結果に表示されます。

セルラーフレームワークモジュールの API

セルラーフレームワークモジュールの API 要約

Function Name	Example API Call and Description
open	g_sf_cellular0.p_api->open (g_sf_cellular0.p_ctrl, g_sf_cellular0.p_cfg); This API function initializes and enables the Cellular module. The open function returns the Cellular control structure, uniquely identifying the instance of the Cellular framework. The Cellular framework open function accepts the Cellular module configuration as an argument, with the following parameters: - Operator Selection Mode (enumeration) - Operator Name Format (enumeration) - Operator Name (string) - Preferred Operator List (array of structures) - Time zone update policy (enumeration)
close	g_sf_cellular0.p_api->close (g_sf_cellular0.p_ctrl); This API un-initializes the Cellular module and disables it. It takes the Cellular control structure as an argument.

Function Name	Example API Call and Description
infoGet	g_sf_cellular0.p_api->infoGet (g_sf_cellular0.p_ctrl, p_cellular_info); This API takes the Cellular control structure as an argument and returns the following information obtained from the Cellular module: - Chipset/driver information (string) - Manufacturer Name (string) - Firmware version (string) - IMEI Number (string) - RSSI value (unsigned integer 16 bits) - Bit Error Rate (unsigned integer 16 bits)
statisticsGet	g_sf_cellular0.p_api->statisticsGet (g_sf_cellular0.p_ctrl, p_stats); This API gets the data statistics from the Cellular module. It takes the Cellular control structure as an argument and returns the following statistics: - Received packets (unsigned integer 32 bits) - Transmitted packets (unsigned integer 32 bits) - Transmit packet errors (unsigned integer 32 bits)
transmit	g_sf_cellular0.p_api->transmit (g_sf_cellular0.p_ctrl, p_buffer, length); This API sends the data/packet out. It takes the Cellular control structure, the data buffer and the data buffer length as an argument. The Cellular framework transmit function passes the data buffer to the PPP driver for transmission.

Function Name	Example API Call and Description
provisioningGet	g_sf_cellular0.p_api->provisioningGet (g_sf_cellular0.p_ctrl, p_cellular_provisioninfo); This API takes the Cellular control structure as an argument and returns the following parameters: - Authentication type(enumeration) - Username (string) - Password (string) - APN Name(string) - PDP Context ID (integer) - PDP Context Type (enumeration) - Airplane mode (enumeration)
provisioningSet	g_sf_cellular0.p_api->provisioningSet (g_sf_cellular0.p_ctrl, p_cellular_provisioninfo); This API sets the authentication credential information. It takes the Cellular control structure and the following parameters as argument to provision the Cellular module: - Authentication type(enumeration) - Username (string) - Password (string) - APN Name(string) - PDP Context ID (integer) - PDP Context Type (enumeration) - Airplane mode (enumeration)
networkConnect	g_sf_cellular0.p_api-> networkConnect (g_sf_cellular0.p_ctrl); This API establishes the Network connection over Cellular Network using which application can communicate to remote host with the help of Network stack. It takes the Cellular control structure as an argument.

Function Name	Example API Call and Description
networkDisconnect	g_sf_cellular0.p_api->networkDisconnect (g_sf_cellular0.p_ctrl); This API terminates the Network connection established using networkConnect API. It takes the Cellular control structure as an argument.
simPinSet	g_sf_cellular0.p_api->simPinSet (g_sf_cellular0.p_ctrl, p_pin); This API allows the application/user to change the PIN required to register on Cellular Network. It takes the Cellular control structure, old PIN and New PIN as arguments.
simLock	g_sf_cellular0.p_api->simLock (g_sf_cellular0.p_ctrl, p_pin); This API locks the SIM. It takes the Cellular control structure and Sim PIN as arguments.
simUnlock	g_sf_cellular0.p_api->simUnlock (g_sf_cellular0.p_ctrl, p_pin); This API unlocks the SIM. It takes the Cellular control structure and Sim PIN as arguments.
simIDGet	g_sf_cellular0.p_api->simIDGet (g_sf_cellular0.p_ctrl, p_sim_id); This API reads the Sim ID from the Cellular module. It takes the Cellular control structure as argument and returns the SIM ID read from the Cellular module.
commandSend	g_sf_cellular0.p_api->commandSend (g_sf_cellular0.p_ctrl, p_input_at_commmand, p_output, timeout); Send AT command directly to Cellular Modem

Function Name	Example API Call and Description
networkStatusGet	g_sf_cellular0.p_api->networkStatusGet (g_sf_cellular0.p_ctrl, p_status); This API gets Cellular Module Network Status information. It takes the Cellular control structure as argument and returns following parameters: - Country code (integer) - Operator code (integer) - RSSI (integer) - Cell ID (string) - IMSI (string) - Operator name (string) - Service Domain (integer) - Active Band (integer)
fotaCheck	g_sf_cellular0.p_api->fotaCheck (g_sf_cellular0.p_ctrl, p_fota_check); This API checks for available firmware Upgrade. It takes the Cellular control structure and fota check specific structure as argument.
fotaStart	g_sf_cellular0.p_api->fotaStart (g_sf_cellular0.p_ctrl, p_fota_start); This API starts the firmware upgrade. It takes the Cellular control structure and fota start specific structure as argument.
fotaStop	g_sf_cellular0.p_api->fotaStop (g_sf_cellular0.p_ctrl, p_fota_stop); This API stops the firmware upgrade. It takes the Cellular control structure and fota stop specific structure as argument.
versionGet	g_sf_cellular0.p_api->versionGet (p_version); This API retrieves the version for the API using the version pointer.

Function Name	Example API Call and Description
reset	g_sf_cellular0.p_api->reset (g_sf_cellular0.p_ctrl, reset_type); Reset the cellular hardware module.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

セルラーソケットフレームワークモジュールの API 要約

これらの API を使用して、オンチップネットワークスタックを使用するときに Wi-Fi モジュールを構成できます。これは、インタフェースの IP アドレスの構成と、DHCP サーバーの起動と停止（AP モードで構成されている場合）に役立ちます。

オンチップネットワークスタックサポート Wi-Fi フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_cellular_socket0.p_api->open (g_sf_cellular_socket0.p_ctrl, g_sf_cellular_socket0.p_cfg); This API calls the Cellular Framework's lower level Open () API to Initialize the Cellular Device Driver.
close	g_sf_cellular_socket0.p_api->close (g_sf_cellular_socket0.p_ctrl); This API calls the Cellular Framework's lower level Close () API to close the Cellular Device Driver.
versionGet	g_sf_cellular_socket0.p_api->versionGet (p_version); This API retrieves the version for the API using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

これらの API 関数は、アプリケーションがソケットを使用したデータ転送を実行するために使用できます。これには BSD API 関数に準拠するソケット API 関数が含まれます。

- socket
- close
- connect
- send
- recv
- sendto
- recvfrom
- select

セルラーフレームワークのエラーコード

次の表に、セルラーフレームワーク固有のエラーコードを示します。これらのエラーコードは ssp_err_t. に含まれます。

セルラーフレームワークのエラーコード

Error Codes	Description
SSP_SUCCESS	Call successful
SSP_ERR_CELLULAR_CONFIG_FAILED	Configuration failed
SSP_ERR_CELLULAR_INIT_FAILED	Initialization failed
SSP_ERR_CELLULAR_TRANSMIT_FAILED	Transmit failed
SSP_ERR_CELLULAR_FW_UPTODATE	Up to date
SSP_ERR_CELLULAR_FW_UPGRADE_FAILED	Upgrade failed
SSP_ERR_CELLULAR_FAILED	General failure
SSP_ERR_CELLULAR_INVALID_STATE	Invalid state

4.1.12.3 セルラーフレームワークモジュールの動作の概要

セルラーフレームワークによって、アプリケーションを変更する必要がないまま、さまざまなベンダのセルラーハードウェアモジュールとシームレスに通信するための汎用インタフェースが提供されます。このフレームワークは、主に、ネットワークスタックとさまざまなセルラーハードウェアモジュールのインタフェースになる共通セットの API で構成

されます。このセクションでは、セルラーフレームワークの基本ブロックと重要な機能について説明します。これに基づいて、セルラーフレームワークを使用して目的のセルラーアプリケーションを開発できるかどうかを判断できます。

注 : セルラーフレームワークモジュールの動作に関する詳しい説明が、Renesas Web サイトの「Cellular Application Note」にあります。画面上部の検索バーで「R30AN0311」を検索すると、アプリケーションの注意事項とアプリケーションプロジェクトが検索結果に表示されます。

提供されている API と抽象化により、セルラーハードウェアモジュール用に開発されたアプリケーションを、簡単に移植して別のセルラーハードウェアモジュールを使用できます。ネットワーキングスタック NetX も、ネットワークソフトウェア抽象化レイヤー（NSAL）を使用してこのフレームワークに統合されます。

Synergy セルラーフレームワークは以下の論理ブロックで構成されています。

- Synergy セルラーフレームワークアプリケーションインタフェース
- NetX TCP/IP スタック用のネットワークスタック抽象化レイヤー（NSAL）
- セルラーデバイスドライバー（セルラーチップセットとやりとりするための AT コマンドインタフェース）
- オンチップネットワーキングスタックをサポートするセルラーハードウェアモジュールのインタフェースになる BSD ソケット互換 API
- Synergy ソフトウェアパッケージ（SSP）HAL インタフェース

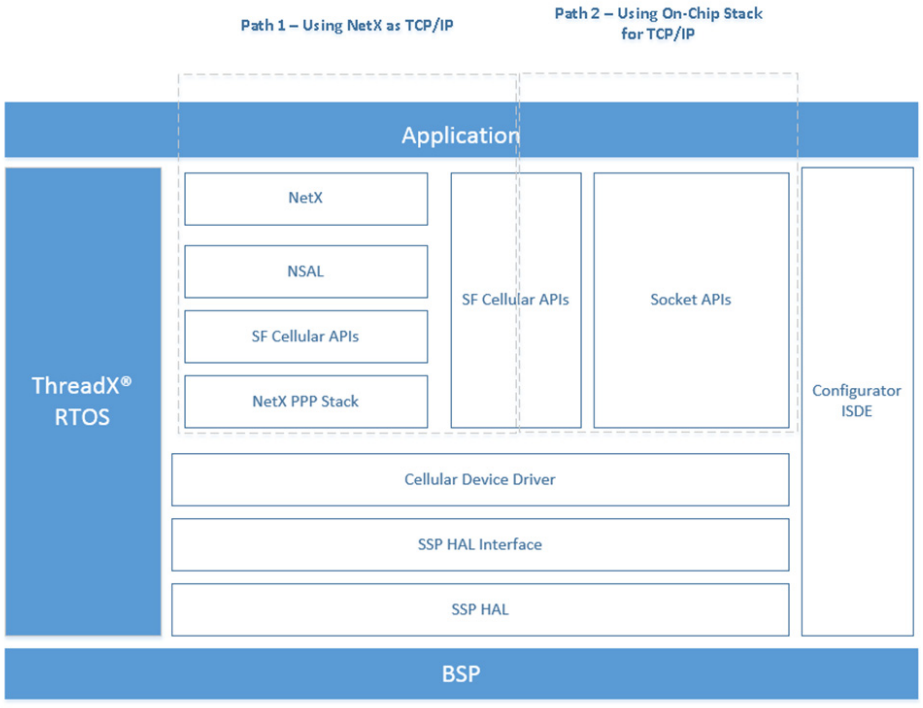


図 50: セルラーフレームワークモジュールのアプリケーション概要

セルラーフレームワークによってアプリケーションにインタフェースの共通セットが提供され、アプリケーションはセルラーハードウェアモジュールの構成、プロビジョニング、通信を行います。ユーザーは、このような汎用インタフェースを使用し、Synergy MCU を使用してセルラーベースのアプリケーションを開発できます。セルラーハード

ウェアモジュールには、一連の 3GPP 規格で指定されているさまざまな構成パラメータがあります。個別のデバイスドライバーまたはセルラーチップセットやモジュール（あるいは両方）ですべての構成パラメータがサポートされるとは限りません。ネットワークオペレータ、アクセスポイント名（APN）、セキュリティ資格証明は、このモジュールが機能するために最低限必要です。

ネットワークスタック抽象化レイヤー

セルラーフレームワークは、ネットワークスタック抽象化レイヤー（NSAL）を提供します。NSAL は、WAN リンク上のデータ通信に使用される PPP スタックを使用して NetX とセルラードライバーに接続するレイヤーです。

ソケットインタフェースレイヤー

セルラーフレームワークによってアプリケーションにソケットレベル API が提供され、アプリケーションはセルラーハードウェアモジュールに存在するオンチップネットワークスタックとやりとりします。これには、セルラーハードウェアモジュールまたはドライバーが、オンチップネットワークスタックとソケットインタフェースをサポートする必要があります。アプリケーションがこれらの API を使用するとき、セルラーハードウェアモジュールに存在するオンチップネットワークスタックを使用し、NSAL または NetX とソケット API は使用しません。また、Synergy MCU グループで実行しているネットワークスタックも使用しません。

PPP スタック

ポイントツーポイントプロトコル（PPP）はデータ通信で一般的に使用される WAN プロトコルです。PPP スタックは、SSP で使用可能な NetX スタックの一部です。NSAL は、PPP スタックを利用して、シリアルインタフェースを介してセルラーサービスプロバイダのネットワークと通信します。PPP 構成には、PAP/CHAP などの認証方法が用意されています。これらの認証メカニズムは、リンクの確立ではオプションです。NSAL は、フレームワーク API を使用して、セルラーハードウェアモジュールとの間でデータの送受信を行います。NSAL を使用すると、ネットワークスタックに関して変更せずにセルラーデバイスドライバーを再利用することもできます。

セルラーデバイスドライバー（セルラーチップセットとやりとりするための AT コマンドインタフェース）

セルラーフレームワークは AT コマンドセットを使用し、シリアルドライバーを使用してセルラーモデムとやりとりします。モデムとのやりとりに使用されるシリアルインタフェースは UART です。フレームワークのデフォルトで使用される UART 速度は、最大 115200 ビット / 秒です。

セルラーフレームワークモジュールの動作に関する重要な注意事項と制限事項

セルラーモデムで使用可能なサポートに応じて、通信のために 2 つのパスでアプリケーションをこのフレームワークで使用できます。TCP/IP スタックをホストエンドで使用するオプションが提供されるモジュールや、セルラーモデムそのものに存在する TCP/IP スタックを使用するオプションが提供されるモジュールがあります。場合によっては、セルラーハードウェアモジュールによって両方が提供されます。ホスト TCP/IP スタック（NetX）が使用されるとき、NetX、NSAL、PPP の論理レイヤーはアーキテクチャ図のように使用されます。オンチップスタックが使用されると、ソケット API が、セルラーモデムに存在する TCP/IP スタックと通信するために使用されます。ただし、ユーザーは同時に両方を使用できません。

次の制御フロー図に示すように、セルラーモデムの要件に応じてユーザーが提供する構成を使用した初期化の際には、nx_ip_create がコールされます。この API は、内部で NSAL ドライバーのエントリ関数（リンクレベルの初期化を行い、セルラーハードウェアモジュールを初期化する）を呼び出します。また、これは、モジュールをプロビジョニングし、PPP インタフェースを使用してネットワーク接続を確立します。

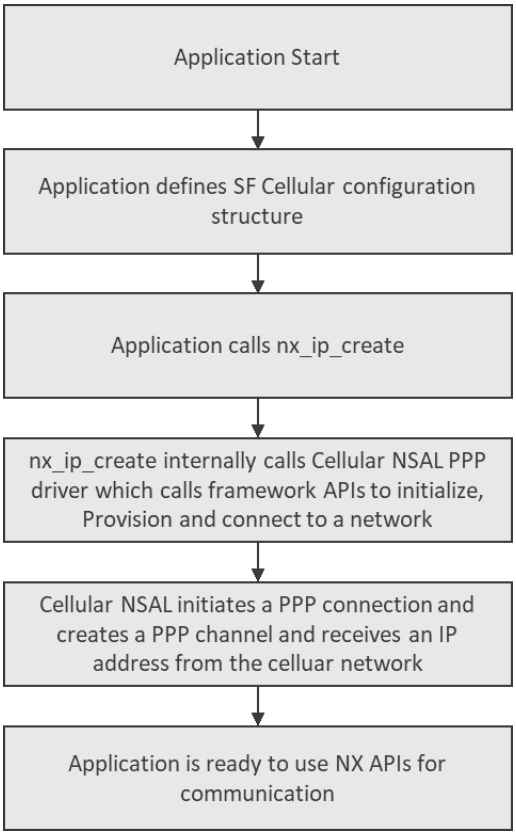


図 51:セルラーフレームワークモジュールの初期化シーケンス

セルラーモデムのプロビジョニング中には、制御構造体とユーザー構成構造体が引数として渡されます。プロビジョニングで使用される各ユーザー引数は、認証、APN、ユーザー名、パスワードです。

次のフロー図には、セルラーソケットインタフェースでのオンチップスタックパスの使用に関するフローが示されます。

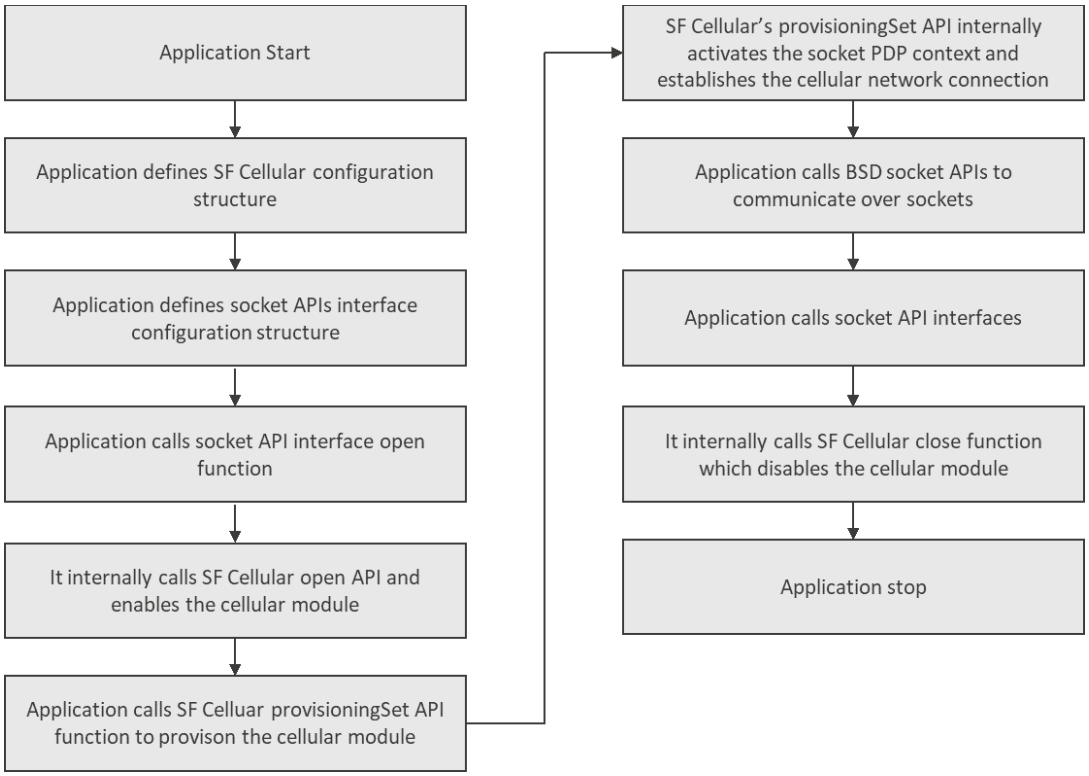


図 52: セルラーフレームワークモジュールのソケットインタフェース

次のフロー図には、パケット送信が NetX アプリケーションで使用する手順のシーケンスが示されます。

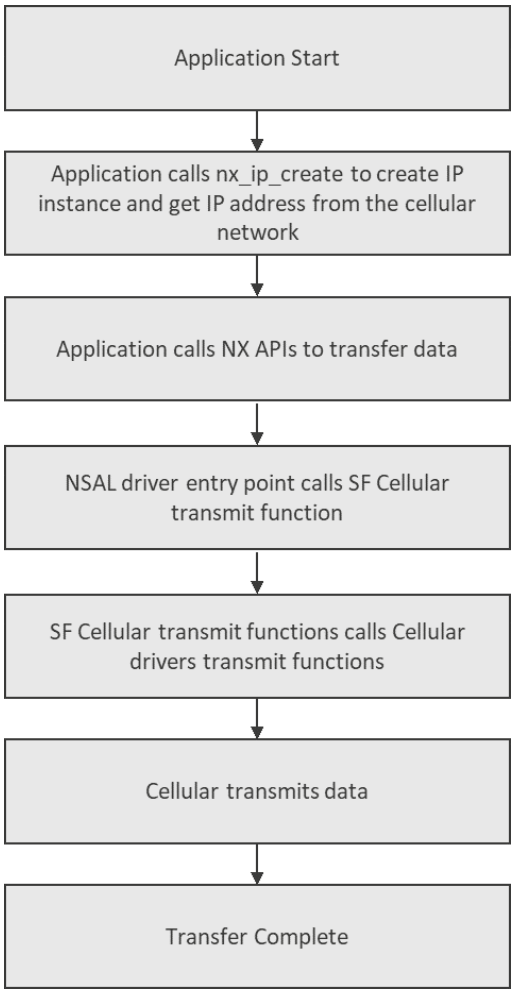


図 53: セルラーフレームワークモジュールのパケット送信シーケンス

次のフロー図には、NetX を使用したセルラーフレームワークのパケット受信が示されます。受信のケースでは、データがシリアルインタフェースで受信されるとき、処理スレッドがコールバック関数をトリガし、コールバック関数がデータを処理し、さらに処理するために NetX レイヤーに送信します。

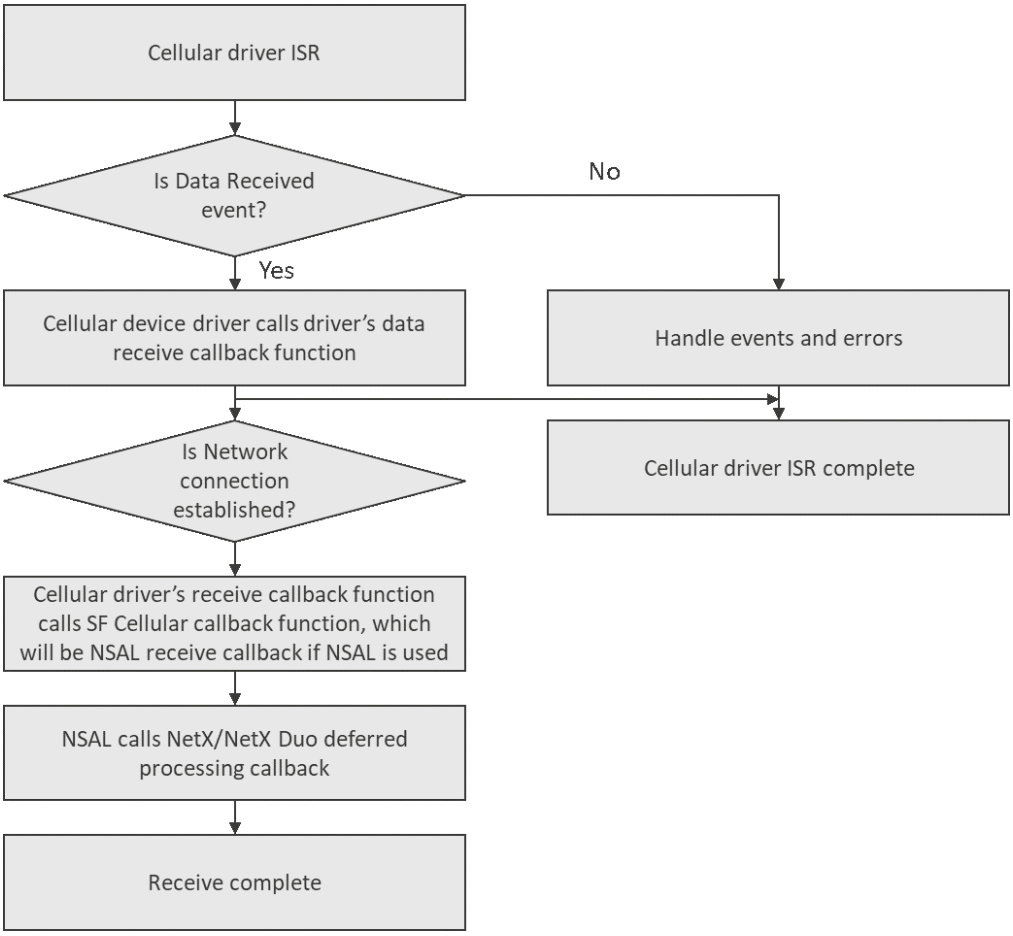


図 54: セルラーフレームワークモジュールのパケット受信シーケンス

- 現在のフレームワークは、以下のセルラーモジュールに対応します。
  - NimbeLink CAT3（NL-SW-LTE-TSVG）Verizon-US
  - NimbeLink CAT3（NL-SW-LTE-TEUG）India and Europe
  - NimbeLink CAT1（NL-SW-LTE-GELS3-B および NL-SW-LTE-GELS3-C）
  - Quectel BG96（CAT M1、NB-IoT、および GPRS）
- 無線によるファームウェアアップグレード（FOTA）は、いずれのモジュールのセルラーフレームワークでもサポートされません。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.12.4 アプリケーションへのセルラーフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにセルラーフレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。セルラーフレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。(コンソールフレームワークモジュールのデフォルト名は g_sf_cellular0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

セルラーフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_cellular_socket0 BSD Socket using CAT1/CAT3/Quectel CATM1 On-Chip Stack on CAT1/CAT3/Quectel CATM1 Cellular Framework	Threads	New Stack> Framework> Networking> Cellular > BSD Socket using CAT1/CAT3/Quectel CATM1 On-Chip Stack on CAT1/CAT3/Quectel CATM1 Cellular Framework
g_sf_cellular_0 Cellular Framework on CAT1/CAT3/Quectel CATM1 Modem	Threads	New Stack> Framework> Networking> Cellular > Cellular Framework on CAT1/CAT3/Quectel CATM1 Modem
g_sf_el_nx0 NetX Port using Cellular Framework on sf_cellular_nsal_nx	Threads	New Stack> Framework> Networking> Cellular > NetX Port using Cellular Framework on sf_cellular_nsal_nx

次の図に示すように、sf_cellular のセルラーフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

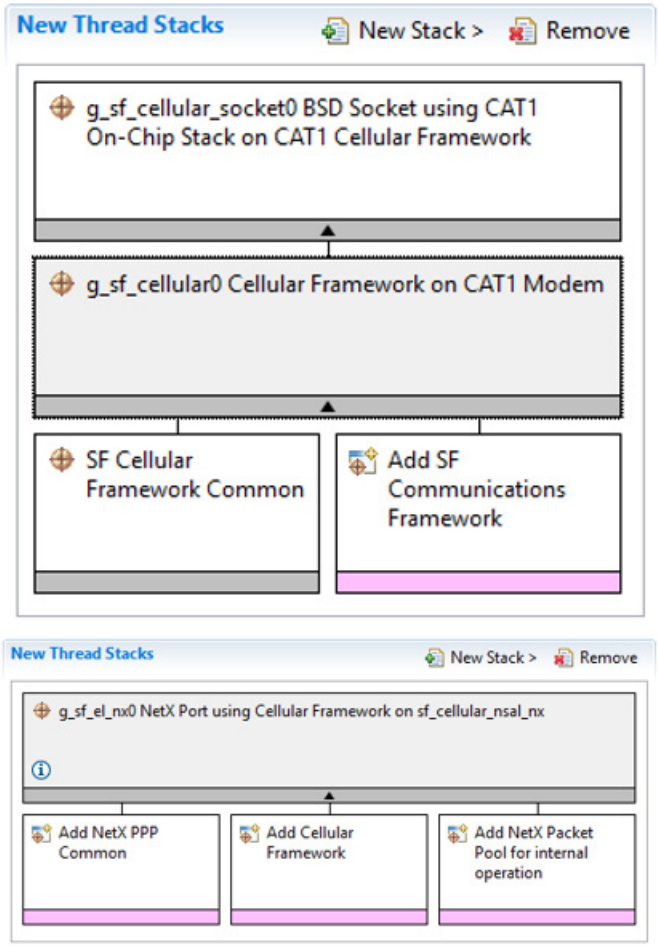


図 55: セルラーフレームワークモジュールのスタック

上記のスタックで、[Add SF Communications Framework] ブロックはまだ設定されていません。通信フレームワークには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。一般的なオプションは以下のとおりです。

- sf_comms_telnet 通信フレームワーク
- sf_el_ux_comms_v2 通信フレームワーク
- sf_uart_comms 通信フレームワーク
- sf_el_nx_comms 通信フレームワーク [非推奨]
- sf_el_ux_comms 通信フレームワーク [非推奨]

4.1.12.5 セルラーフレームワークモジュールの構成

ユーザーは必要な動作に合わせてセルラーフレームワークモジュールを構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます（ブロックが赤で強調表示される）。

これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成表の設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

CAT1 セルラーフレームワークでの CAT1 オンチップスタックを使用した BSD ソケットの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_cellular_socket0	Module name.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

CAT3 セルラーフレームワークでの CAT3 オンチップスタックを使用した BSD ソケットの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_cellular_socket0	Module name.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

Quectel CATM1 セルラーフレームワークでの Quectel CATM1 オンチップスタックを使用した BSD ソケットの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_cellular_socket0	Module name.
Name of generated initialization function	sf_cellular_qctcatm1_socket_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

CAT1 モデム上のセルラーフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking
On-Chip Stack Support	Enabled, Disabled Default: Disabled	On-chip stack support selection
AT Command Retry Count	5	Modem selection
Name	g_sf_cellular0	Module name
SIM Pin (Used to Unlock SIM)	1111	SIM Pin selection
SIM PUK Pin (Used to Unlock SIM)	12345678	SIM PUK Pin selection

ISDE Property	Value	Description
Number of Preferred Operator	0	Number of preferred operator selection
Preferred Operator 1 Name	40422	Preferred operator 1 name selection
Preferred Operator 1 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 1 name format selection
Preferred Operator 2 Name	40424	Preferred operator 2 name selection
Preferred Operator 2 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 2 name format selection
Preferred Operator 3 Name	40422	Preferred operator 3 name selection
Preferred Operator 3 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 3 name format selection
Preferred Operator 4 Name	40424	Preferred operator 4 name selection
Preferred Operator 4 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 4 name format selection
Preferred Operator 5 Name	40422	Preferred operator 5 name selection
Preferred Operator 5 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 5 name format selection
Operator Select Mode	Auto, Manual, Deregister, Manual Fallback Default: Auto	Operator select mode selection

ISDE Property	Value	Description
Operator Name (Manual Mode Selection)	40422	Operator name selection
Operator Name Format (Manual Mode Selection)	Long, Short, Numeric Default: Numeric	Operator name format selection
Time Zone Update Policy	Enabled, Disabled Default: Enabled	Time zone update policy selection
Receive Data Callback	sf_cellular_nsal_recv_callback	Receive data callback selection
Provisioning Callback	celr_prov_callback	Provisioning callback selection
Circular Queue Size in Bytes	256	Circular queue size selection
SF Communications Framework Thread Stack Size	512	SF communications framework thread stack size selection
Numerical priority of SF Communication Framework Thread. Legal values range from 0 through (TX_MAX_PRIORITIES-1), where a value of 0 represents the hightest priority.	5	Numerical priority of SF communication framework thread selection
Cellular Module Reset IO Pin	IOPORT_PORT_10_PIN_05	Cellular module reset IO pin selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

CAT3 モデム上のセルラーフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking
On-Chip Stack Support	Enabled, Disabled Default: Disabled	On-chip stack support selection
Modem	TEUG, TSVG Default: TEUG	Modem selection
AT Retry Command Count	5	AT command retry command count selection
Name	g_sf_cellular0	Module name
SIM Pin (Used to Unlock SIM)	1111	SIM Pin selection
SIM PUK Pin (Used to Unlock SIM)	12345678	SIM PUK Pin selection
Number of Preferred Operator	0	Number of preferred operator selection
Preferred Operator 1 Name	40422	Preferred operator 1 name selection
Preferred Operator 1 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 1 name format selection
Preferred Operator 2 Name	40424	Preferred operator 2 name selection
Preferred Operator 2 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 2 name format selection

ISDE Property	Value	Description
Preferred Operator 3 Name	40422	Preferred operator 3 name selection
Preferred Operator 3 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 3 name format selection
Preferred Operator 4 Name	40424	Preferred operator 4 name selection
Preferred Operator 4 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 4 name format selection
Preferred Operator 5 Name	40422	Preferred operator 5 name selection
Preferred Operator 5 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 5 name format selection
Operator Select Mode	Auto, Manual, Deregister, Manual Fallback Default: Auto	Operator select mode selection
Operator Name (Manual Mode Selection)	40422	Operator name selection
Operator Name Format (Manual Mode Selection)	Long, Short, Numeric Default: Numeric	Operator name format selection
Time Zone Update Policy	Enabled, Disabled Default: Enabled	Time zone update policy selection
Receive Data Callback	sf_cellular_nsal_recv_callback	Receive data callback selection
Provisioning Callback	celr_prov_callback	Provisioning callback selection

ISDE Property	Value	Description
Circular Queue Size in Bytes	256	Circular queue size selection
SF Communications Framework Thread Stack Size	512	SF communications framework thread stack size selection
Numerical priority of SF Communication Framework Thread. Legal values range from 0 through (TX_MAX_PRIORITIES-1), where a value of 0 represents the hightest priority.	5	Numerical priority of SF communication framework thread selection
Cellular Module Reset IO Pin	IOPORT_PORT_10_PIN_05	Cellular module reset IO pin selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

Quectel CATM1 モデム上のセルラーフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking
On-Chip Stack Support	Enabled, Disabled Default: Disabled	On-chip stack support selection
AT Command Retry Count	5	Modem selection
Name	g_sf_cellular0	Module name
SIM Pin (Used to Unlock SIM)	1111	SIM Pin selection
SIM PUK Pin (Used to Unlock SIM)	12345678	SIM PUK Pin selection

ISDE Property	Value	Description
Number of Preferred Operator	0	Number of preferred operator selection
Preferred Operator 1 Name	40422	Preferred operator 1 name selection
Preferred Operator 1 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 1 name format selection
Preferred Operator 2 Name	40424	Preferred operator 2 name selection
Preferred Operator 2 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 2 name format selection
Preferred Operator 3 Name	40422	Preferred operator 3 name selection
Preferred Operator 3 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 3 name format selection
Preferred Operator 4 Name	40424	Preferred operator 4 name selection
Preferred Operator 4 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 4 name format selection
Preferred Operator 5 Name	40422	Preferred operator 5 name selection
Preferred Operator 5 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 5 name format selection
Operator Select Mode	Auto, Manual, Deregister, Manual Fallback Default: Auto	Operator select mode selection

ISDE Property	Value	Description
Operator Name (Manaul Mode Selection)	40422	Operator name selection
Operator Name Format (Manual Mode Selection)	Long, Short, Numeric Default: Numeric	Operator name format selection
Time Zone Update Policy	Enabled, Disabled Default: Enabled	Time zone update policy selection
Receive Data Callback	sf_cellular_nsal_recv_callback	Receive data callback selection
Provisioning Callback	celr_prov_callback	Provisioning callback selection
Circular Queue Size in Bytes	256	Circular queue size selection
SF Communications Framework Thread Stack Size	512	SF communications framework thread stack size selection
Numerical priority of SF Communication Framework Thread. Legal values range from 0 through (TX_MAX_PRIORITIES-1), where a value of 0 represents the hightest priority.	5	Numerical priority of SF communication framework thread selection
Cellular Module Reset IO Pin	IOPORT_PORT_01_PIN_06	Cellular module reset IO pin selection

ISDE Property	Value	Description
Network Scan Sequence	LTE cat.M1-> LTE Cat.NB1-> GSM, LTE Cat.M1-> GSM-> LTE Cat.NB1, GSM-> LTE Cat.NB1-> LTE Cat.M1, GSM-> LTE Cat.M1-> LTE Cat.NB1, LTE Cat.NB1 -> LTE Cat.M1 -> GSM, LTE Cat.NB1 -> GSM -> LTE Cat.M1 Default: LTE cat.M1-> LTE Cat.NB1-> GSM	Network scan sequence selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

セルラーフレームワーク共通インスタンスの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

セルラーフレームワークを使用した NetX ポートの構成

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、ローレベルモジュールの [properties] セクションのすべての設定を示します。

sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポートの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking
Name	g_sf_el_nx0	Module name
PPP Stack Size in Bytes	2048	PPP stack size selection
Name	g_nx_ppp0	Module name
Numerical priority of PPP Thread	3	Specify the PPP thread priority. The priority must be lower than IP Helper thread. Legal values range from 0 through (TX_MAX_PRIORITIES-1), where a value of 0 represents the highest priority.
Authentication Method	None, PAP, CHAP Default: None	Authentication method selection
Invalid Packet Handler Callback	NULL	Invalid packet handler callback selection
Link Down Callback	ppp_link_down_callback	Link down callback selection
Link Up Callback	ppp_link_up_callback	Link up callback selection
PAP Login Callback	NULL	A user callback function can be provided.
PAP Verify Login Callback	NULL	A user callback function can be provided.
Get Challenges Values Callback	NULL	A user callback function can be provided.
Get Responder Values Callback	NULL	A user callback function can be provided.

ISDE Property	Value	Description
Get Verification Callback	NULL	A user callback function can be provided.
Local IPv4 Address (use commas for separation)	0,0,0,0	Local IPv4 address selection
Peer IPv4 Address (use commas for separation)	0,0,0,0	Peer IPv4 address selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX PPP 共通インスタンスの構成設定

ISDE Property	Value	Description
Name	g_nx_ppp_common0	Module name

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size (bytes)	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initilization function selection
Auto initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

nx 上の NetX 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注 : モジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

セルラーフレームワークモジュールのクロック構成

セルラーフレームワークモジュールは、選択された特定のローレベルモジュールに必要なクロックを使用します。

セルラーフレームワークモジュールのピン構成

セルラーフレームワークモジュールは、選択されたローレベルモジュールに依存する入力ピンと出力ピンを使用します。

4.1.12.6 アプリケーションでのセルラーフレームワークモジュールの使用

一般的なセルラーアプリケーションでは、動作の多くは、構成済みのセルラーモジュールスタックに基づいて SSP によって実行されます。コンフィギュレータを使用して IP インスタンスがセルラーフレームワークと共に追加されると、PPP スタックがフレームワークの一部として含まれます。また、NSAL およびセルラーデバイスドライバーコードも含まれます。セルラーの初期化は、自動生成コードの役割です。

データ接続と ICMP への ping は、ユーザーが追加するコードの役割です。ユーザーが入力したパブリック IP アドレスに ping 要求を送信し、ping レスポンスを確認する責任があります。PPP リンクダウン、PP リンクアップ、セルラープロビジョニングのためにコールバック関数を実装することができます。

一般的なアプリケーションでセルラーフレームワークを使用する際の手順は次のとおりです。

1) 生成されたコードによる初期化
2) nx_ip_status_check API を使用したリンク確立の待機
3) nx_icmp_ping API を使用したネットワークへの ping
4) nx_event_flags_set API を使用したイベントフラグのチェック

次の図は、一般的な手順を示した通常の動作フロー図です。

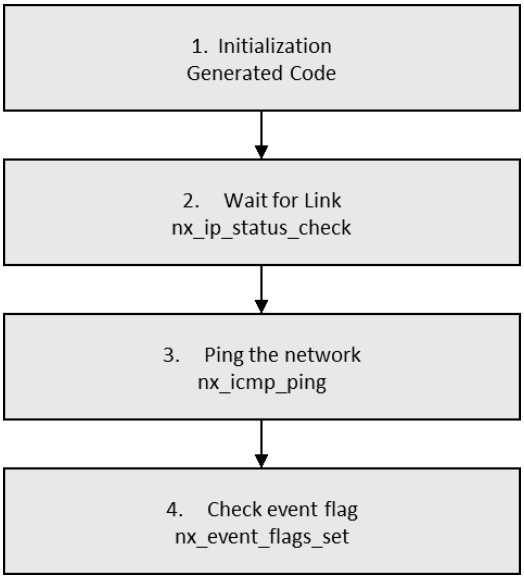


図 56: セルラーモジュールの初期化を使用したアプリケーションの制御フロー

4.1.13 NX 通信フレームワーク

NX 通信フレームワークは、通信フレームワークアプリケーション向けにハイレベルの API を実装し、Synergy MCU 上のイーサネットペリフェラルを使用します。

注 : このモジュールは、今後のリリースで削除される可能性があることを示すために SSP Rev 1.5.0 で非推奨になりました。このモジュールを使用している既存のプロジェクトは sf_comms_telnet フレームワークに移行し、すべての新規の設計では sf_comms_telnet を使用することを強く推奨します。

4.1.13.1 NetX 通信フレームワークモジュールの特長

- イーサネット上でハイレベルの接続性がサポートされますが、アプリケーションコードを変更せずに UART や USB 接続に簡単に変更できます。
- 専用アクセスのためのチャネルロックをサポートします。
- Thread 対応実装はミューテックスとイベントフラグを内部で使用します。

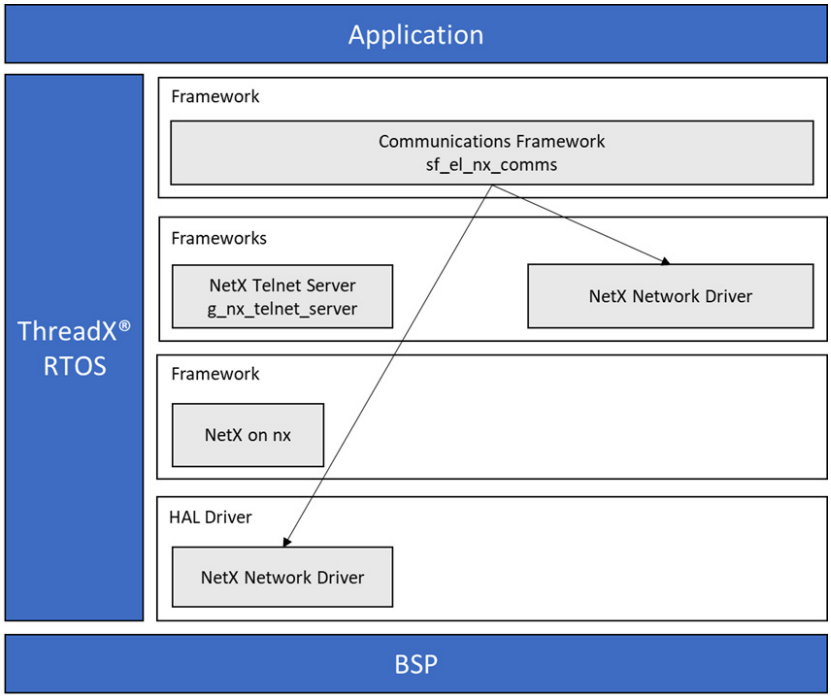


図 57:NetX 通信フレームワークモジュールのブロック図

4.1.13.2 NetX 通信フレームワークモジュールの API の概要

NetX 通信フレームワークモジュールは、USB 接続を介したオープン、クローズ、リード、ライトの API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX 通信フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_comms0.p_api->open(g_sf_comms0.p_ctrl, g_sf_comms0.p_cfg); Initialize communications driver.
close	g_sf_comms0.p_api->close(g_sf_comms0.p_ctrl); Clean up communications driver.

Function Name	Example API Call and Description
read	g_sf_comms0.p_api->read(g_sf_comms0.p_ctrl, &destination, bytes, timeout); Read data from communications driver. This call will return after the number of bytes requested is read or if a timeout occurs while waiting for access to the driver.
write	g_sf_comms0.p_api->write(g_sf_comms0.p_ctrl, &source, bytes, timeout); Write data to communications driver. This call will return after all bytes are written or if a timeout occurs while waiting for access to the driver.
lock	g_sf_comms0.p_api->lock(g_sf_comms0.p_ctrl, lock_type, timeout); Lock the communications driver. Reserve exclusive access to the communications driver.
unlock	g_sf_comms0.p_api->unlock(g_sf_comms0.p_ctrl, lock_type); Unlock the communications driver. Release exclusive access to the communications driver.
versionGet	g_sf_comms0.p_api->version(&version); Store the driver version in the provided version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Channel opened successfully.
SSP_ERR_IN_USE	Channel already in use.
SSP_ERR_ASSERTION	Pointer to UART control block or configuration structure is NULL.
SSP_ERR_INTERNAL	Internal error occurs.
SSP_ERR_TIMEOUT	Timeout error.
SSP_ERR_NOT_OPEN	Module is not opened.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.13.3 NetX 通信フレームワークモジュールの動作の概要

NetX 通信フレームワークによって、イーサネットポートを介した使用しやすい接続が提供されます。ハイレベル API 関数は、他の接続実装（UART や USB など）と互換性があるため、アプリケーションコードを変更せずに簡単に実装を切り替えることができます。

このモジュールは、sf_comms_api_t::open API を使用したモジュールのオープンと、sf_comms_api_t::close API を使用したモジュールのクローズをサポートします。リードは sf_comms_api_t::read API によって実装され、ライトは sf_comms_api_t::write API によって実装されます。コールされた API 関数が動作している間だけモジュールをロックする sf_comms_api_t::read や sf_comms_api_t::write とは異なり、sf_comms_api_t::lock API は、同じモジュールインスタンスに対して sf_comms_api_t::unlock API がコールされるまでモジュールをロックします。基礎となる NetX ドライバーでは、ユーザーが IP アドレスとネットワークマスクを構成し、イーサネットチャネルを選ぶこともできます。

NetX 通信フレームワークモジュールの動作に関する重要な注意事項と制限事項

NetX 通信フレームワークモジュールの動作に関する重要な注意事項

- イーサネットペリフェラルでは、MCU の機能に応じて RMII または MII を使用できます。

NetX 通信フレームワークモジュールの動作に関する制限事項

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.13.4 アプリケーションへの NetX 通信フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX 通信フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各

手順を管理する方法を確認してください。NetX 通信フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。(NetX 通信フレームワークモジュールのデフォルト名は g_sf_comms0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

NetX 通信フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_comms0 NetX Communications Framework on sf_el_nx_comms	Threads	New Stack> Framework> Connectivity> NetX Communications Framework on sf_el_nx_comms

次の図に示すように、sf_el_nx_comms の NetX 通信フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

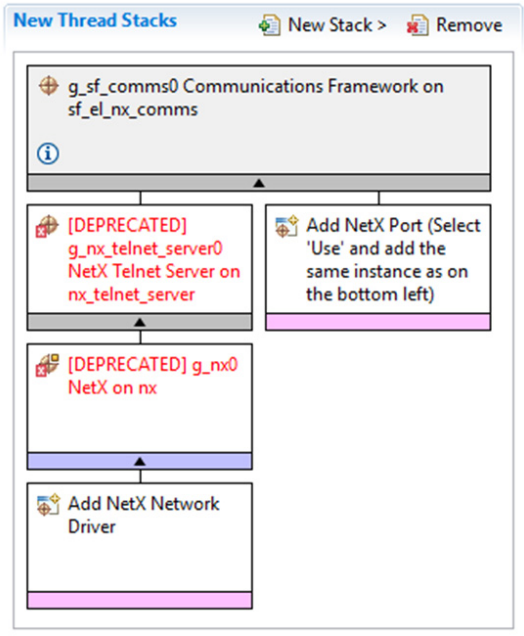


図 58: NetX 通信フレームワークモジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.1.13.5 NetX 通信フレームワークモジュールの構成

ユーザーは必要な動作に合わせて、NetX 通信フレームワークモジュールを構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成表の設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_el_nx_comms 上の NetX 通信フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_comms0	Module name.
Channel	0	Underlying channel used by Ethernet driver.
IP Address Byte 1	192	IP Address Byte 1 selection
IP Address Byte 2	168	IP Address Byte 2 selection
IP Address Byte 3	0	IP Address Byte 3 selection
IP Address Byte 4	0	IP Address Byte 4 selection

ISDE Property	Value	Description
Subnet Mask Byte 1	255	Subnet Mask Byte 1 selection
Subnet Mask Byte 2	255	Subnet Mask Byte 2 selection
Subnet Mask Byte 3	255	Subnet Mask Byte 3 selection
Subnet Mask Byte 4	0	Subnet Mask Byte 4 selection
Name of generated initialized function	sf_comms_init0	Name of generated initialized function
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

デフォルト以外の設定が望ましい場合もあります。たとえば、アプリケーションに応じた異なる IP アドレスの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

NetX 通信フレームワークのローレベルモジュールの構成

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、ローレベルモジュールの [properties] セクションのすべての設定を示します。

NetX Telnet サーバーの構成設定

ISDE Property	Value	Description
Name	g_nx_telnet_server0	Module name

ISDE Property	Value	Description
Show deprecation warning	Enabled, Disabled Default: Enabled	Show deprecation warning selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

nx 上の NetX モジュールの構成設定

ISDE Property	Value	Description
Name	g_nx0	Module name
Show deprecation warning	Enabled, Disabled Default: Enabled	Show deprecation warning selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注 : モジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

NetX 通信フレームワークモジュールのクロック構成

イーサネットペリフェラルモジュールは PCLKA をクロックソースとして使用します。これは、ISDE の [Clocks] 構成タブを使用して設定できます。

NetX 通信フレームワークモジュールのピン構成

イーサネットペリフェラルモジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はイーサネットピンの選択例を示しています。

ETHERC1 のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
Ethernet	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1:RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Pin Configuration Property	Value	Description
Pin Group Selection	Mixed, A only Default: A only	Select pin group.
Operating Mode	Disabled, Custom, RMII Default: Disabled	Select RMII for operation in RMMI mode.
REF50CK	P701	REF50 Clock.
TXD0	P700	TX D0 Pin.
TXD1	P406	TX D1 Pin.
TXD_EN	P405	TX D Enable Pin.
RXD0	P700	RX D0 Pin.
RXD1	P406	RX D1 Pin.
RXD_EN	P405	RX D Enable Pin.
CRS_DV	P705	CRS DV Pin.
MDC	P403	MDC Pin.
MDIO	P404	MD IO Pin.

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.1.13.6 アプリケーションでの NetX 通信フレームワークモジュールの使用

アプリケーションで NetX 通信フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_comms_api_t::open API を使用して、NX 通信フレームワークを初期化します。
2) sf_comms_api_t::lock API を使用して、連続通信のためにチャネルをロックします（必要な場合）。
3) sf_comms_api_t::read API を使用して、データを受信します。
4) sf_comms_api_t::write API を使用して、データを送信します。
5) sf_comms_api_t::unlock API コマンドを使用して連続通信用のチャネルのロックを解除します（必要な場合）。
6) sf_comms_api_t::close API を使用して、チャネルを閉じます。

これらの一般的な手順を、次の図の通常の動作フローに示します。

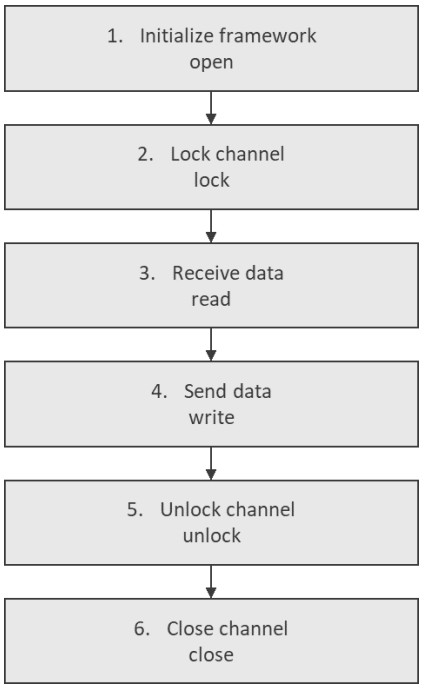


図 59:一般的な NetX 通信フレームワークモジュールアプリケーションのフロー図

4.1.14 Telnet 通信フレームワーク

NX 通信フレームワークは、通信フレームワークアプリケーションにハイレベルの API を提供し、Synergy MCU 上のイーサネットペリフェラルを使用します。

4.1.14.1 Telnet 通信フレームワークモジュールの特長

- イーサネット上でハイレベルの接続性がサポートされますが、API を変更せずに UART や USB 接続に簡単に変更できます。
- 専用アクセスのためのチャネルロックをサポートします。
- Thread 対応実装はミューテックスとイベントフラグを内部で使用します。

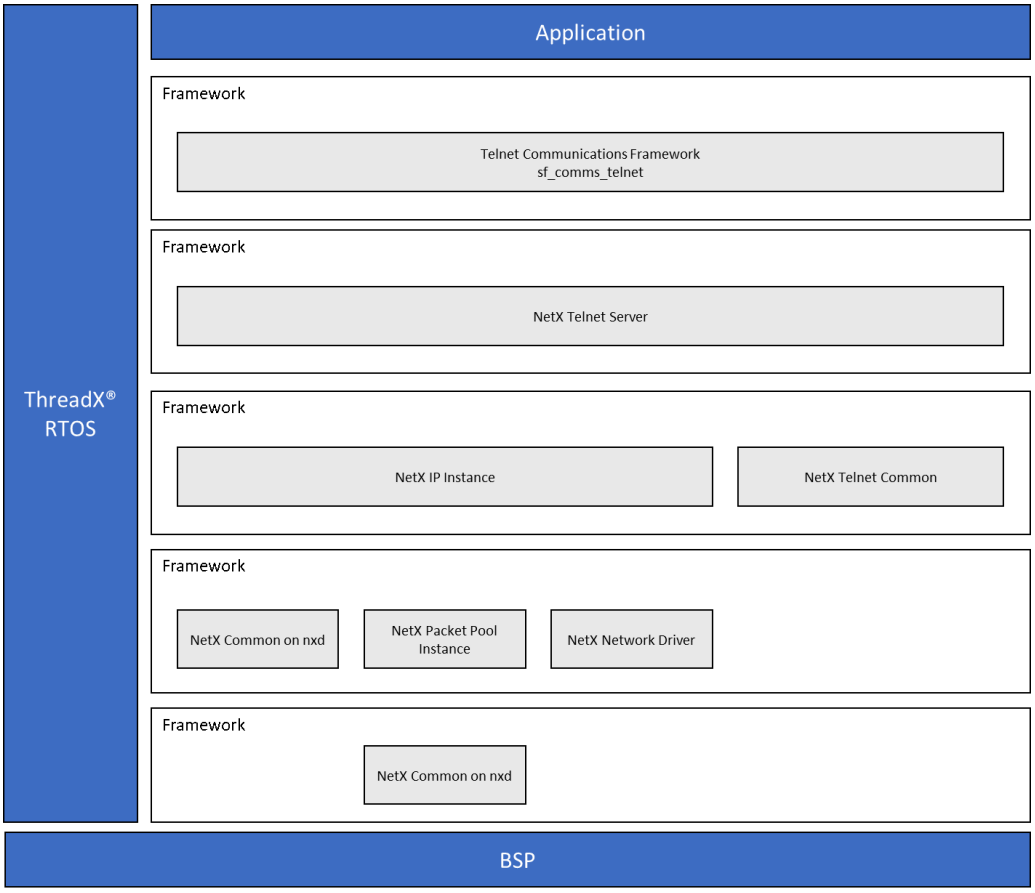


図 60:Telnet 通信フレームワークモジュールのブロック図

4.1.14.2 Telnet 通信フレームワークモジュールの API の概要

Telnet 通信フレームワークモジュールは、モジュールのオープン、リード、ライト、ロック、ロック解除、クローズを行う API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

Telnet 通信フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_comms_telnet0.p_api->open (g_sf_comms_telnet0.p_ctrl, g_sf_comms_telnet0.p_cfg); Initializes module.
read	g_sf_comms_telnet0.p_api->read (g_sf_comms_telnet0.p_ctrl, p_dest, bytes, timeout); Read a number of bytes of data into the destination.
write	g_ sf_comms_telnet0.p_api->write (g_sf_comms_telnet0.p_ctrl, P_src, bytes, timeout); Write a number of bytes from the source.
lock	g_sf_comms_telnet0.p_api->lock (g_sf_comms_telnet0.p_ctrl, locktype, timeout); Acquire lock type for the Telnet comms instance.
unlock	g_sf_comms_telnet0.p_api->unlock (g_sf_comms_telnet0.p_ctrl, locktype); Release the lock type for the Telnet comms inctance.

Function Name	Example API Call and Description
close	g_sf_comms_telnet0.p_api->close (g_sf_comms_telnet0.p_ctrl); Disconnect Telnet server and clean up resources.
versionGet	g_ sf_comms_telnet0.p_api->versionGet(&version); Gets version and stores it in provided version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_INTERNAL	An internal TheadX error has occurred.
SSP_ERR_NOT_OPEN	Unit is not open
SSP_ERR_ASSERTION	A parameter is NULL.
SSP_ERR_IN_USE	Peripheral is still running in another mode; perform Close first.
SSP_ERR_UNSUPPORTED	Command not supported.
SSP_ERR_OUT_OF_MEMORY	Can't allocate pool memory.
SSP_ERR_TIMEOUT	An event timed out.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.14.3 Telnet 通信フレームワークモジュールの動作の概要

NX 上の Telnet を使用する通信フレームワークによって、イーサネットポートを介した使用しやすい接続が提供されます。ハイレベル API は、他の接続プロトコル（UART や USB など）と互換性があるため、API を変更せずに簡単に実装を切り替えることができます。

フレームワークによってサポートされている動作には、sf_comms_api_t::open API を使用したモジュールの初期化と、sf_comms_api_t::close API を使用したモジュールのクローズがあります。通信のリードは sf_comms_api_t::read API によって実装され、通信のライトは sf_comms_api_t::write API によって実装されます。sf_comms_api_t::read と sf_comms_api_t::write は、コールされた API が動作している間だけモジュールをロックします。

sf_comms_api_t::lock API は、同じモジュールインスタンスに対して sf_comms_api_t::unlock API がコールされるまでモジュールをロックします。これにより、次の API 関数コールに進む前に処理を確実に完了することができます。

基礎となる NetX ドライバーでは、IP アドレス、ネットワークマスク、イーサネットチャネルの構成がサポートされます。別の通信実装（USB など）が使用される場合、そのインタフェースの定義には別のローレベルモジュール構成設定が使用されます。したがって、アプリケーションのコードは一切変更する必要がなく、構成設定のみを変更します。アプリケーションレベルでは同じ API コールが維持されます。

Telnet 通信フレームワークモジュールの動作に関する重要な注意事項と制限事項

Telnet 通信フレームワークモジュールの動作に関する重要な注意事項

- イーサネットペリフェラルでは、MCU の機能に応じて RMII または MII を使用できます。

Telnet 通信フレームワークモジュールの動作に関する制限事項

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.14.4 アプリケーションへの Telnet 通信フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに Telnet 通信フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

Telnet 通信フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。(Telnet 通信フレームワークモジュールのデフォルト名は sf_comms_telnet0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

Telnet 通信フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
sf_comms_telnet0 Telnet Communications Framework on sf_comms_telnet	Threads	New Stack> Framework> Connectivity> Telnet Communications Framework on sf_comms_telnet

次の図に示すように、sf_comms_telnet の Telnet 通信フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

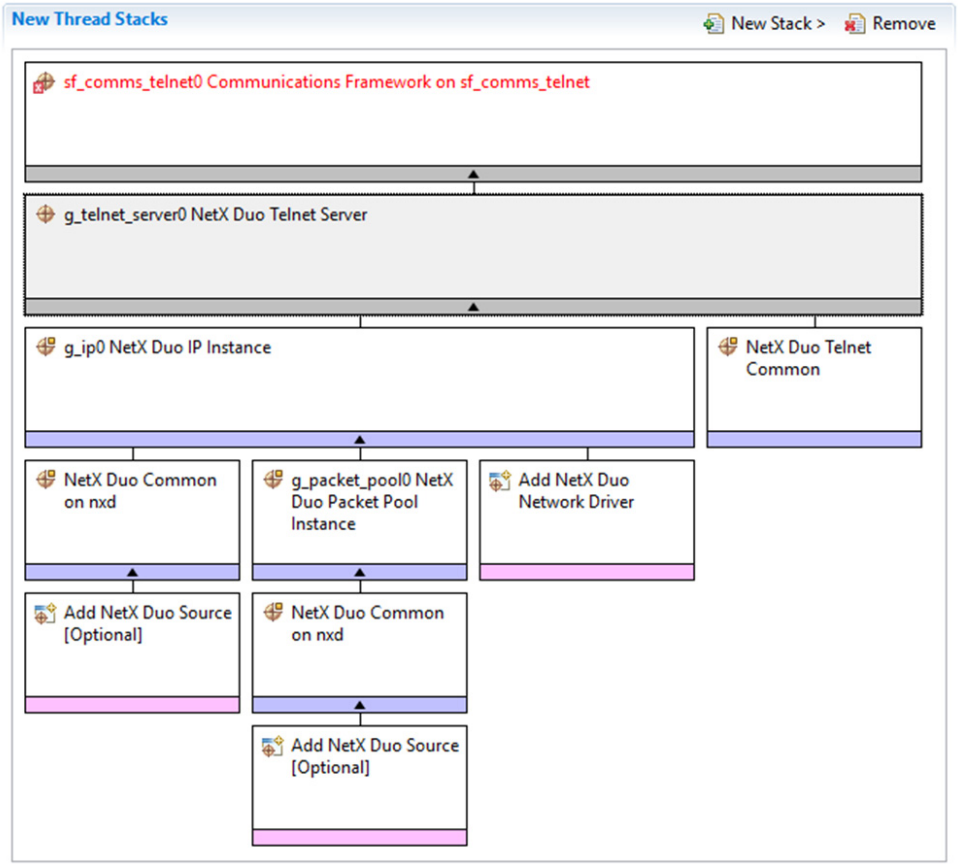


図 61:Telnet 通信フレームワークモジュールのスタック

4.1.14.5 Telnet 通信フレームワークモジュールの構成

ユーザーは必要な動作に合わせて、Telnet 通信フレームワークモジュールを構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成表の設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_comms_telnet 上の Telnet 通信フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking
Packet size in pool memory (bytes)	1536	Packet size in pool memory selection
Packets to allocate in pool memory (units)	5	Packets to allocate in pool memory selection
Timeout for internal options (ticks)	10	Timeout for internal options selection
Maximum number of instances	4	Maximum instances that can be open at any given time
Name	sf_comms_telnet0	Module name
Name of generated initialization function	sf_comms_telnet_init0	Name of generated initialization selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

Telnet 通信フレームワークのローレベルモジュールの構成

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

telnet_server 上の NetX/NetX Duo Telnet サーバーの構成設定

ISDE Property	Value	Description
Internal thread priority	16	Internal thread priority selection
Maximum clients to serve simultaneously	4	Maximum clients to serve simultaneously selection
Socket window size (bytes)	2048	Socket window size selection
Server time out (seconds)	10	Duration internal services will suspend for
Client inactivity timeout (seconds)	600	Client inactivity duration for disconnection
Timeout check period (seconds)	60	Client activity timeout check interval
Option negotiation	Enable, Disable Default: Enable	Option negotiation selection
Use application packet pool	Enable, Disable Default: Disable	Use application packet pool selection
Packet size in the pool (bytes)	300	Telnet Server only creates this packet pool if 'Option negotiation' is enabled
Total packet pool size (bytes)	2048	Telnet Server only creates this packet pool if NX_TELNET_SERVER_OPTION_DISABLE
Name	g_telnet_server0	Module name
Thread Stack Size (bytes)	2048	Thread stack size selection
Name of Client Connect Callback Function	NULL	Name of client connect callback function selection
Name of Receive Data Callback Function	NULL	Name of receive data callback function selection

ISDE Property	Value	Description
Name of Client Disconnect Callback Function	NULL	Name of client disconnect callback function selection
Name of generated initialization function	telnet_server_init0	Name of generated initialization function selection
Auto Initialization	Disable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	0,0,0,0	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection

ISDE Property	Value	Description
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection
Link status change callback	NULL	Link status change callback selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注 :** は、NetX Duo でのみ使用できるプロパティを表します。

NetX Telnet 共通の構成設定

ISDE Property	Value	Description
Type of Service for TCP requests	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost Default: Normal	Type of service UDP requests selection
Fragmentation option	Don't fragment, Fragment okay Default: Don't fragment	Fragment option selection
Server TCP port number	23	Server TCP port number selection
Time to live	128	Time to live selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

nxd 上の NetX/NetX Duo 共通の構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX ポート ETHER の構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking
Channel 0 Phy Reset Pin	IOPORT_PORT_09_PIN_03	Channel 0 Phy reset pin selection
Channel 0 MAC Address High Bits	0x00002E09	Channel 0 MAC address high bits selection
Channel 0 MAC Address Low Bits	0x0A0076C7	Channel 0 MAC address low bits selection
Channel 1 Phy Reset Pin	IOPORT_PORT_07_PIN_06	Channel 1 Phy reset pin selection
Channel 1 MAC Address High Bits	0x00002E09	Channel 1 MAC address high bits selection

ISDE Property	Value	Description
Channel 1 MAC Address Low Bits	0x0A0076C8	Channel 1 MAC address low bits selection
Number of Receive Buffer Descriptors	8	Number of receive buffer descriptors selection
Number of Transmit Buffer Descriptors	32	Number of transmit buffer descriptors selection
Ethernet Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Ethernet interrupt priority selection
Link status monitoring method	PHY Interrupt (Uses LINKSTA Pin), PHY Polling Default: PHY Polling	Link status monitoring method selection
Name	g_sf_el_nx	Module name
Channel	0	Channel selection
Callback	NULL	Callback selection
Unknown packet receive Callback	NULL	Unknown packet receive callback selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

Telnet 通信フレームワークモジュールのクロック構成

Telnet 通信フレームワークモジュールは、PCLKA をクロックソースとして使用するイーサネットペリフェラルを使用します。

実行時にクロック周波数を変更するには、CGC インタフェースを使用します。

Telnet 通信フレームワークモジュールのピン構成

Telnet 通信フレームワークモジュールを使用するには、ペリフェラルの入力と出力のポートピンを ISDE のピンコンフィギュレータで設定する必要があります。次の表では、ISDE [Configuration] ウィンドウでのピンの選択方法を示します。

Telnet 通信フレームワークモジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SSI	Pins	Select Peripherals > Peripherals>Connectivity:ETHERC> ETHERC0.RMI or ETHERC1.RMII

4.1.14.6 アプリケーションでの Telnet 通信フレームワークモジュールの使用

アプリケーションで Telnet 通信フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_comms_api_t::open API を使用して、NX 通信フレームワークを初期化します。
2) sf_comms_api_t::lock API を使用して、連続通信のためにチャネルをロックします（必要な場合）。
3) sf_comms_api_t::read API を使用して、データを受信します。
4) sf_comms_api_t::write API を使用して、データを送信します。
5) sf_comms_api_t::unlock コマンドを使用して連続通信用のチャネルのロックを解除します（必要な場合）。
6) sf_comms_api_t::close API を使用して、チャネルを閉じます。

これらの一般的な手順を、次の図の通常の動作フローに示します。

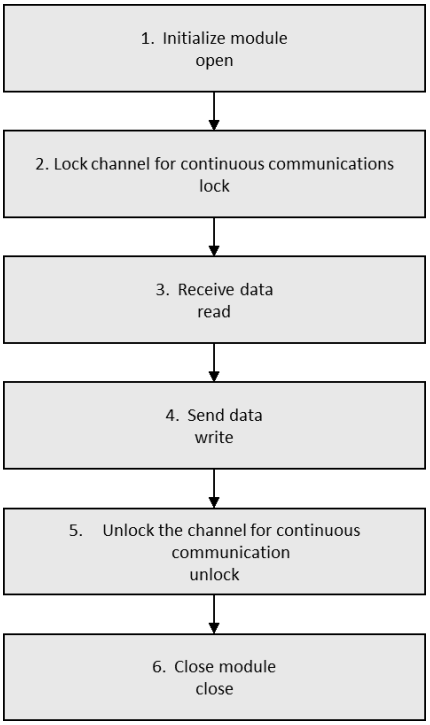


図 62:一般的な Telnet 通信フレームワークモジュールアプリケーションのフロー図

4.1.15 USBX ™通信フレームワーク

USBX™ 通信フレームワーク（sf_el_ux_comms）は、USB ポートを介した使用しやすい接続を提供するハイレベルの API を通信アプリケーション向けに実装します。このフレームワークのハイレベル API 関数は、他の接続実装（UART やイーサネットなど）と互換性があるため、アプリケーションコードを変更せずに簡単に実装を切り替えることができます。USBX 通信フレームワークは、Synergy MCU デバイス上の USB ペリフェラルを使用します。

注 :sf_el_ux_comms は非推奨です。新規の設計には sf_el_ux_comms_v2 の使用を強く推奨します。

4.1.15.1 USBX 通信フレームワークモジュールの特長

- USB 上で高レベルの接続性がサポートされますが、API を変更せずに UART やイーサネット接続に簡単に変更できます。
- 専用アクセスのためのチャネルロックをサポートします。
- USB のハイスピード（HS）またはフルスピード（FS）動作をサポートします。
- Synergy MCU でデータ転送（DMA または DTC）ペリフェラルをサポートします。
- ThreadX® 対応実装でミューテックスを使用します。

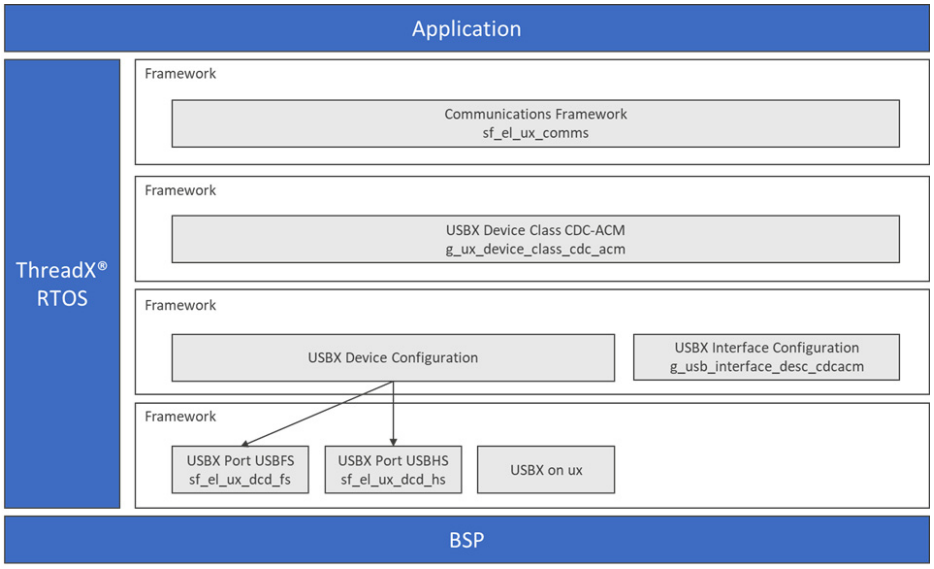


図 63: **USBX 上の通信フレームワークモジュールのブロック図**

4.1.15.2 USBX 上の通信フレームワークモジュールの API の概要

USBX 通信フレームワークは、USB 接続を介したオープン、クローズ、リード、ライトの API 関数を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

USBX 通信フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_comms0.p_api->open(g_sf_comms0.p_ctrl, g_sf_comms0.p_cfg); Initialize communications driver.
close	g_sf_comms0.p_api->close(g_sf_comms0.p_ctrl); Clean up communications driver.

Function Name	Example API Call and Description
read	g_sf_comms0.p_api->read(g_sf_comms0.p_ctrl, &destination, bytes, timeout); Read data from communications driver. This call returns after the number of bytes requested is read or if a timeout occurs while waiting for access to the driver.
write	g_sf_comms0.p_api->write(g_sf_comms0.p_ctrl, &source, bytes, timeout); Write data to communications driver. This call returns after all bytes are written or if a timeout occurs while waiting for access to the driver.
lock	g_sf_comms0.p_api->lock(g_sf_comms0.p_ctrl, lock_type, timeout); Lock the communications driver. Reserve exclusive access to the communications driver.
unlock	g_sf_comms0.p_api->unlock(g_sf_comms0.p_ctrl, lock_type); Unlock the communications driver. Release exclusive access to the communications driver.
versionGet	g_sf_comms0.p_api->versionGet(&version); Retrieve the API version in the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Channel opened successfully.
SSP_ERR_IN_USE	Channel already in use.
SSP_ERR_ASSERTION	Pointer to UART control block or configuration structure is NULL.
SSP_ERR_INTERNAL	Internal error occurs.
SSP_ERR_TIMEOUT	Timeout error.
SSP_ERR_NOT_OPEN	Module is not opened.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.15.3 USBX 上の通信フレームワークモジュールの動作の概要

USBX 上の通信フレームワークによって、USB ポートを介した使用しやすい接続が提供されます。このフレームワークのハイレベル API 関数は、他の接続実装（UART やイーサネットなど）と互換性があるため、アプリケーションコードを変更せずに簡単に実装を切り替えることができます。このモジュールは、トランザクションの完了に同期するためにミューテックスなどの ThreadX オブジェクトを使用します。USBX 通信フレームワークモジュールはロック機能をサポートします。つまり、ユーザーが通信フレームワークをスレッドにロックすることができ、複数のスレッドが同一の USBX ポートを安全に使用できます。ロックすると、アプリケーションが一定時間（sf_comms_api_t::lock API コールから sf_comms_api_t::unlock API コールまで）USB ポートを確保できます。ハイレベル API（sf_comms_api_t::read API、sf_comms_api_t::write API）を使用して、USBX CDC-ACM 通信インタフェースを介してホストとのデータの送受信をサポートします。

USBX 上の通信フレームワークモジュールの動作に関する重要な注意事項と制限事項

USBX ドライバーは、ターゲット MCU でサポートされるバージョンに応じて、HS または FS の USB 周辺機器に実装できます。

USBX CDC-ACM インスタンスのアクティブ化と非アクティブ化のために、それぞれ事前定義された弱いコールバック関数が用意されています。ユーザーは必要に応じて、Synergy コンフィギュレータで提供されているコールバック関数名を使用してユーザー関数を定義することにより、この 2 つのコールバック関数をオーバーライドすることができます。

このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.1.15.4 アプリケーションへの USBX 上の通信フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに USBX 通信フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユー

ザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。USBX 通信フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

USBX 上の通信フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_comms0 Communications Framework on sf_el_ux_comms	Threads	New Stack> Framework> Connectivity> Communications Framework on sf_el_ux_comms

次の図に示すように、USBX 通信フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

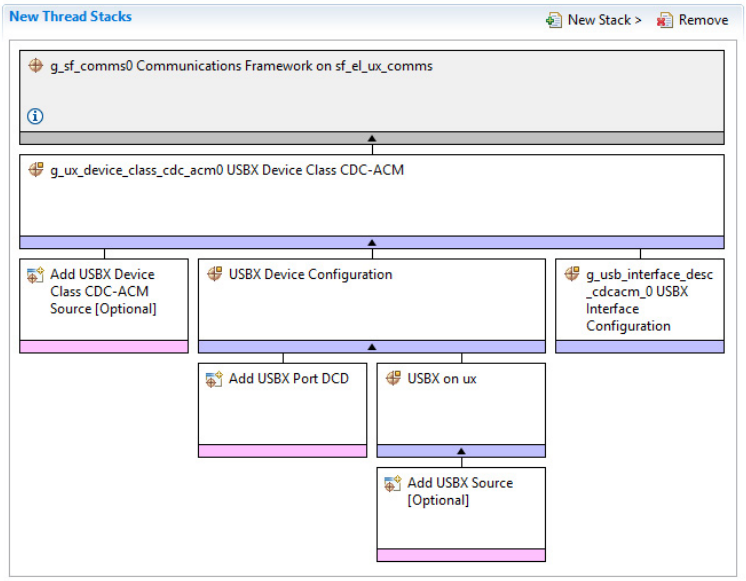


図 64:USBX 上の通信フレームワークモジュールのスタック

4.1.15.5 USBX 上の通信フレームワークモジュールの構成

ユーザーは必要な動作に合わせて、USBX 通信フレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

USBX 通信フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Read Input Buffer Size (Bytes)	128	Maximum number of bytes that can be received at a time in the read() API.
Timeout in ticks	1000	Timeout value to suspend a USBX CDC instance creation in the open() API.
Name	g_sf_comms0	Module name.
Name of the generated initialization function	sf_comms_init0	Name of helper function to initialize Communications Framework. The function will be presented in the auto-generated code in the <xxx_thread>.c, where <xxx_thread> is the name of your thread symbol given to the Thread property. The function is to be called in the auto-generated code if Auto sf_comms Initialization property is Enabled. If Disabled, the function can be called in the user application.

ISDE Property	Value	Description
Auto Initialization	Enable, Disable Default: Enable	Auto Initialization support of Communications Framework. The helper function above will be called in the auto-generated code if this configuration is enabled. Else, the function will not be called automatically and user can call it sometime later.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

USBX 通信フレームワークのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

USBX デバイスクラス CDC-ACM モジュールの構成設定

ISDE Property	Value	Description
Name	g_ux_device_class_cdc_acm0	Specify the name of the USBX Device CDC-ACM Class module instance. It must be a valid C symbol.
USBX CDC-ACM instance_activate Function Callback	ux_cdc_device0_instance_activate	Specify the name of the instance_activate user callback function for the USBX Device CDC-ACM Class module. Name must be a valid C symbol. See the USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device CDC-ACM Class" for more information about the instance_activate callback function.

ISDE Property	Value	Description
USBX CDC-ACM instance_deactivate Function Callback	ux-cdc_device0_instance_deactivate	Specify the name of the instance_deactivate user callback function for the USBX Device CDC-ACM Class module. Name must be a valid C symbol. Refer to the USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device CDC-ACM Class" for more information about the instance_activate callback function.
Name of generated initialization function	ux_device_class_cdc_acm_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX デバイス構成インスタンスの構成設定

ISDE Property	Value	Description
Vendor ID	0x045B	Specify Vendor ID assigned by USB-IF. This configuration is a part of the USB Device Descriptor (idVendor).
Product ID	0x0000	Specify Product ID assigned by manufacturer. This configuration is a part of the Device Descriptor (idProduct).

ISDE Property	Value	Description
Device Release Number	0x0000	Specify Device Release Number in binary-coded decimal. This configuration is a part of the USB Device Descriptor (bcdDevice).
Index of Manufacturing String Descriptor	0x00	Specify the Index of Manufacturer String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iManufacturer). Set zero if String Descriptor is not used. See section USBX-String-Framework-Configuration for more information.
Index of Product String Descriptor	0x00	Specify the Index of Product String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iProduct). Set zero if String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Index of Serial Number String Descriptor	0x00	Specify the Index of Serial Number String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iSerialNumber). Set zero if the String Descriptor is not used. See section "USBX String Framework Configuration" for more information.

ISDE Property	Value	Description
Class Code	Communications(CDC), HID, Mass Storage, Miscellaneous, Vendor specific Default: Communications(CDC)	Select the USB Device Class Code. This configuration is a part of the USB Configuration Descriptor (bDeviceClass).
Index of String Descriptor describing this configuration	0x00	Specify the Index of String Descriptor describing this configuration. This configuration is a part of the USB Configuration Descriptor (iConfiguration). Set zero if String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Size of USB Descriptor in bytes for this configuration (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	Specify the size of USB Descriptor in bytes. Modify the value for Vendor-specific Class, otherwise you can set zero to calculate the size automatically in the auto-generated code from Synergy Configuration tool. This configuration is a part of the USB Configuration Descriptor (wTotalLength).
Number of Interfaces (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	Specify the Number of interfaces supported by this configuration. Modify the value for Vendor-specific Class, otherwise you can set zero to calculate the value automatically in the auto-generated code from Synergy Configuration tool. This configuration is a part of the USB Configuration Descriptor (bNumInterfaces).

ISDE Property	Value	Description
Self-Powered	Enable, Disable Default: Enable	Enable this configuration if your USB Device is a self-powered device. This configuration is a part of the USB Configuration Descriptor (bmAttributes bit6).
Remote Wakeup	Enable, Disable Default: Disable	Enable this configuration if your USB Device supports remote wakeup. This configuration is a part of the USB Configuration Descriptor (bmAttributes bit5).
Maximum Power Consumption (in 2mA units)	50	Set the maximum power consumption of your device to indicate the amount of bus power required. This configuration is 2mA units, thus, the maximum 500 mA can be specified. This configuration is a part of the USB Configuration Descriptor (bMaxPower).
Supported Language Code	0x0409	Specify the Language ID Code. For example, 0x0409 English - United States. This configuration is used for Language ID Framework code generation. See section "USBX Language Framework Configuration" for more information.
Name of USBX String Framework	NULL	Specify the name of user defined USBX String Framework. This must be a valid C symbol. Set NULL if the String Descriptor is not used. See section "USBX String Framework Configuration" for more information.

ISDE Property	Value	Description
Total index number of USB String Descriptors in USB String Framework	0	Specify the total number of index for String Descriptor. See section "USBX String Framework Configuration" for more information.
Name of USBX Language Framework	NULL	Specify the name of user defined USBX Language Framework. This must be a valid C symbol. If '0' is set to the property "Total Number of Language Support", this configuration is ignored. See section "USBX Language Framework Configuration" for more information.
Number of Languages to support (US English is applied if zero is set)	0	Specify the total number of languages to support. See section "USBX String Framework Configuration" for more information. If '0' is set here, US English (0x0409) is applied as the default language.
Name of generated initialization function	ux_device_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX インタフェース構成 CDC-ACM インスタンスの構成設定

ISDE Property	Value	Description
Name	g_usb_interface_desc_cdcacm_0	Specify the name of USBX Interface Descriptor for CDC-ACM. It must be a valid C symbol.

ISDE Property	Value	Description
Interface Number of Communications Class interface	0x00	Specify the index number of Communications Class interface. This configuration is a part of the USB Interface Descriptor (bInterface). The number must not be duplicated with the Interface Number of Data Class interface. Also must not be duplicated with any Interface Numbers if your USB device consists of a USB composite device.
Interrupt Transfer endpoint to use for Communications Class	Endpoint 1-9 Default: Endpoint 3	Specify the Endpoint Number of Interrupt Endpoint. It must not be duplicated with ones for the other Endpoints.
Polling period for Interrupt Endpoint (in mS/125us units for FS/HS)	0x0F	Specify the Interval for polling Endpoint transfers. This configuration is valid for Interrupt Endpoint and ignored for Bulk Endpoints. Value is in frame counts (1ms units for FS mode and 125us units for HS mode).
Interface Number of Data Class interface	0x01	Specify the index number of Data Class interface. This configuration is a part of the USB Interface Descriptor (bInterface). The number must not be duplicated with the Interface Number of Communications Class interface. Also must not be duplicated with any Interface Numbers if your USB device consists of a USB composite device.
Bulk In Transfer endpoint to use for Data Class	Endpoint 1-9 Default: Endpoint 1	Specify the Endpoint Number of Bulk In Endpoint. It must not be duplicated with ones for the other Endpoints.

ISDE Property	Value	Description
Bulk Out Transfer endpoint to use for Data Class	Endpoint 1-9 Default: Endpoint 2	Specify the Endpoint Number of Bulk Out Endpoint. It must not be duplicated with ones for the other Endpoints.
Index of String Descriptor Describing Communications Class interface (Interface Descriptor: Interface)	0x00	Specify the index number of String Descriptor Describing Communications Class interface. This configuration is a part of the USB Interface Descriptor (iInterface). Set '0' if do not have String information for the interface.
Index of String Descriptor Describing Data Class interface (Interface Descriptor: Interface)	0x00	Specify the index number of String Descriptor Describing Data Class interface. This configuration is a part of the USB Interface Descriptor (iInterface). Set '0' if do not have String information for the interface.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBFS の sf_el_ux 上の USBX ポート DCD の構成設定

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for full-speed USB.
LDO Regulator (Only for S3 and S1 part MCUs)	Enable, Disable Default: Disable	Select the LDO regulator will be enabled.
Name	g_sf_el_ux_dcd_fs_0	Module name.
USB Controller Selection	USBFS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBHS 上の USBX ポート DCD の構成設定

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for high speed USB.
Name	g_sf_el_ux_dcd_hs_0	Module name.
USB Controller Selection	USBHS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ux 上の USBX インスタンスの構成設定

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	Name must be a valid C symbol.
USBX Pool Memory Size	18432	See section "Express Logic USBX Memory Requirements" for the required memory size for each classes.
User Callback for Host Event Notification (Only valid for USB Host)	NULL	Name must be a valid C symbol. The name of User defined USBX Host event notification can be given to this property.
Name of generated initialization function	ux_common_init0	Name of generated initialization function selection.

ISDE Property	Value	Description
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX 上の通信フレームワークモジュールのクロック構成

USB ペリフェラルモジュールは UCLK をクロックソースとして使用します。UCLK は 48 MHz の動作に合わせて構成する必要があります。SSP 構成ウィンドウで、[Clocks] タブを選択してクロックソースの設定を確認します。

USBX 上の通信フレームワークモジュールのピン構成

USB ペリフェラルモジュールは、MCU ピンを使用して外部デバイスと通信します。I/O ピンを選択し、外部デバイスの要件に合うように構成します。次の表では [SSP Configuration] ウィンドウでのピンの選択方法を示し、それ以降の表では USB ピンを例に使って選択プロセスを示します。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

USBFS および USBHS のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
USBFS	Pins	Select Peripherals > Connectivity: USBFS> USBFS0
USBHS	Pins	Select Peripherals > Connectivity: USBHS> USBHS0

注 : 選択シーケンスでは、USBFS0 または USBHS0 がドライバーに必要なハードウェアターゲットであることを想定しています。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select device as the Operation Mode

Property	Value	Description
USBDP	USBDP	USBDP pin
USBDM	USBDM	USBDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	None	VBUSEN pin
VBUS	None, P407 Default: P407	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID Pin
VCCUSB	VCCUSB	VCCUSB pin
VSSUSB	VSSUSB	VSSUSB pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select Device as the Operation Mode
USBHSDP	USBHSDP	USBHSDP pin
USBHSDM	USBHSDM	USBHSDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	PB00	VBUSEN pin

Property	Value	Description
VBUS	PB01	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID pin
USBHSRREF	USBHSRREF	USBHSRREF pin
AVCCUSBHS	AVCCUSBHS	AVCCUSBHS pin
AVSSUSBHS	AVSSUSBHS	AVSSUSBHS pin
PVSSUSBHS	PVSSUSBHS	PVSSUSBHS pin
VCCUSBHS	VCCUSBHS	VCCUSBHS pin
VSS1USBHS	VSS1USBHS	VSS1USBHS pin
VSS2USBHS	VSS2USBHS	VSS2USBHS pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.1.15.6 アプリケーションでの USBX 上の通信フレームワークモジュールの使用

アプリケーションで USBX 通信フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_comms_api_t::open API を使用して、USBX 通信フレームワークを初期化します。
2) sf_comms_api_t::lock API を使用して、連続通信のためにチャネルをロックします（必要な場合）。
3) sf_comms_api_t::read API を使用して、データを受信します。
4) sf_comms_api_t::write API を使用して、データを送信します。
5) sf_comms_api_t::unlock コマンドを使用して連続通信用のチャネルのロックを解除します（必要な場合）。
6) sf_comms_api_t::close API を使用して、チャネルを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

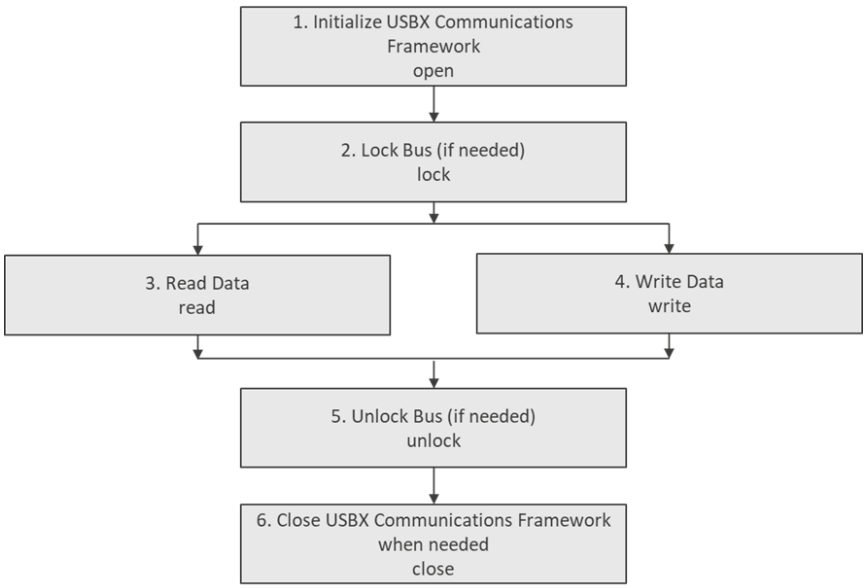


図 65:通常の USBX 通信フレームワークモジュールアプリケーションのフロー図

4.1.16 USBX ™ V2 通信フレームワーク

USBX ™ V2 通信フレームワーク（sf_el_ux_comms_v2）は、USB ポートを介した使用しやすい接続を提供するハイレベルの API を通信アプリケーション向けに実装します。このフレームワークのハイレベル API 関数は、他の接続実装（UART やイーサネットなど）と互換性があるため、アプリケーションコードを変更せずに簡単に実装を切り替えることができます。USBX V2 通信フレームワークは、Synergy MCU デバイス上の USB ペリフェラルを使用します。

注 :sf_el_ux_comms は非推奨です。新規の設計には sf_el_ux_comms_v2 の使用を強く推奨します。

4.1.16.1 USBX V2 通信フレームワークモジュールの特長

- USB 上で高レベルの接続性がサポートされますが、API を変更せずに UART やイーサネット接続に簡単に変更できます。
- 専用アクセスのためのチャネルロックをサポートします。
- USB のハイスピード（HS）またはフルスピード（FS）動作をサポートします。
- Synergy MCU でデータ転送（DMA または DTC）ペリフェラルをサポートします。
- ThreadX® 対応実装でミューテックスを使用します。

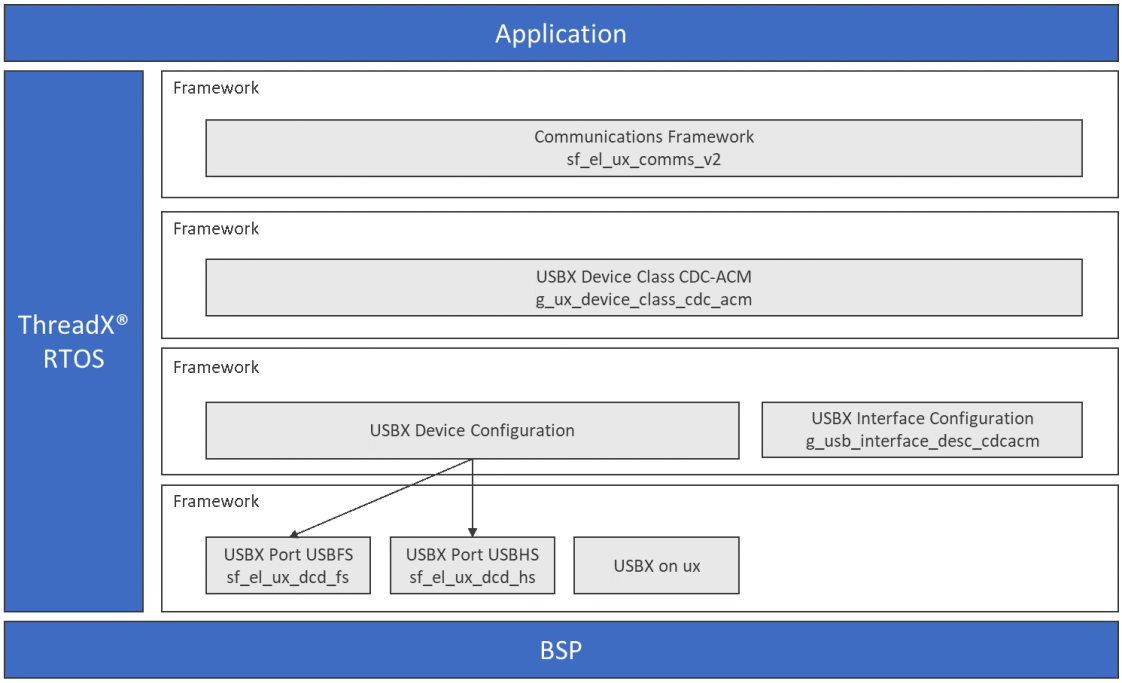


図 66: **USBX V2 通信フレームワークモジュールのブロック図**

4.1.16.2 USBX V2 通信フレームワークモジュールの API の概要

USBX V2 通信フレームワークは、USB 接続を介したオープン、クローズ、リード、ライトの API 関数を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

USBX V2 通信フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_comms0.p_api->open(g_sf_comms0.p_ctrl, g_sf_comms0.p_cfg); Initialize communications driver.
close	g_sf_comms0.p_api->close(g_sf_comms0.p_ctrl); Clean up communications driver.

Function Name	Example API Call and Description
read	g_sf_comms0.p_api->read(g_sf_comms0.p_ctrl, &destination, bytes, timeout); Read data from communications driver. This call returns after the number of bytes requested is read or if a timeout occurs while waiting for access to the driver.
write	g_sf_comms0.p_api->write(g_sf_comms0.p_ctrl, &source, bytes, timeout); Write data to communications driver. This call returns after all bytes are written or if a timeout occurs while waiting for access to the driver.
lock	g_sf_comms0.p_api->lock(g_sf_comms0.p_ctrl, lock_type, timeout); Lock the communications driver. Reserve exclusive access to the communications driver.
unlock	g_sf_comms0.p_api->unlock(g_sf_comms0.p_ctrl, lock_type); Unlock the communications driver. Release exclusive access to the communications driver.
versionGet	g_sf_comms0.p_api->versionGet(&version); Retrieve the API version in the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Channel opened successfully.
SSP_ERR_IN_USE	Channel already in use.
SSP_ERR_ASSERTION	Pointer to UART control block or configuration structure is NULL.
SSP_ERR_INTERNAL	Internal error occurs.
SSP_ERR_TIMEOUT	Timeout error.
SSP_ERR_NOT_OPEN	Module is not opened.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.16.3 USBX V2 通信フレームワークモジュールの動作の概要

USBX V2 通信フレームワークによって、USB ポートを介した使用しやすい接続が提供されます。このフレームワークのハイレベル API 関数は、他の接続実装（UART やイーサネットなど）と互換性があるため、アプリケーションコードを変更せずに簡単に実装を切り替えることができます。このモジュールは、トランザクションの完了に同期するためにミューテックスなどの ThreadX オブジェクトを使用します。USBX V2 通信フレームワークモジュールはロック機能をサポートします。つまり、ユーザーが通信フレームワークをスレッドにロックすることができ、複数のスレッドが同一の USBX ポートを安全に使用できます。ロックすると、アプリケーションが一定時間（sf_comms_api_t::lock API コールから sf_comms_api_t::unlock API コールまで）USB ポートを確保できます。ハイレベル API（sf_comms_api_t::read API、sf_comms_api_t::write API）を使用して、USBX CDC-ACM 通信インタフェースを介してホストとのデータの送受信をサポートします。

USBX V2 通信フレームワークモジュールの動作に関する重要な注意事項と制限事項

USBX V2 通信フレームワークモジュールの動作に関する重要な注意事項

USBX ドライバーは、ターゲット MCU でサポートされるバージョンに応じて、HS または FS の USB 周辺機器に実装できます。

USBX CDC-ACM インスタンスのアクティブ化と非アクティブ化のために、それぞれ事前定義された弱いコールバック関数が用意されています。ユーザーは必要に応じて、Synergy コンフィギュレータで提供されているコールバック関数名を使用してユーザー関数を定義することにより、この 2 つのコールバック関数をオーバーライドすることができます。

USBX V2 通信フレームワークモジュールの動作に関する制限事項

このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.1.16.4 アプリケーションへの USBX V2 通信フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに USBX V2 通信フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

USBX V2 通信フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

USBX V2 通信フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_comms0 Communications Framework on sf_el_ux_comms_v2	Threads	New Stack> Framework> Connectivity> Communications Framework on sf_el_ux_comms_v2

次の図に示すように、USBX V2 通信フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

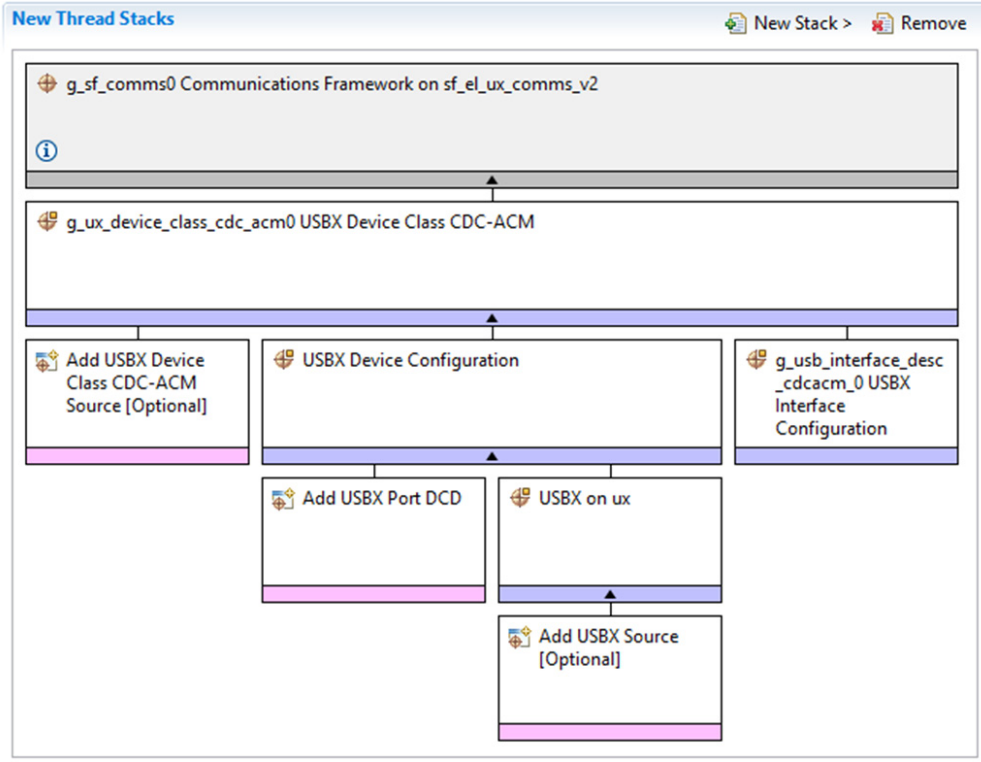


図 67:**USBX V2 通信フレームワークモジュールのスタック**

4.1.16.5 USBX V2 通信フレームワークモジュールの構成

ユーザーは必要な動作に合わせて、USBX V2 通信フレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

USBX V2 通信フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Read Input Buffer Size (Bytes)	128	Maximum number of bytes that can be received at a time in the read() API.
Timeout in ticks	1000	Timeout value to suspend a USBX CDC instance creation in the open() API.
Name	g_sf_comms0	Module name.
Name of the generated initialization function	sf_comms_init0	Name of helper function to initialize Communications Framework. The function will be presented in the auto-generated code in the <xxx_thread>.c, where <xxx_thread> is the name of your thread symbol given to the Thread property. The function is to be called in the auto-generated code if Auto sf_comms Initialization property is Enabled. If Disabled, the function can be called in the user application.
Auto Initialization	Enable, Disable Default: Enable	Auto Initialization support of Communications Framework. The helper function above will be called in the auto-generated code if this configuration is enabled. Else, the function will not be called automatically and user can call it sometime later.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

USBX V2 通信フレームワークのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

USBX デバイスクラス CDC-ACM モジュールの構成設定

ISDE Property	Value	Description
Name	g_ux_device_class_cdc_acm0	Specify the name of the USBX Device CDC-ACM Class module instance. It must be a valid C symbol.
USBX CDC-ACM instance_activate Function Callback	ux_cdc_device0_instance_activate	Specify the name of the instance_activate user callback function for the USBX Device CDC-ACM Class module. Name must be a valid C symbol. See the USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device CDC-ACM Class" for more information about the instance_activate callback function.
USBX CDC-ACM instance_deactivate Function Callback	ux-cdc_device0_instance_deactivate	Specify the name of the instance_deactivate user callback function for the USBX Device CDC-ACM Class module. Name must be a valid C symbol. Refer to the USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device CDC-ACM Class" for more information about the instance_activate callback function.

ISDE Property	Value	Description
Name of generated initialization function	ux_device_class_cdc_acm_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX デバイス構成インスタンスの構成設定

ISDE Property	Value	Description
Vendor ID	0x045B	Specify Vendor ID assigned by USB-IF. This configuration is a part of the USB Device Descriptor (idVendor).
Product ID	0x0000	Specify Product ID assigned by manufacturer. This configuration is a part of the Device Descriptor (idProduct).
Device Release Number	0x0000	Specify Device Release Number in binary-coded decimal. This configuration is a part of the USB Device Descriptor (bcdDevice).
Index of Manufacturing String Descriptor	0x00	Specify the Index of Manufacturer String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iManufacturer). Set zero if String Descriptor is not used. See section USBX-String-Framework-Configuration for more information.

ISDE Property	Value	Description
Index of Product String Descriptor	0x00	Specify the Index of Product String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iProduct). Set zero if String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Index of Serial Number String Descriptor	0x00	Specify the Index of Serial Number String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iSerialNumber). Set zero if the String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Class Code	Communications(CDC), HID, Mass Storage, Miscellaneous, Vendor specific Default: Communications(CDC)	Select the USB Device Class Code. This configuration is a part of the USB Configuration Descriptor (bDeviceClass).
Index of String Descriptor describing this configuration	0x00	Specify the Index of String Descriptor describing this configuration. This configuration is a part of the USB Configuration Descriptor (iConfiguration). Set zero if String Descriptor is not used. See section "USBX String Framework Configuration" for more information.

ISDE Property	Value	Description
Size of USB Descriptor in bytes for this configuration (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	Specify the size of USB Descriptor in bytes. Modify the value for Vendor-specific Class, otherwise you can set zero to calculate the size automatically in the auto-generated code from Synergy Configuration tool. This configuration is a part of the USB Configuration Descriptor (wTotalLength).
Number of Interfaces (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	Specify the Number of interfaces supported by this configuration. Modify the value for Vendor-specific Class, otherwise you can set zero to calculate the value automatically in the auto-generated code from Synergy Configuration tool. This configuration is a part of the USB Configuration Descriptor (bNumInterfaces).
Self-Powered	Enable, Disable Default: Enable	Enable this configuration if your USB Device is a self-powered device. This configuration is a part of the USB Configuration Descriptor (bmAttributes bit6).
Remote Wakeup	Enable, Disable Default: Disable	Enable this configuration if your USB Device supports remote wakeup. This configuration is a part of the USB Configuration Descriptor (bmAttributes bit5).

ISDE Property	Value	Description
Maximum Power Consumption (in 2mA units)	50	Set the maximum power consumption of your device to indicate the amount of bus power required. This configuration is 2mA units, thus, the maximum 500 mA can be specified. This configuration is a part of the USB Configuration Descriptor (bMaxPower).
Supported Language Code	0x0409	Specify the Language ID Code. For example, 0x0409 English - United States. This configuration is used for Language ID Framework code generation. See section "USBX Language Framework Configuration" for more information.
Name of USBX String Framework	NULL	Specify the name of user defined USBX String Framework. This must be a valid C symbol. Set NULL if the String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Total index number of USB String Descriptors in USB String Framework	0	Specify the total number of index for String Descriptor. See section "USBX String Framework Configuration" for more information.
Name of USBX Language Framework	NULL	Specify the name of user defined USBX Language Framework. This must be a valid C symbol. If '0' is set to the property "Total Number of Language Support", this configuration is ignored. See section "USBX Language Framework Configuration" for more information.

ISDE Property	Value	Description
Number of Languages to support (US English is applied if zero is set)	0	Specify the total number of languages to support. See section "USBX String Framework Configuration" for more information. If '0' is set here, US English (0x0409) is applied as the default language.
Name of generated initialization function	ux_device_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX インタフェース構成 CDC-ACM インスタンスの構成設定

ISDE Property	Value	Description
Name	g_usb_interface_desc_cdcacm_0	Specify the name of USBX Interface Descriptor for CDC-ACM. It must be a valid C symbol.
Interface Number of Communications Class interface	0x00	Specify the index number of Communications Class interface. This configuration is a part of the USB Interface Descriptor (bInterface). The number must not be duplicated with the Interface Number of Data Class interface. Also must not be duplicated with any Interface Numbers if your USB device consists of a USB composite device.

ISDE Property	Value	Description
Interrupt Transfer endpoint to use for Communications Class	Endpoint 1-9 Default: Endpoint 3	Specify the Endpoint Number of Interrupt Endpoint. It must not be duplicated with ones for the other Endpoints.
Polling period for Interrupt Endpoint (in mS/125us units for FS/HS)	0x0F	Specify the Interval for polling Endpoint transfers. This configuration is valid for Interrupt Endpoint and ignored for Bulk Endpoints. Value is in frame counts (1ms units for FS mode and 125us units for HS mode).
Interface Number of Data Class interface	0x01	Specify the index number of Data Class interface. This configuration is a part of the USB Interface Descriptor (bInterface). The number must not be duplicated with the Interface Number of Communications Class interface. Also must not be duplicated with any Interface Numbers if your USB device consists of a USB composite device.
Bulk In Transfer endpoint to use for Data Class	Endpoint 1-9 Default: Endpoint 1	Specify the Endpoint Number of Bulk In Endpoint. It must not be duplicated with ones for the other Endpoints.
Bulk Out Transfer endpoint to use for Data Class	Endpoint 1-9 Default: Endpoint 2	Specify the Endpoint Number of Bulk Out Endpoint. It must not be duplicated with ones for the other Endpoints.
Index of String Descriptor Describing Communications Class interface (Interface Descriptor: Interface)	0x00	Specify the index number of String Descriptor Describing Communications Class interface. This configuration is a part of the USB Interface Descriptor (iInterface). Set '0' if do not have String information for the interface.

ISDE Property	Value	Description
Index of String Descriptor Describing Data Class interface (Interface Descriptor: Interface)	0x00	Specify the index number of String Descriptor Describing Data Class interface. This configuration is a part of the USB Interface Descriptor (iInterface). Set '0' if do not have String information for the interface.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBFS の sf_el_ux 上の USBX ポート DCD の構成設定

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for full-speed USB.
LDO Regulator (Only for S3 and S1 part MCUs)	Enable, Disable Default: Disable	Select the LDO regulator will be enabled.
Name	g_sf_el_ux_dcd_fs_0	Module name.
USB Controller Selection	USBFS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBHS 上の USBX ポート DCD の構成設定

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for high speed USB.
Name	g_sf_el_ux_dcd_hs_0	Module name.
USB Controller Selection	USBHS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ux 上の USBX インスタンスの構成設定

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	Name must be a valid C symbol.
USBX Pool Memory Size	18432	See section "Express Logic USBX Memory Requirements" for the required memory size for each classes.
User Callback for Host Event Notification (Only valid for USB Host)	NULL	Name must be a valid C symbol. The name of User defined USBX Host event notification can be given to this property.
Name of generated initialization function	ux_common_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX V2 通信フレームワークモジュールのクロック構成

USB ペリフェラルモジュールは UCLK をクロックソースとして使用します。UCLK は 48 MHz の動作に合わせて構成する必要があります。SSP 構成ウィンドウで、[Clocks] タブを選択してクロックソースの設定を確認します。

USBX V2 通信フレームワークモジュールのピン構成

USB ペリフェラルモジュールは、MCU ピンを使用して外部デバイスと通信します。I/O ピンを選択し、外部デバイスの要件に合うように構成します。次の表では [SSP Configuration] ウィンドウでのピンの選択方法を示し、それ以降の表では USB ピンを例に使って選択プロセスを示します。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

USBFS および USBHS のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
USBFS	Pins	Select Peripherals > Connectivity: USBFS> USBFS0
USBHS	Pins	Select Peripherals > Connectivity: USBHS> USBHS0

注 : 選択シーケンスでは、USBFS0 または USBHS0 がドライバーに必要なハードウェアターゲットであることを想定しています。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select device as the Operation Mode
USBDP	USBDP	USBDP pin
USBDM	USBDM	USBDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin

Property	Value	Description
VBUSEN	None	VBUSEN pin
VBUS	None, P407 Default: P407	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID Pin
VCCUSB	VCCUSB	VCCUSB pin
VSSUSB	VSSUSB	VSSUSB pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select Device as the Operation Mode
USBHSDP	USBHSDP	USBHSDP pin
USBHSDM	USBHSDM	USBHSDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	PB00	VBUSEN pin
VBUS	PB01	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID pin
USBHSRREF	USBHSRREF	USBHSRREF pin

Property	Value	Description
AVCCUSBHS	AVCCUSBHS	AVCCUSBHS pin
AVSSUSBHS	AVSSUSBHS	AVSSUSBHS pin
PVSSUSBHS	PVSSUSBHS	PVSSUSBHS pin
VCCUSBHS	VCCUSBHS	VCCUSBHS pin
VSS1USBHS	VSS1USBHS	VSS1USBHS pin
VSS2USBHS	VSS2USBHS	VSS2USBHS pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.1.16.6 アプリケーションでの USBX V2 通信フレームワークモジュールの使用

アプリケーションで USBX V2 通信フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_comms_api_t::open API を使用して、USBX V2 通信フレームワークを初期化します。
2) sf_comms_api_t::lock API を使用して、連続通信のためにチャネルをロックします（必要な場合）。
3) sf_comms_api_t::read API を使用して、データを受信します。
4) sf_comms_api_t::write API を使用して、データを送信します。
5) sf_comms_api_t::unlock コマンドを使用して連続通信用のチャネルのロックを解除します（必要な場合）。
6) sf_comms_api_t::close API を使用して、チャネルを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

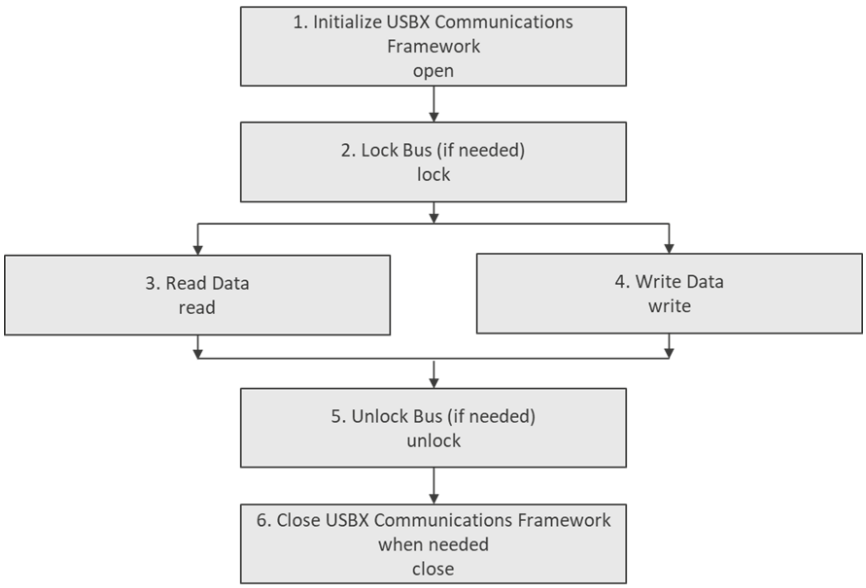


図 68:通常の USBX V2 通信フレームワークモジュールアプリケーションのフロー図

4.1.17 コンソールフレームワーク

コンソールフレームワークは、ThreadX RTOS を使用するメニュー駆動型コンソールコマンドラインインタフェース（CLI）用の完全な API 実装です。このコンソールフレームワークモジュールではローレベルの通信インタフェースを使用して、UART、USB、イーサネット Telnet 接続のいずれかに対応するハードウェアオプションに接続します。コンソールフレームワークモジュールには、ユーザー定義のコマンドのメニューとさまざまな API があり、プロンプトの表示、メニューコマンドのコールバックの識別および発行、入力文字列のリード、ライト、解析を行うことができます。

4.1.17.1 コンソールフレームワークモジュールの特長

コンソールフレームワークは以下の機能に対応します。

- メニューに基づくコマンドラインインタフェースの作成
- サブメニューと、単一呼び出しによる複数メニュー内の移動
- 親メニューへの移動、またはメニューのルートに戻る
- メニューごとのヘルプメニュー
- NULL 終端文字列の書き込みと、改行文字までの読み取り
- 引数をコマンドラインに解析する API
- 大文字と小文字を区別しない入力

コンソールフレームワークモジュールの編成（SSP コンフィギュレータの [Thread Stack] ウィンドウに表示される）を次の図に示します。各実装の選択肢（イーサネット、UART、USB）にはそれぞれローレベルモジュールがあり、開

発者が選択した実装に基づいて自動的に追加されます。ほとんどのケースで、必要なすべての構成情報がモジュールに自動的に追加されるため、開発者が選択する必要があるのははごくわずかの重要な構成設定のみです。

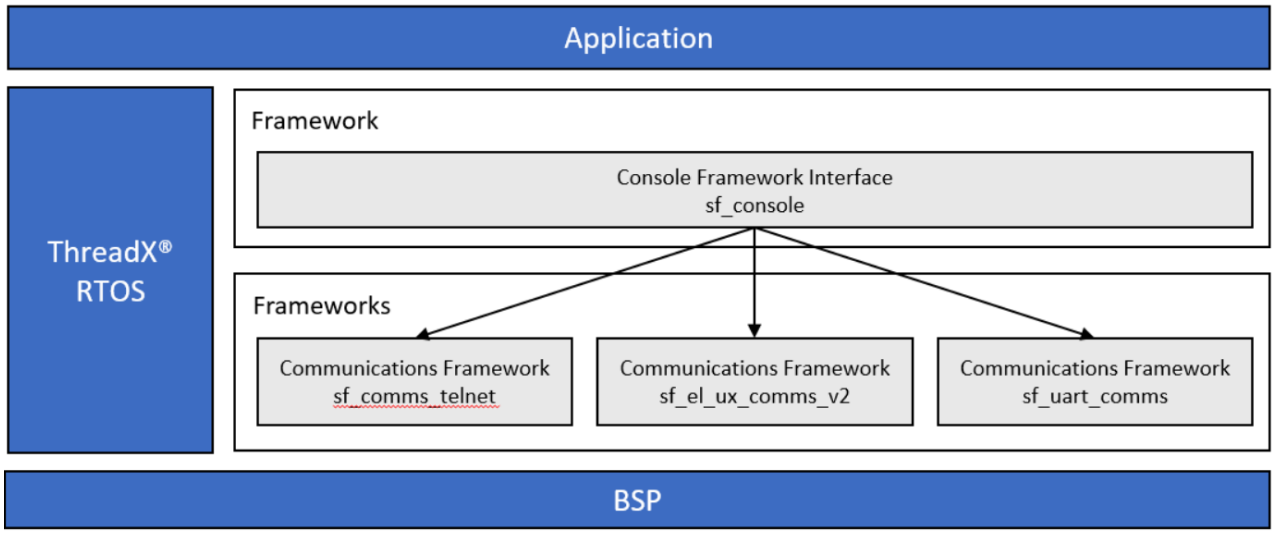


図 69: コンソールフレームワークモジュールのブロック図

4.1.17.2 コンソールフレームワークモジュール API の概要

コンソールフレームワークは、入力プロンプトのオープン、クローズ、リード、ライト、発行の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

コンソールフレームワークモジュール API 要約

Function Name	Example API Call and Description
open	g_sf_console0.p_api->open(g_sf_console0.p_ctrl, g_sf_console0.p_cfg) The open API configures the console. This function must be called before any other console functions. **L :NOTE:** This call is made automatically during system initialization, prior to entering the users thread. Unless the user closes the console, open will not need to be called.

Function Name	Example API Call and Description
close	g_sf_console0.p_api->close(g_sf_console0.p_ctrl); The close API handles the clean-up of internal driver data.
prompt	g_sf_console0.p_api->prompt(g_sf_console0.p_ctrl, NULL, TX_WAIT_FOREVER); The prompt API prints the prompt string from the menu, waits for input, parses the input based on the menu, and calls the appropriate callback function if a command is identified.
parse	g_sf_console0.p_api->parse(g_sf_console0.p_ctrl, commands, input, s_length); The parse API looks for an input string in the command menu and, if one is found, calls the appropriate callback function.
read	g_sf_console0.p_api->read(g_sf_console0.p_ctrl, ch, 1, TX_WAIT_FOREVER); The read API puts data into the destination, byte-by-byte and echoes the input to the console. Backspace, delete, and left/right arrow keys are supported. Read completes when a line ending with CR, CR+LF, or CR+NULL is received, or when the input exceeds the number of bytes allowed. If the buffer overflows SF_CONSOLE_MAX_INPUT_LENGTH, the read will return an error code.
write	g_sf_console0.p_api->write(g_sf_console0.p_ctrl, (uint8_t*)data_string, TX_WAIT_FOREVER); The write API gets the buffer mutex object and handles data transmission at the HAL layer. It obtains the event flag to synchronize the completion of a data transfer.

Function Name	Example API Call and Description
argumentFind	g_sf_console0.p_api->argumentFind("LED", p_args->p_remaining_string, NULL, &led_num); The argumentFind API locates a command line argument in an input string and returns the index of the character immediately following the argument. Any string numbers are converted to integers.
versionGet	g_sf_console0.p_api->versionGet(&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_ASSERTION	p_ctrl is NULL.
SSP_ERR_UNSUPPORTED	Command not found in the current menu.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.17.3 コンソールフレームワークモジュールの動作の概要

コンソールフレームワークモジュールは、ThreadX 対応のコマンドラインインタフェース（CLI）です。このモジュールは、ミューテックスなどの ThreadX オブジェクトを使用してブロックを行い、イベント フラグなどの同期手法を使用してトランザクションを完了します。コンソールフレームワークの動作の重要な要素は初期化および入力処理です。それぞれについて次のセクションで説明します。

コンソールフレームワークモジュールの初期化

open コールは、モジュールが追加されたファイル内に ISDE によって自動的に生成されます。この open 呼び出しを実行するには、デフォルトでコンフィギュレータにおける名前 (g_sf_console_root_menu) と一致する変数名を持つルートメニューをアプリケーションで定義する必要があります。必要なハードウェア接続が確立されていれば、実行がスレッドエントリ関数に達するまでにモジュールが使用可能になります。

コンソールフレームワークモジュールの入力処理

コンソールフレームワークモジュールでは、一連のメニュー、コマンド構造体、およびコールバックが必要です。コンソールフレームワークモジュールは通常はプロンプトで作動し、多くの場合、entry スレッド内の while ループに配置されます。このフレームワークの sf_console_api_t::prompt API は、現在のメニューをプロンプトとして出力してから、入力を読み取り、それをコンソールにエコーバックします（プロパティでエコーが無効化されている場合は除く）。

次の動作がユーザー入力に対して実行されます。

コンソールで入力が行われる際の動作は次のとおりです。

- BackSpace キーを押すとカーソルの前の文字が削除されます。
- Delete キーを押すとカーソルの後の文字が削除されます。
- 左右の矢印キーを押すとカーソルが移動します。
- 上方向キーを押すと、他に何も入力していない場合にのみ、最後のコマンドが入力されます。

注 : 最後のコマンドより前の履歴は保持されません。つまり、上方向キーを 2 回押しても、最後のコマンドの前に入力されたコマンドの情報は残っていないため、最後のコマンドが引き続き表示されます。

リード入力時に改行文字が現れると、入力文字列が解析され、関連するコールバックがコールされるか、次のメニューに切り替わります(コマンドのコールバックの代わりに SF_CONSOLE_CALLBACK_NEXT_FUNCTION が使用されている場合)。解析はコールバック関数が呼び出されるまで続行されます。sf_console_api_t::prompt API が再びコールされた場合は、コールバック関数を含むメニューを表示して入力待ちになります。親メニューに移動するには、'^' を押します。サブメニューからルート メニューに移動するには、'~' を押します。

パーサーは空白文字または文字列の終わりまで入力コマンドを解析します。

例 :

コマンドリスト [2] = {"echo", "setbitrate"}; の場合、

1）>echo は有効です。

2）>echo<space> は有効です。

3）>echo3 はサポートされていないコマンドです。

4）>setbitrate は有効です。

5）>setbitrate 9600 ではビットレートが 9600 に設定されます。

コンソールフレームワークモジュールの必須構造体の作成 – メニュー

コンソールフレームワークではメニューが必要です。メニューの実装にコンソールフレームワークが使用する構造体の作成は開発者が行います。コンソールメニューの構造体（次の図を参照）には、前のメニューへのポインタ、メニューの名前（複数レベルのメニューを作成するため）、メニュー内のコマンド数、コマンド構造体の配列へのポインタが含まれます。次の図に示すように、コマンド配列の各エントリには、コマンド名文字列へのポインタ、help コマンドの説明文字列、関連するコマンドコールバック関数、コールバックに提供されるコンテキストパラメータが含まれます。

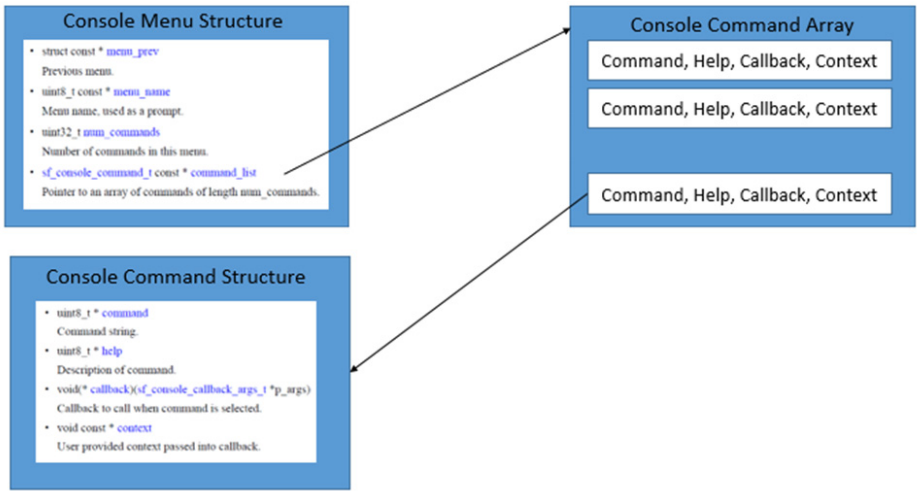


図 70: コンソールフレームワークモジュールのメニュー構造体

アプリケーションプロジェクトの例として、1 つのメニューがあるコンソールフレームワークを示します。これは、CLI で LED の切り替えを制御します。コンソールコマンド配列 (g_sf_console_commands, 下図右側) にはコマンドの配列が格納されます（このケースでは 1 つのコマンドのみ）。コマンド構造体によって、コマンドが "LED TOGGLE"、ヘルプの説明が "Toggle an LED"、コールバックが led_toggle_callback, コンテキストが NULL（この例では使用されないため）であることが定義されます。

ルートメニュー（下図左側）は g_sf_console_root_menu 構造体です。この構造体では、sf_console_menu_t::menu_prev エントリが NULL（1 つのメニューしかないため）、sf_console_menu_t::menu_name が「Root」、sf_console_menu_t::num_commands は「1」（エントリの合計数を求めるために配列サイズをエントリサイズで割った値）、sf_console_menu_t::command_list の開始アドレスは「address」（コマンド配列の最初のエントリの場所）が定義されます。

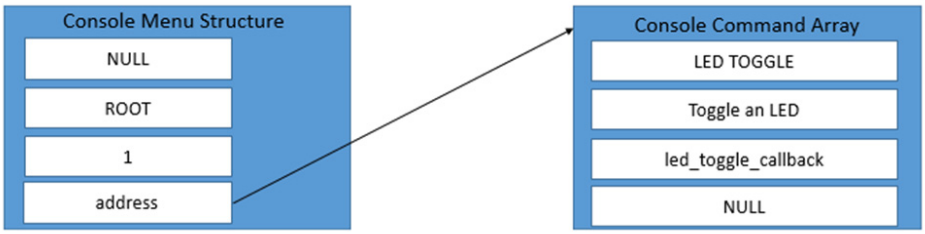


図 71: コンソールフレームワークモジュールのメニュー構造体例の図

コンソールフレームワークモジュールの動作に関する重要な注意事項と制限事項

コンソールフレームワークモジュールの動作に関する重要な注意事項

- コンソールフレームワークモジュールの sf_console_api_t::prompt API を使用するには、まずメニュー、コマンド構造体、およびコールバックを設定します。

コンソールフレームワークモジュールの動作に関する制限事項

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.17.4 アプリケーションへのコンソールフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにコンソールフレームワークモジュールを組み込む方法について説明します。

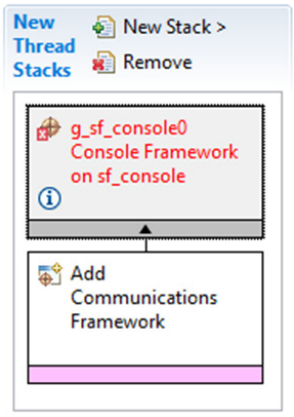
注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

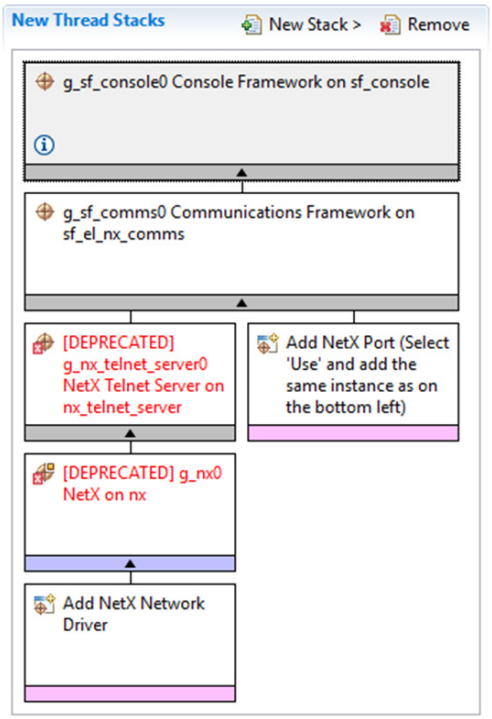
コンソールフレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。(コンソールフレームワークモジュールのデフォルト名は g_sf_console0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

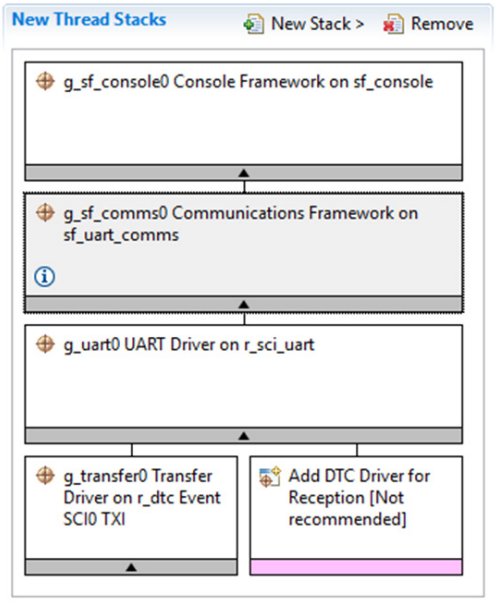
コンソールフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_console0 Console Framework on sf_console	Threads	New Stack > Framework > Services > Console Framework on sf_console

次の図に示すように、sf_console のコンソールフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。


New Thread Stacks
New Stack >
Remove
g_sf_console0 Console Framework on sf_console
Add Communications Framework


New Thread Stacks
New Stack >
Remove
g_sf_console0 Console Framework on sf_console
g_sf_comms0 Communications Framework on sf_el_nx_comms
[DEPRECATED] g_nx_telnet_server0 NetX Telnet Server on nx_telnet_server
Add NetX Port (Select 'Use' and add the same instance as on the bottom left)
[DEPRECATED] g_nx0 NetX on nx
Add NetX Network Driver


New Thread Stacks
New Stack >
Remove
g_sf_console0 Console Framework on sf_console
g_sf_comms0 Communications Framework on sf_uart_comms
g_uart0 UART Driver on r_sci_uart
g_transfer0 Transfer Driver on r_dtc Event SCI0 TXI
Add DTC Driver for Reception [Not recommended]

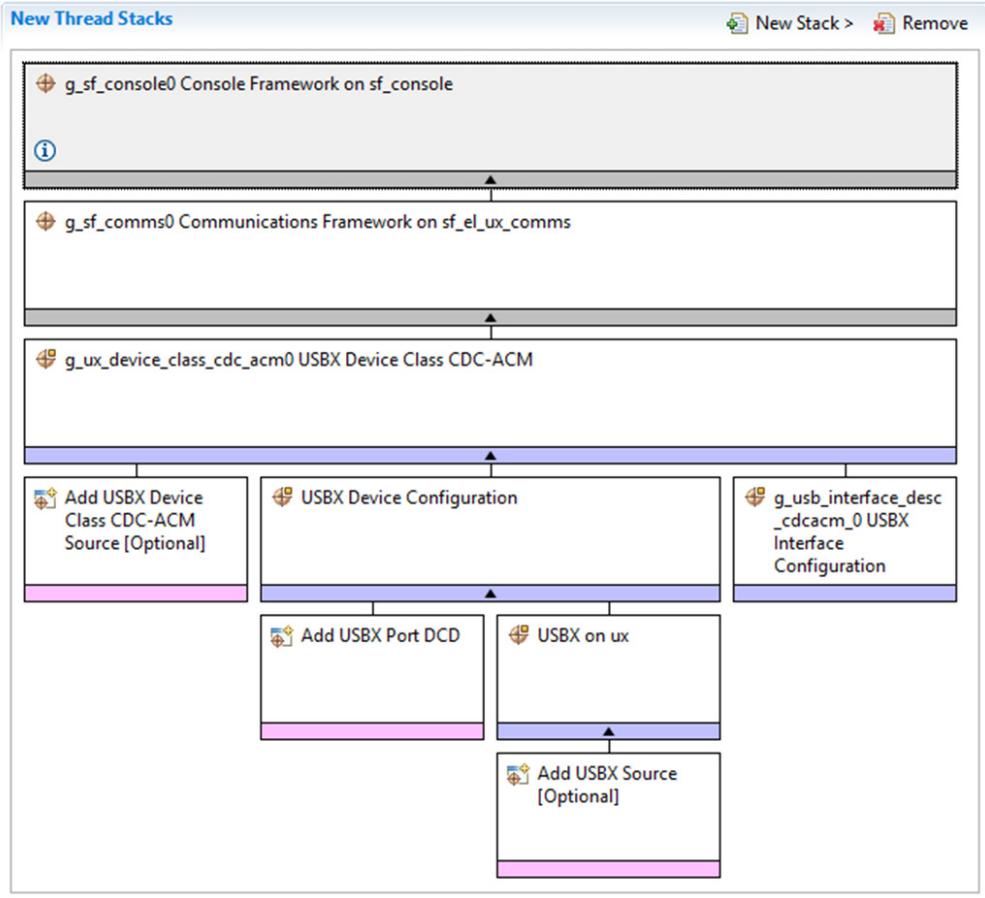


図 72: コンソールフレームワークモジュールのスタック

4.1.17.5 コンソールフレームワークモジュールの構成

ユーザーは必要な動作に合わせてコンソールフレームワークモジュールを構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成表の設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_console 上のコンソールフレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Maximum Input String Length	128	Maximum input string length selection
Maximum Write String Length	128	Maximum input string length selection
Console print timeout value	0xFFFFFFFF	Console print timeout value selection
Name	g_sf_console0	Module name
Name of Initial Menu (Application Defined)	g_sf_console_root_menu	Initial menu name
Echo	True, False Default: True	Echo selection
Autostart	True, False Default: False	Autostart selection
Name of generated initialization function	sf_console_init0	Name of generated initialization function
Auto Initialization	Enable, Disable Default: Enable	Auto Initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

コンソールフレームワークのローレベルモジュールの構成

コンソールフレームワークには、通信フレームワークの実装として NetX 通信フレームワーク、USBX 通信フレームワーク、UART 通信フレームワークの 3 種類のオプションが用意されています。これらのフレームワークにはそれぞれローレベルのモジュールオプションがあり、フレームワークの構成情報は sf_el_nx_comms, sf_el_ux_comms、sf_uart_comms の各モジュールの概要で提供されます。

4.1.17.6 アプリケーションでのコンソールフレームワークモジュールの使用

アプリケーションでコンソールフレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) メニューおよびコマンド構造体を作成します。
2) 必要なコールバックを実装します。
3) sf_console_api_t::open API を使用して、コンソールフレームワークを初期化します。
4) sf_console_api_t::prompt API を使用して、プロンプトを生成し、コマンドを処理します。
5) 必要に応じて他の API（sf_console_api_t::read、sf_console_api_t::write、sf_console_api_t::parse、または sf_console_api_t::arguementFind）を使用してコマンドを処理します。
6) 必要に応じて sf_console_api_t::close API を使用して、モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フローに示します。

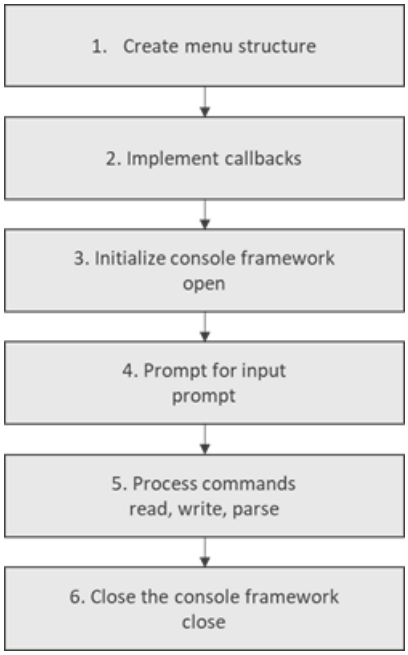


図 73:通常のコンソールフレームワークモジュールアプリケーションのフロー図

4.1.18 暗号化フレームワーク

暗号化フレームワークレイヤーは、さまざまな暗号化サービスを提供する複数の暗号化モジュールで構成されています。以下が含まれます。

- 暗号化ハードウェアの初期化、および暗号化モジュール間のリソース同期のための SF_CRYPTO。
- 真性乱数生成のための SF_CRYPTO_TRNG。
- メッセージダイジェスト生成のための SF_CRYPTO_HASH。MD5、SHA1、SHA 224、SHA 256 の各アルゴリズムがサポートされます。

- キー生成サービスのための SF_CRYPTO_KEY。AES、ECC、RSA の各キーがサポートされます。
- 暗号化および復号化サービスのための SF_CRYPTO_CIPHER。AES および RSA アルゴリズムがサポートされます。
- RSA 署名生成および検証サービスのための SF_CRYPTO_SIGNATURE。
- キーインストールサービスのための SF_CRYPTO_KEY_INSTALLATION。AES、ECC、RSA の各キーがサポートされます。

4.1.18.1 暗号化フレームワークモジュールの特長

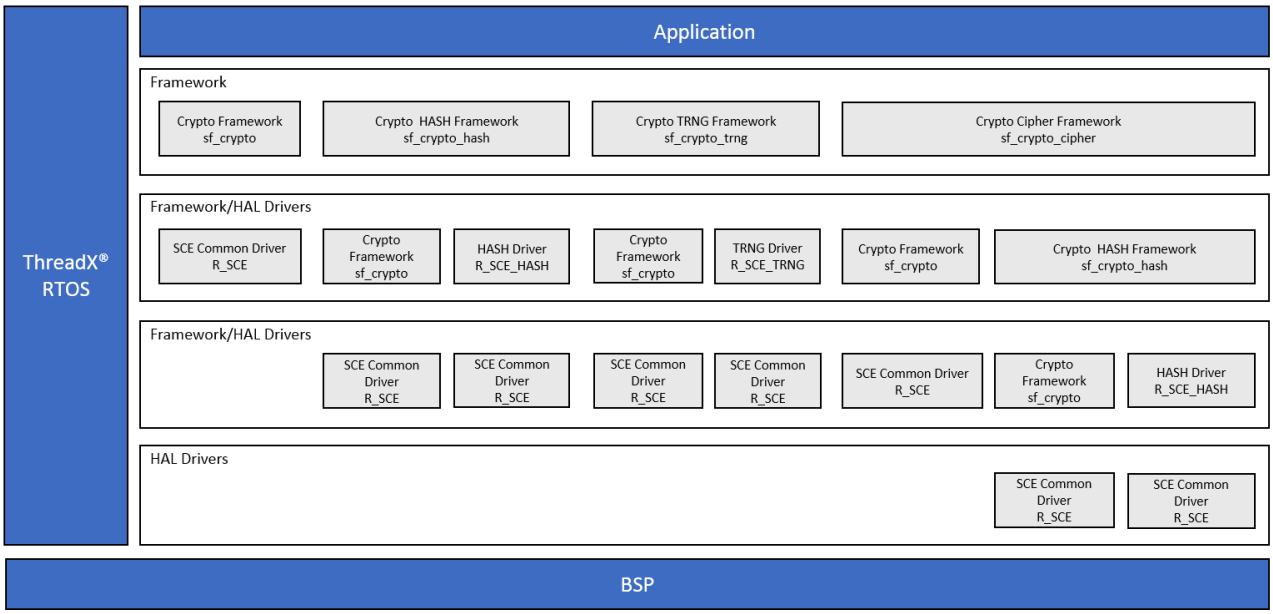
SSP のコンテキストでは、「キーのラップ」と「キーのインストール」という用語は次のように定義されます。

キーのラップ：Synergy プラットフォーム上で対称キーまたは非対称キーのペアを生成する API で、秘密キーがラップキー（暗号化キー）になります。

キーのインストール：ユーザーが生成した PC（Synergy プラットフォーム以外のシステム）上の秘密キーが Synergy プラットフォームにインストールされ（保管はなし）、ラップされた秘密キーがユーザーに返されます。

ラップキーには以下のメリットがあります。

- ラップキーは、それが生成された Synergy プラットフォーム（platform(MCU）でのみ使用可能です。
- 別の Synergy プラットフォーム（platform(MCU）に移動することはできません。
- ラップキーから元のキーを復元することはできません。



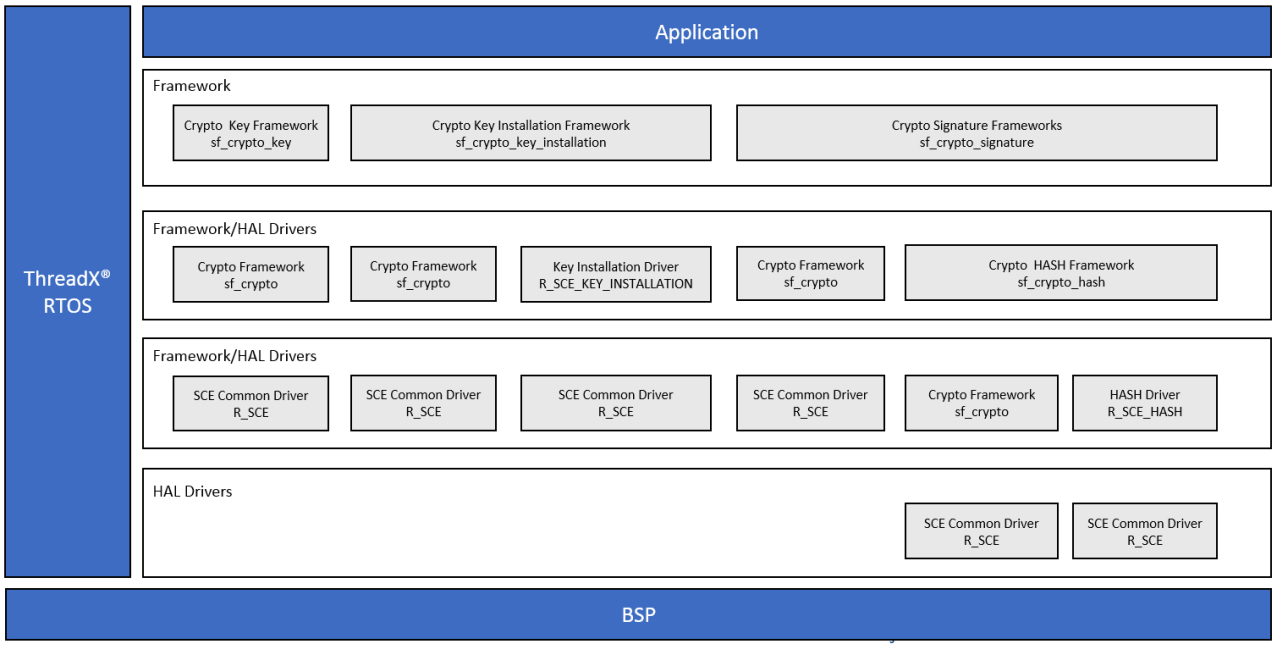


図 74:暗号化フレームワークモジュールのブロック図

4.1.18.2 暗号化フレームワークモジュール API の概要

暗号化フレームワークモジュールは、各種の暗号化サービス向けの API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

暗号化フレームワークモジュール API の要約

Function Name	Example API Call and Description
open	g_sf_crypto0.p_api->open(g_sf_crypto0.p_ctrl, g_sf_crypto0.p_cfg); This function is used to open the crypto framework module and the r_sce HAL module.

Function Name	Example API Call and Description
close	g_sf_crypto0.p_api->close(g_sf_crypto0.p_ctrl); This function is used to close the crypto framework module and the r_sce HAL module.
lock	g_sf_crypto0.p_api->lock(g_sf_crypto0.p_ctrl); This function locks shared resources for cryptography operations.
unlock	g_sf_crypto0.p_api->unlock(g_sf_crypto0.p_ctrl); Unlock shared resources for cryptography operations.
statusGet	g_sf_crypto0.p_api->statusGet(g_sf_crypto0.p_ctrl, &p_status); This function gets the status of the Crypto Framework common module.
versionGet	g_sf_crypto0.p_api->versionGet(&version); This function retrieves the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

暗号化 HASH フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_crypto_hash0.p_api->open(g_sf_crypto_hash0.p_ctrl, g_sf_crypto_hash0.p_cfg); This function is used to open the crypto hash framework module.
close	g_sf_crypto_hash0.p_api->close(g_sf_crypto_hash0.p_ctrl); This function is used to close the crypto hash framework module.
hashInit	g_sf_crypto_hash0.p_api->hashInit(g_sf_crypto_hash0.p_ctrl); This function initializes the calculation of the hash function. It has to be invoked first and only once, at the beginning of each new calculation.
hashUpdate	g_sf_crypto_hash0.p_api->hashUpdate(g_sf_crypto_hash0.p_ctrl, &p_data_in); This function calculates the hash on the data pointed to by the p_data_in pointer.
hashFinal	g_sf_crypto_hash0.p_api->hashFinal(g_sf_crypto_hash0.p_ctrl, &p_data_in, &p_size); This function finalizes the hash operation on the data pointed to by the p_data_in pointer and stores the output size in the 32-bit word defined by the p_size pointer.
versionGet	g_sf_crypto_hash0.p_api->versionGet(&version); This function retrieves the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

暗号化 TRNG フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_crypto_trng0.p_api->open(g_sf_crypto_trng0.p_ctrl, g_sf_crypto_trng0.p_cfg); This function is used to open the Crypto TRNG framework module.
close	g_sf_crypto_trng0.p_api->close(g_sf_crypto_trng0.p_ctrl); This function is used to close the Crypto TRNG framework module.
randomNumberGenerate	g_sf_crypto_trng0.p_api->randomNumberGenerate (&p_buffer); This function generates the random number and stores it in memory starting at the address defined by the p_buffer pointer.
versionGet	g_sf_crypto_trng0.p_api->versionGet(&version); This function retrieves the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

暗号化 Key フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_crypto_key0.p_api->open(g_sf_crypto_key0.p_ctrl, g_sf_crypto_key0.p_cfg); This function is used to open the crypto key framework module.
keyGenerate	g_sf_crypto_key0.p_api->keyGenerate(g_sf_crypto_key0.p_ctrl, &p_secret_key, &p_public_key); This function is used to generate a key or key pair and store it at the p_secret_key and p_public_key pointers.
close	g_sf_crypto_key0.p_api->close(g_sf_crypto_key0.p_ctrl); This function is used to close the crypto key framework module and the r_sce HAL module.
versionGet	g_sf_crypto_key0.p_api->versionGet(&version); This function retrieves the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

暗号化 Key Installation フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_crypto_key_installation.p_api->open(g_sf_crypto_key_installation.p_ctrl, g_sf_crypto_key_installation.p_cfg); This function is used to open the crypto key initialization framework module.
keyInstall2	g_sf_crypto_key_installation.p_api->keyInstall2(g_sf_crypto_key_installation.p_ctrl, p_user_key_input, p_install_key_input, p_key_data_out); This function is used to install a key using the p_user_key_input, p_install_key_input and p_key_data_out pointers.
close	g_sf_crypto_key_installation.p_api->close(g_sf_crypto_key_installation.p_ctrl); This function is used to close the crypto key installation framework module.
versionGet	g_sf_crypto_key_installation.p_api->versionGet(&version); This function retrieves the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

暗号化 Cipher フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_crypto_cipher.p_api->open(g_sf_crypto_cipher.p_ctrl, g_sf_crypto_cipher.p_cfg); This function is used to open the crypto cipher framework module.
cipherInit	g_sf_crypto_cipher.p_api->cipherInit(g_sf_crypto_cipher.p_ctrl, mode, p_key, p_params); This function is used to initialize the crypto cipher framework module.
cipherUpdate	g_sf_crypto_cipher.p_api->cipherUpdate(g_sf_crypto_cipher.p_ctrl, p_datain, p_dataout); This function performs the cipher encrypt or decrypt operation. It can be called multiple times on additional blocks of data. Result is placed in p_dataout.
cipherFinal	g_sf_crypto_cipher.p_api->cipherFinal(g_sf_crypto_signature.p_ctrl, p_datain, p_dataout); This function performs the final encrypt or decrypt operation. Result is placed in p_dataout.
cipherAadUpdate	g_sf_crypto_cipher.p_api->cipherAadUpdate(g_sf_crypto_cipher.p_ctrl, p_aad); This function updates AAD (Additional Authenticated Data) for AES GCM operation using the p_aad pointer to the AAD data and length. It can be called multiple times on additional blocks of data.

Function Name	Example API Call and Description
close	g_sf_crypto_cipher.p_api->close(g_sf_crypto_cipher.p_ctrl); This function is used to close the crypto signature framework module.
versionGet	g_sf_crypto_cipher.p_api->versionGet(&version); This function retrieves the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

暗号化 Signature フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_crypto_signature.p_api->open(g_sf_crypto_signature.p_ctrl, g_sf_crypto_signature.p_cfg); This function is used to open the crypto signature framework module.
contextInit	g_sf_crypto_signature.p_api->contextInit(g_sf_crypto_signature.p_ctrl, mode, p_params, p_key); This function is used to initialize the crypto signature framework module.
signUpdate	g_sf_crypto_signature.p_api->signUpdate(g_sf_crypto_signature.p_ctrl, p_message); This function performs the signature update operation. It can be called multiple times to accumulate the message to be signed.

Function Name	Example API Call and Description
verifyUpdate	g_sf_crypto_signature.p_api->verifyUpdate(g_sf_crypto_signature.p_ctrl, p_message); This function performs the signature verification update operation. It can be called multiple times to accumulate the message whose signature is to be verified.
signFinal	g_sf_crypto_signature.p_api->signFinal(g_sf_crypto_signature.p_ctrl, p_message, p_dest); This function generates the signature and writes it to the destination.
verifyFinal	g_sf_crypto_signature.p_api->verifyFinal(g_sf_crypto_signature.p_ctrl, p_signature, p_message); This function performs the signature verify operation.
close	g_sf_crypto_signature.p_api->close(g_sf_crypto_signature.p_ctrl); This function is used to close the crypto signature framework module.
versionGet	g_sf_crypto_signature.p_api->versionGet(& version); This function retrieves the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

フレームワーク API は一般的な SSP エラーコードを返し、それに加えてローレベルモジュールからの以下のエラーコードを返すことがあります。暗号化フレームワーク固有のエラーコードもあり、この表の最後に記載されています。

ステータス戻り値

Name	Description
SSP_ERR_CRYPTO_CONTINUE	Continue executing function
SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT	Hardware resource busy
SSP_ERR_CRYPTO_SCE_FAIL	Internal I/O buffer is not empty
SSP_ERR_CRYPTO_SCE_HRK_INVALID_INDEX	Invalid index
SSP_ERR_CRYPTO_SCE_RETRY	Retry
SSP_ERR_CRYPTO_SCE_VERIFY_FAIL	Verify failed
SSP_ERR_CRYPTO_SCE_ALREADY_OPEN	Crypto Module is already opened
SSP_ERR_CRYPTO_NOT_OPEN	Hardware module is not initialized
SSP_ERR_CRYPTO_UNKNOWN	Some unknown error occurred
SSP_ERR_CRYPTO_NULL_POINTER	Null pointer input as a parameter
SSP_ERR_CRYPTO_NOT_IMPLEMENTED	Algorithm/size not implemented
SSP_ERR_CRYPTO_RNG_INVALID_PARAM	An invalid parameter is specified
SSP_ERR_CRYPTO_RNG_FATAL_ERROR	A fatal error occurred
SSP_ERR_CRYPTO_INVALID_SIZE	Size specified is invalid
SSP_ERR_CRYPTO_INVALID_STATE	Function used in an valid state
SSP_ERR_CRYPTO_ALREADY_OPEN	Control block is already opened
SSP_ERR_CRYPTO_INSTALL_KEY_FAILED	Specified input key is invalid
SSP_ERR_CRYPTO_AUTHENTICATION_FAILED	Authentication failed
Crypto Framework Specific Error Codes	

Name	Description
SSP_ERR_CRYPTO_COMMON_NOT OPENED	Crypto Framework Common is not opened.
SSP_ERR_CRYPTO_HAL_ERROR	Crypto HAL module returned an error.
SSP_ERR_CRYPTO_KEY_BUF_NOT_EN OUGH	Key buffer size isn't large enough to generate a key
SSP_ERR_CRYPTO_BUF_OVERFLOW	Attempt to write data larger than buffer.
SSP_ERR_CRYPTO_INVALID_OPERATIO N_MODE	Invalid operation.
SSP_ERR_MESSAGE_TO_LONG	Message for RSA encryption is too long.
SSP_ERR_RSA_DECRYPTION_ERROR	RSA Decryption error.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.18.3 暗号化フレームワークモジュールの動作の概要

次の注意事項は、SF_CRYPTO, SF_CRYPTO_TRNG, SF_CRYPTO_HASH, SF_CRYPTO_KEY, SF_CRYPTO_KEY_INSTALLATION, SF_CRYPTO_CIPHER、SF_CRYPTO_SIGNATURE の各モジュールに適用されます。

エンディアン構成パラメータ

- Synergy コンフィギュレータでは、暗号化フレームワークモジュールなしで暗号化 HAL ドライバーのみが選択されている場合、エンディアンは構成可能なパラメータであり、デフォルトで *ビッグエンディアン*に設定されます。これは、以前のリリースでサポートされていたモードです。
- Synergy コンフィギュレータでは、暗号化フレームワークモジュールが選択されている場合（対応する HAL コンポーネントが自動的に含まれます）、エンディアンフラグは、バイト配列フォーマットのみがサポートされる *リトルエンディアン*モードにロックされます。
- 暗号化フレームワークレイヤ API では、入力および出力データに対してバイト配列のみがサポートされます。
- 暗号化 HAL API では、入力および出力データに対してワード /32 ビット (uint32_t) 配列がサポートされます。
- バイト配列フォーマットのデータを持つユーザーは、暗号化フレームワークモジュールを使用する必要があります。
- 対応するフレームワーク API のない暗号化 HAL API を使用している既存のプロジェクトでは、ビッグエンディアンデータが使用されている可能性があり、その場合は、リトルエンディアンに変換してフレームワーク API を使用するか、引き続き既存の HAL API を直接使用する必要があります。
- HAL API を直接使用する場合、ユーザーは（uint32_t）としてキャストされたデータが、使用されている HAL モジュールのエンディアンと一致することを確認する必要があります。

データバッファのアラインメント

- 入力バッファに渡すために割り当てられたすべてのデータバッファは、ワード境界でアラインする必要があります。

ブロッキングコール

- すべての暗号化フレームワーク API はブロッキング呼び出しです。

SF CRYPTO フレームワークサービス

- SF CRYPTO フレームワークモジュールには、以下が含まれます。
  - ハードウェアの初期化、および暗号化モジュール間のリソース同期のための SF_CRYPTO。
  - 真性乱数生成のための SF_CRYPTO_TRNG。
  - メッセージダイジェスト生成のための SF_CRYPTO_HASH。ハッシュ操作の終了処理の前にデータをチャンク単位で処理する機能を提供します。
  - RSA、AES、ECC 用の SF_CRYPTO_KEY。RSA および ECC 向けに、プレーンテキストキーに加えてラップキーを生成するオプションを提供します。AES ではラップキーのみがサポートされます。
  - RSA、AES、ECC キーのキーインストールサービスを提供する SF_CRYPTO_KEY_INSTALLATION。
  - AES および RSA キーの暗号化および復号化サービスを提供する SF_CRYPTO_CIPHER。
  - RSA 署名生成および検証サービスを提供する SF_CRYPTO_SIGNATURE。

VersionGet API

この API は、モジュールが開かれる前であっても、いつでも呼び出すことができます。

SF CRYPTO フレームワークモジュールの概要説明

SF CRYPTO フレームワークはハイレベルの API を提供し、sf_crypto. に実装されます。SF CRYPTO フレームワークは、セキュア暗号化エンジン（SCE）HAL モジュールを通じてフレームワーク暗号化サービスの基盤を提供します。

SF CRYPTO フレームワークモジュールの特長

- SF CRYPTO フレームワークは、基礎となるハードウェアセキュア暗号化エンジンを開き、初期化します。
- 他の暗号化フレームワークモジュール SF_CRYPTO_TRNG, SF_CRYPTO_HASH, SF_CRYPTO_KEY, SF_CRYPTO_CIPHER, SF_CRYPTO_SIGNATURE、SF_CRYPTO_KEY_INSTALLATION. の共有リソースにアクセスするためのサービスを提供します。

SF CRYPTO フレームワークモジュールの動作に関する注意事項

- sf_crypto_api_t::open API が初期化中にコールされます。これは、[Auto Initialization] 構成オプションがデフォルトで有効になっているためです。
- SF_CRYPTO_XXX モジュールでは sf_crypto_api_t::lock API と sf_crypto_api_t::unlock API が使用されます。ただし、アプリケーションが HAL API を直接コールする場合は、HAL モジュールをコールする直前に sf_crypto_api_t::lock API を使用します。
- sf_crypto_api_t::unlock API は、HAL モジュールのコールから戻った直後に使用します。
- sf_crypto_api_t::statusGet API は、入力パラメータとして制御ブロック block/p_ctrl を必要とします。このため、SF_CRYPTO モジュールの open API が呼び出された後にのみ呼び出す必要があります。

SF CRYPTO フレームワークモジュールの構成設定

Configuration parameter	Description
uint32_t wait_option	Wait option for RTOS service calls Defines how the service behaves if there is not enough memory available. The wait options are defined as follows: TX_NO_WAIT (0x00000000) T X_WAIT_FOREVER (0xFFFFFFFF) timeout value (0x00000001 through 0xFFFFFFFE) Selecting TX_NO_WAIT results in an immediate return from this service regardless of whether or not it was successful. This is the only valid option if the service is called from initialization. Selecting TX_WAIT_FOREVER causes the calling thread to suspend indefinitely until enough memory is available. Selecting a numeric value (1-0xFFFFFFFE) specifies the maximum number of timer-ticks to stay suspended while waiting for the memory. Default set through ISDE is TX_WAIT_FOREVER.
crypto_instance_t * p_lower_lvl_crypto	Pointer to a low-level Crypto engine HAL driver instance. Configured by Synergy Configurator.
void const * p_extend	Extension parameter for hardware specific settings. Configured by Synergy Configurator.
void const * p_context	Placeholder for user data. For future expansion.
void * p_memory_pool	Byte pool address. Configured by Synergy Configurator.

Configuration parameter	Description
uint32_t memory_pool_size	Byte pool size. Default set by ISDE is 128 bytes. Caller to allocate pool based on the number of SF_CRYPTO_XXX module instances created. Recommended sizes: In addition to the default byte pool: NOTE: An additional 12 bytes per instance is needed for RTOS housekeeping. 24 + 12 bytes per instance of SF_CRYPTO_KEY module if RSA Key is required. 264 +12bytes per instance of SF_CRYPTO_KEY module if AES Key is required. SF_CRYPTO_KEY module if AES Key is required. 112 +12 bytes per instance of SF_CRYPTO_HASH module. 1200 bytes per instance of SF_CRYPTO_CIPHER module for RSA algorithm is required. 600 bytes per instance of SF_CRYPTO_CIPHER module for AES algorithm is required. 1450 bytes per instance of SF_CRYPTO_SIGNATURE module.

Configuration parameter	Description
sf_crypto_close_option_t close_option	Close option Selects how the SCE module can be closed. SF_CRYPTO_CLOSE_OPTION_DEFAULT - Close the module only if none of the other SF_CRYPTO_XXX modules are opened. Note: This is the default setting. SF_CRYPTO_CLOSE_OPTION_FORCE_CLOSE - Close the module regardless of SF_CRYPTO_XXX modules status. Note: With either option, It is the responsibility of the caller to ensure that the SF CRYPTO module is not closed when any of the other Crypto Framework modules are open.

SF CRYPTO フレームワークモジュールの制限事項

他の暗号化フレームワークモジュールが開いている場合はSF CRYPTOモジュールが閉じられないようにすることは、コール側の責任です。

アプリケーションでの SF CRYPTO フレームワークモジュールの使用

アプリケーションで SF CRYPTO フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

- SF CRYPTO モジュールを使用するには：
  - 構成パラメータ（上記の表で指定されています）をアプリケーションのニーズに合わせて確実に設定します。
  - sf_crypto_api_t::open API を使用して、SCE 共通ドライバーで SF_CRYPTO と SCE HAL モジュール（R_SCE）を初期化します。これは、[Auto Initialize] 設定がデフォルトのままの場合、Synergy コンフィギュレータによって実行されます。
  - リトルエンディアンモードがデフォルトで設定されます。これはロックされており、構成できません。
  - open 関数は、モジュールが閉じられるまで再度呼び出すことはできません。
  - 暗号化フレームワークサービスとHW SCEを終了するにはsf_crypto_api_t::close APIを使用します。

SF CRYPTO TRNG フレームワークモジュールの概要説明

SF CRYPTO TRNG フレームワークはハイレベルの API を提供し、sf_crypto_trng. に実装されます。SF CRYPTO TRNG フレームワークは、セキュア暗号化エンジン（SCE）HAL モジュールを通じて真性乱数生成サービスを提供します。

SF CRYPTO TRNG フレームワークモジュールの特長

- SF CRYPTO TRNG フレームワークモジュールは、基礎となるセキュア暗号化エンジンへの TRNG HAL インタフェースを使用します。
- SF CRYPTO フレームワークモジュールを通じて共有リソースにアクセスします。

SF TRNG CRYPTO フレームワークモジュールの構成設定

Configuration Parameter	Description
sf_crypto_instance_t * p_lower_lvl_common;	Pointer to a low-level Crypto engine HAL driver instance. Configured by Synergy Configurator.
trng_instance_t * p_lower_lvl_instance;	Pointer to a TRNG HAL driver instance. Configured by Synergy Configurator.
void * p_extend;	Extension parameter for hardware specific settings. For future use.

構成は、TRNG HAL ドライバーの ISDE を通じて設定されます。

r_sce_trng 上の TRNG HAL モジュールの構成パラメータ

Configuration parameter	Description
uint32_t num_attempts	Number of attempts within which a true random number is to be generated. Set to 2 by default.

SF CRYPTO TRNG フレームワークモジュールの制限事項

なし。

SF CRYPTO HASH フレームワークモジュールの概要説明

SF CRYPTO HASH フレームワークはハイレベルの API を提供し、sf_crypto_hash. に実装されます。SF CRYPTO HASH フレームワークは、セキュア暗号化エンジン（SCE）HAL モジュールを通じてハッシュおよびメッセージダイジェストサービスを提供します。

サポートされるハッシュ関数は、MD5、SHA-1、SHA-224、SHA-256 です。

FIPS セキュアハッシュ標準から :

すべてのアルゴリズムは反復的な一方向ハッシュ関数で、メッセージを処理して*メッセージダイジェスト*と呼ばれる凝縮表現を生成できます。これらのアルゴリズムより、メッセージの整合性を判定できます。メッセージに変更を加えると、非常に高い確率で、異なるメッセージダイジェストが生成されます。この特性はデジタル署名とメッセージ認証コードの生成および検証と、乱数またはビットの生成に役立ちます。

サポートされるハッシュ関数については、メッセージサイズを $<2^{64}$ ビットにする必要があります。

メッセージダイジェストサイズは次のとおりです。

MD5：16 バイト

SHA-1: 20 バイト

SHA-224: 28 バイト

SHA-256: 32 バイト

SF CRYPTO HASH フレームワークモジュールの特長

- SF CRYPTO HASH フレームワークは、基礎となるセキュア暗号化エンジンを利用して HASH サービスを提供します。
- このモジュールは、以下のような HAL ドライバーに対する機能強化を提供します。
  - HASH 値の初期化
  - ハッシュ操作の終了処理の前に sf_crypto_hash_api_t::hashUpdate API を通じてデータのチャンクを処理します。
  - 最終 HASH 処理の最終ブロックのフォーマットを設定します。

SF CRYPTO HASH フレームワークモジュールの構成設定

Configuration Parameter	Description
sf_crypto_hash_type_t hash_type;	HASH algorithm type. Select MD5, SHA1, SHA 224 or SHA256. Please refer to the header file for the the definitions.
crypto_instance_t * p_lower_lvl_crypto	Pointer to a low-level Crypto engine HAL driver instance. Configured by Synergy Configurator.
hash_instance_t * p_lower_lvl_instance;	pointer to HASH lower-level module instance. Configured by ISDE.
void * p_extend	Extension parameter for hardware specific settings. Optional. Configured by Synergy Configurator.

SF CRYPTO HASH フレームワークモジュールの制限事項

なし

SF CRYPTO KEY フレームワークモジュールの概要説明

SF CRYPTO KEY フレームワークはハイレベルの API を提供し、sf_crypto_key. に実装されます。SF CRYPTO KEY フレームワークは、セキュア暗号化エンジン（SCE）HAL モジュールを通じて暗号キー生成サービスを提供します。

ラップキーは、暗号化キー、キーハンドル、ラップキーなどの異なる名前で参照されることがあります。Synergy プラットフォームでのキーのラップは安全とみなされます。これらのキーはプラットフォームの外部で使用できないためです。

SF CRYPTO KEY フレームワークモジュールの特長

SF CRYPTO KEY モジュールのサービスを使用して次のキータイプを生成できます。

- 標準および CRT フォーマットの RSA 2048 ビット、1024 ビットプレーンテキスト / 未加工キー。
- RSA 2048 ビット、1024 ビット標準フォーマットのラップ秘密キー（公開キーはプレーンテキスト）。
- ECB、CBC、CTR および GCM チェーンモード用の AES 128 ビット、192 ビットおよび 256 ビットラップキー。
- XTS チェーンモード用の AES 128 ビットおよび 256 ビットラップキー。
- ECC 192 ビット、256 ビットのプレーンテキストキーおよびラップキー（ラップキーは秘密キーのみが対象）。

SF CRYPTO KEY フレームワークモジュールの動作に関する注意事項

AES ラップキーのサイズは次のとおりです。

```
/** Return Wrapped AES secret key size in bytes for a 128-bit AES Key */
#define AES128_WRAPPPED_SECRET_KEY_SIZE_BYTES  (36U)

/** Return Wrapped AES secret key size in bytes for a 192-bit AES Key */
#define AES192_WRAPPPED_SECRET_KEY_SIZE_BYTES  (52U)

/** Return Wrapped AES secret key size in bytes for a 256-bit AES Key */
#define AES256_WRAPPPED_SECRET_KEY_SIZE_BYTES  (52U)

/** Return Wrapped AES-XTS secret key size in bytes for a 128-bit AES XTS Mode K
ey */
#define AES_XTS_128_WRAPPPED_SECRET_KEY_SIZE_BYTES  (52U)

/** Return AES-XTS secret key size in bytes for a 256-bit AES XTS Mode Key */
#define AES_XTS_256_WRAPPPED_SECRET_KEY_SIZE_BYTES  (84U)
```

RSA キー：

keyGenerate API によって生成される RSA プレーンテキストキーのフォーマットは次のとおりです。

RSA 公開キー

バイト 0 ～バイト 3: 公開キー指数

バイト 4：RSA モジュールの開始

（RSA 1024 ビットキーの場合は 128 バイト、RSA 2048 ビットキーの場合は 256 バイト）

標準フォーマットの RSA 秘密キー

バイト 0: 秘密キー指数（RSA 1024 ビットキーの場合は 128 バイト、RSA 2048 ビットキーの場合は 256 バイト）

RSA 係数が続きます。(RSA 1024 ビットキーの場合は 128 バイト、RSA 2048 ビットキーの場合は 256 バイト)

CRT フォーマットの RSA 秘密キー

コンポーネントはバイト 0 を exponent2 として次の順序で並べられます。

exponent2 // 第 2 係数の CRT 指数、正の整数

prime2 // 第 2 係数、正の整数

exponent1 // 第 1 係数の CRT 指数、正の整数

prime1 // 第 1 係数、正の整数

coefficient // (第 1) CRT 係数、正の整数

keyGenerate API によって生成される RSA ラップキーのフォーマットは次のとおりです。

RSA 公開キーは常にプレーンテキストです。

バイト 0 ～バイト 3: 公開キー指数

バイト 4：RSA モジュールの開始

(RSA 1024 ビットキーの場合は 128 バイト、RSA 2048 ビットキーの場合は 256 バイト)

標準フォーマットの RSA 秘密キー

バイト 0：秘密キー指数はラップされます(長さは、RSA 1024 ビットキーでは 148 バイト、RSA 2048 ビットキーでは 276 バイト)。

プレーンテキストの RSA モジュールが続きます(長さは、RSA 1024 ビットキーでは 128 バイト、RSA 2048 ビットキーでは 256 バイト)。

SF CRYPTO KEY フレームワークモジュールの構成設定

Configuration parameter	Description
sf_crypto_key_type_t key_type	Key type to be generated. Plain text or Wrapped keys. Please refer to the header file for the the definitions.
sf_crypto_key_size_t key_size	Key size to be generated. Please refer to the header file for the the definitions.

Configuration parameter	Description
sf_crypto_data_handle_t domain_params;	Pointer to domain parameters for the requested key type. Structure contains, pointer to the data (contains the domain data) and data length. Structure contains the domain data in the order a, b, p, n for ECC as defined in FIPS186-3 and data length. To be filled appropriately for ECC and set to NULL for RSA and AES Key types, since this parameter only applies to the ECC function.
sf_crypto_data_handle_t generator_point	Pointer to the generator base point of curve in the order Gx, Gy for ECC (where Gx and Gy are x and y coordinates respectively) and data length. To be set to NULL for RSA and AES key types.
sf_crypto_instance_t * p_lower_lvl_crypto_common	Pointer to a Crypto Framework common instance. Configured by Synergy Configurator.
void const * p_extend	Extension parameter for hardware specific settings (Reserved for future use). Configured by Synergy Configurator.

SF CRYPTO KEY フレームワークモジュールの制限事項

現在、AES ではラップキーのみがサポートされます。

SF Crypto Signature フレームワークモジュールの概要説明

SF CRYPTO Signature フレームワークはハイレベルの API を提供し、sf_crypto_signature. に実装されます。このモジュールは、ハードウェアレベルでの暗号化操作を行うために Synergy HAL ドライバーとのインタフェースを備えた、SSP フレームワークレイヤーにあります。ここで指定されるモジュール関数は、入力として使用されるメッセージに対して署名と検証を行うために使用されます。

SF Crypto Signature モジュールの特長

- このモジュールは、現在 RSA アルゴリズム用の署名生成と署名検証をサポートしています。標準フォーマットのプレーンテキスト、標準フォーマットのラップ秘密キー、および CRT プレーンテキストキーに対して 1024 ビットおよび 2048 ビットのキー長をサポートしています。
- 署名方式として、RSASSA-PKCS1 v1.5 をサポートしています。入力メッセージを、署名または検証のために未加工メッセージとして渡すか、または署名あるいは検証前に PKCS1 v1.5 によってエンコードおよびパディングすることができます。

- SHA1、SHA224、SHA256 が、PKCS1 v1.5 エンコードおよびパディング方式のハッシュアルゴリズムとしてサポートされています。
- このモジュールでは、ブロックサイズよりも小さなデータに対して署名生成および検証を行うことができます。
- 一度にすべてのデータが使用できない場合は、更新 API を使用して、入力データのチャンクを蓄積し、メッセージ全体が収集された時点で、最終的にそのメッセージに対して署名や検証を行うことができます。
- このモジュールでは、メモリの割り当てや割り当て解除に関与しない、コストの少ない API コールを使用して、署名操作と検証操作を切り替えることができます。

SF Crypto Signature フレームワークモジュールの動作に関する注意事項

1) Signature フレームワークモジュールの動作モード：
   a) SF_CRYPTO_SIGNATURE_MODE_SIGN
   b) SF_CRYPTO_SIGNATURE_MODE_VERIFY
2) 署名または検証操作を実行するための Signature Framework API に対する入力メッセージのフォーマット：
   a) SF_CRYPTO_SIGNATURE_MESSAGE_OPERATION_NONE
   b) SF_CRYPTO_SIGNATURE_MESSAGE_OPERATION_RSA_SHA1_PKCS1_1_5
   c) SF_CRYPTO_SIGNATURE_MESSAGE_OPERATION_RSA_SHA224_PKCS1_1_5
   d) SF_CRYPTO_SIGNATURE_MESSAGE_OPERATION_RSA_SHA256_PKCS1_1_5

SF CRYPTO Signature フレームワークモジュールの構成設定

Configuration parameter	Description
sf_crypto_key_type_t key_type;	Type of Key RSA plain text or wrapped. Please refer to the header file for the the definitions.
sf_crypto_key_size_t key_size	Size of Key RSA 1024-bit /2048-bit key. Please refer to the header file for the the definitions.
sf_crypto_hash_instance_t * p_lower_lvl_sf_crypto_hash	Pointer to the Crypto Hash framework module instance structure. Configured by Synergy Configurator.
sf_crypto_instance_t * p_lower_lvl_crypto_common	Pointer to Crypto Common module instance. Configured by Synergy Configurator.
void const * p_extend	Extension parameter for hardware specific settings (Reserved for future use). Configured by Synergy Configurator.

SF CRYPTO Signature フレームワークモジュールの制限事項

なし

SF Crypto Cipher フレームワークモジュールの概要説明

SF CRYPTO Cipher フレームワークはハイレベルの API を提供し、sf_crypto_cipher. に実装されます。このモジュールは SSP フレームワークレイヤーにあり、セキュア暗号化エンジン（SCE）HAL モジュールを通じて暗号化および復号化サービスを提供します。

SF Crypto Cipher モジュールの特長

- このモジュールは、現在 AES および RSA アルゴリズム用の暗号化および復号化をサポートしています。
- このモジュールでは、データがチャンクで使用できる場合に、最後のチャンクまたはデータ全体が収集されたときに sf_crypto_cipher_api_t::cipherUpdate API および final() を介してデータの暗号化および復号化が可能です。
- 特定のキータイプおよびキーサイズに対してモジュールがオープンされると、sf_crypto_cipher_api_t::cipherInit API をコールすることで暗号モードおよび復号モードを切り替えることができます。

AES アルゴリズムのサポート：

- 次のチェーンモードに対する AES 128 ビット、192 ビット、256 ビットのプレーンテキストキーおよびラップキー。
  - ECB（Electronic Code Book）
  - CBC（Cipher Block Chaining）
  - CTR（カウンタモード）
  - GCM（ガロアカウンタモード）
- XTS（変更を加えられた XEX ベースのコードブロックモード。暗号文窃取を使用）用の AES 128 ビットおよび 256 ビットのプレーンテキストキーおよびラップキーがサポートされます。
- AES GCM 操作に対して指定される IV は、96 ビット、または 96 ビット IV を 128 ビットにフォーマットしたものです。
  - 例：
  - 96 ビット IV：94c1935afc061cbf254b936f
  - 96 ビット IV を 128 ビットにフォーマットしたもの：94c1935afc061cbf254b936f00000001
- ECB モードと CBC モードでは PKCS#7 パディング方式がサポートされています。この方式は、データが次の場合に使用できます。
- データが AES ブロックサイズの倍数と一致する場合、どのモードでもパディングオプションはサポートされません。
  - GCM の場合は、データがブロックサイズの倍数と一致しなくてもパディングオプションはサポートされません。

RSA アルゴリズムのサポート：

- RSA 1024 ビットおよび 2048 ビットのプレーンテキスト標準フォーマット、プレーンテキスト CRT フォーマット、および標準フォーマットのラップキーがサポートされます。
- RSA 暗号化操作には RSA 公開キーが、RSA 復号化操作には RSA 秘密キーが必要です。
- RSAES-PKCS1-v1_5（RSA 暗号化方式）

- RSA-PKCS1 v1.5 エンコーディング / パディング方式がサポートされます。
- 入力された未加工のメッセージは、暗号化のためにエンコードおよびフォーマットされます。そのためには、選択されたキーのモジュールの長さを k として、メッセージが k-11 未満であることが必要です。

- エンコードおよびフォーマットされるブロック（選択されたキーのモジュラスと完全に同じサイズ）を暗号化および復号化する必要がある場合は、パディングオプションを選択しないでください。

SF Crypto Cipher フレームワークモジュールの動作に関する注意事項

SF Crypto Cipher フレームワークモジュールの構成設定

Configuration Parameter	Description
sf_crypto_key_type_t key_type	Key type for cipher operation: AES or RSA plain text or wrapped. Please refer to the header file for the the definitions.
sf_crypto_key_size_t key_size	Key size for cipher operation: AES 128-bit/192-bit/256-bit key or RSA 1024-bit /2048-bit key. Please refer to the header file for the the definitions.
sf_crypto_cipher_mode_t cipher_chaining_mode	Applicable only to AES algorithm: ECB, CBC, CTR, XTS or GCM chaining modes Set to ECB for RSA. Please refer to the header file for the the definitions.
sf_crypto_instance_t * p_lower_lvl_crypto_common	Pointer to Crypto Framework Common module instance. Configured by Synergy Configurator.
sf_crypto_trng_instance_t * p_lower_lvl_crypto_trng	Pointer to Crypto Framework TRNG module instance. Configured by Synergy Configurator.
void const * p_extend	Extension parameter for hardware specific settings.Optional. Configured by Synergy Configurator.

SF CRYPTO Cipher フレームワークモジュールの制限事項

- AES XTS モードでは、32 ビット（4 バイト）の倍数の長さのみがサポートされます。

SF Crypto Key Installation フレームワークモジュールの概要説明

SF Crypto Key Installation フレームワークはハイレベルの API を提供し、sf_crypto_key_installation. に実装されます。SF Crypto Key Installation フレームワークは、セキュア暗号化エンジン（SCE）HAL モジュールを通じてキーインストールサービスを提供します。

注 :1. この API はユーザーにラップキー(ラップキーのフォーマットおよびサイズの詳細については、SF CRYPTO KEY モジュールのセクションを参照）を返しますが、後で使用するためにキーを保存することはありません。後で使用するためにラップキーを不揮発性メモリに格納するのは、アプリケーションの責任です。

*注 :2. このフレームワークモジュールは、一切の拡張機能なしで Crypto HAL モジュールと同じレベルのサービスを提供します。これはスレッドセーフ操作のため、およびフレームワークレイヤーにおける暗号化サポートの完全を期すために用意されています。*SF Crypto Key Installation フレームワークモジュールの特長

- SF Crypto Key Installation フレームワークモジュールは、基礎となるセキュア暗号化エンジンへの KEY INSTALLATION HAL インタフェースを使用します。
- SF CRYPTO フレームワークモジュールを通じて共有リソースにアクセスします。
- このモジュールでは、次のキーに対してキーのインストールがサポートされています。
  - RSA：
    1024 ビットおよび 2048 ビットのプレーンテキスト標準フォーマットキー
    注：CRT キーはサポートされません。
  - AES：
    ECB、CBC、CTR、GCM チェーンモード用の 128 ビット、192 ビット、256 ビットプレーンテキストキー。
    XTS チェーンモード用の 128 ビットおよび 256 ビットプレーンテキストキー。,
  - ECC：
    192 ビットおよび 256 ビットプレーンテキストキー。
- ユーザーおよびインストールキーは、指定したフォーマットでキーインストール API に提供する必要があります。
- インストール後、キーインストールサービスによってラップキーが呼び出し側に返され、以降そのデバイスでインストールキーを使用できます。

注 : キーインストールのためのユーザーキーの準備に関する詳細は、付録 A に記載されています。

SF Crypto Key Installation フレームワークモジュールの構成設定

Configuration parameter	Description
sf_crypto_key_type_t key_type;	Type of key to be installed. The prepared key will be of the encrypted type. Please refer to the header file for the definitions.
sf_crypto_key_size_t key_size;	Size of key to be installed. Please refer to the header file for the the definitions.
sf_crypto_instance_t * p_lower_lvl_common;	Pointer to a Crypto Framework common instance. Configured by Synergy Configurator.
key_installation_instance_t * p_lower_lvl_instance;	Pointer to Crypto Key Install HAL instance Configured by Synergy Configurator.

Configuration parameter	Description
void const * p_extend;	Extension parameter for hardware specific settings. For future use.

4.1.18.4 アプリケーションへの暗号化フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに暗号化フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

暗号化フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。(暗号化フレームワークモジュールのデフォルト名は g_sf_crypto0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

暗号化フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_crypt0 Crypto Framework on sf_crypto	Threads	New Stack> Framework> Crypto > Crypto Framework on sf_crypto
g_sf_crypt_key0 Crypto Key Framework on sf_crypto_key	Threads	New Stack> Framework> Crypto > Crypto Key Framework on sf_crypto_key
g_sf_crypt0 Crypto TRNG Framework on sf_crypto_trng	Threads	New Stack> Framework> Crypto > Crypto TRNG Framework
g_sf_crypt0 Crypto HASH Framework on sf_crypto_hash	Threads	New Stack> Framework> Crypto > Crypto HASH Framework
g_sf_crypt_key0 Crypto Key Installation Framework on sf_crypto_key_installation	Threads	New Stack> Framework> Crypto > Crypto Key Installation Framework on sf_crypto_key_installation

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_crypt0 Crypto cipher Framework on sf_crypto_cipher	Threads	New Stack> Framework> Crypto > Crypto cipher Framework
g_sf_crypt0 Crypto signature Framework on sf_crypto_signature	Threads	New Stack> Framework> Crypto > Crypto signature Framework

次の図に示すように、sf_crypto の暗号化フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

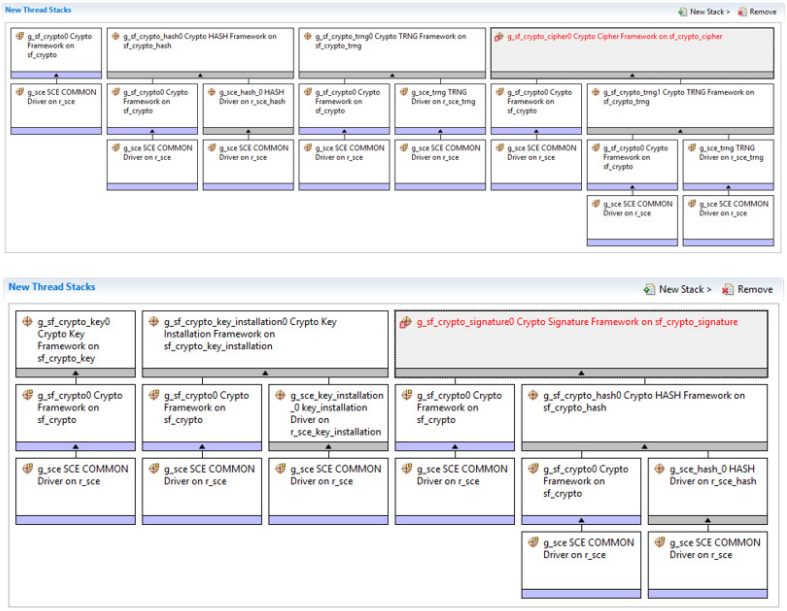


図 75:暗号化フレームワークモジュールスタック

4.1.18.5 暗号化フレームワークモジュールの構成

ユーザーは必要な動作に合わせて暗号化フレームワークモジュールを構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより

構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成表の設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_crypto 上の暗号化フレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Select if code for parameter checking is to be included in the build.
Name	g_sf_crypto0	Module name.
Wait time	TX_WAIT_FOREVER	Value must be a non-negative integer.
Byte Pool Size	128	Specify the size of the byte pool in bytes.
Auto Initialization	Enable, Disable Default: Enable	Select if sf_crypto will be initialized during startup.
Force Closure Support	Default, Force Close Default: Default	Select Force Close to close the crypto module regardless of the status of submodules.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_crypto_hash 上の暗号化 HASH フレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Select if code for parameter checking is to be included in the build.

ISDE Property	Value	Description
Instance Name	g_sf_crypto_hash0	Module name.
Name of generated initialization function	sf_crypto_hash_init0	Select the name of the generated initialization function.
Auto Initialization	Enable, Disable Default: Enable	Select if sf_crypto_hash will be initialized during startup.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_crypto_trng 上の暗号化 TRNG フレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Select if code for parameter checking is to be included in the build.
Name	g_sf_crypto_trng0	Module name.
Name of generated initialization function	sf_crypt_trng_init0	Select the name of the generated initialization function.
Auto Initialization	Enable, Disable Default: Enable	Select if sf_crypto_trng will be initialized during startup.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_crypto_cipher 上の暗号化 Cipher フレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Select if code for parameter checking is to be included in the build.
Name	g_sf_crypto_cipher0	Module name.
Key type	RSA Plain text, RSA CRT Plain text, RSA Wrapped, AES Wrapped, ECC Plain text, ECC Wrapped Default: RSA Plain text	Select the key type.
Key size	RSA 1024-bits, RSA 2048-bits, AES 128-bits, AES XTS 128-bits, AES 192-bits, AES 256-bits, AES XTS 256-bits, ECC 192-bits, ECC 256-bits Default: RSA 2048-bits	Select the key size.
Cipher chaining mode	ECB, CBC, CTR, GCM, XTS Default: ECB	Select the cipher chaining mode.
Name of generated initialization function	sf_crypto_cipher_init0	Select the name of the generated initialization function.
Auto Initialization	Enable, Disable Default: Enable	Select if sf_crypto_cipher will be initialized during startup.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_crypto_key 上の暗号化 Key フレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Select if code for parameter checking is to be included in the build.
Name	g_sf_crypto_key0	Module name.
Key type	RSA Plain text, RSA CRT Plain text, RSA Wrapped, AES Wrapped, ECC Plain text, ECC Wrapped Default: RSA Plain text	Select the key type. For AES, the byte pool size defined in sf_crypto must be >= 280 bytes.
Key size	RSA 1024-bits, RSA 2048-bits, AES 128-bits, AES XTS 128-bits, AES 192-bits, AES 256-bits, AES XTS 256-bits, ECC 192-bits, ECC 256-bits Default: RSA 2048-bits	Select the key size.
Name of generated initialization function	sf_crypto_key_init0	Select the name of the generated initialization function.
Name of Domain Parameter (Applicable only for ECC)	sf_crypto_key_domain_params0	Specify the name of the ECC domain partner.
Name of Generator Point (Applicable only for ECC)	sf_crypto_key_generator_point0	Specify the name of the ECC generator point.
Auto Initialization	Enable, Disable Default: Enable	Select if sf_crypto_key_will be initialized during startup.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_crypto_key_installation 上の暗号化 Key Installation フレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Select if code for parameter checking is to be included in the build.
Name	g_sf_crypto_key_installation 0	Module name.
Key type	Encrypted RSA Key, Encrypted AES Key, Encrypted ECC Key Default: Encrypted RSA Key	Select the key type.
Key size	RSA 1024-bits, RSA 2048-bits, AES 128-bits, AES XTS 128-bits, AES 192-bits, AES 256-bits, AES XTS 256-bits, ECC 192-bits, ECC 256-bits Default: RSA 2048-bits	Select the key size.
Name of generated initialization function	sf_crypto_key_installation_i nit0	Select the name of the generated initialization function.
Auto Initialization	Enable, Disable Default: Enable	Select if sf_crypto_key_installation will be initialized during startup.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_crypto_signature 上の暗号化 Signature フレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Select if code for parameter checking is to be included in the build.
Name	g_sf_crypto_signature0	Module name.
Key type	RSA Plain text, RSA CRT Plain text, RSA Wrapped Default: RSA Plain text	Select the key type. Byte Pool size in sf_crypto must be > = 1450 bytes.
Key size	RSA 1024-bits, RSA 2048-bits Default: RSA 2048-bits	Select the key size.
Name of generated initialization function	sf_crypto_signature_init0	Select the name of the generated initialization function.
Auto Initialization	Enable, Disable Default: Enable	Select if sf_crypto_signature will be initialized during startup.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

4.1.18.6 アプリケーションでの暗号化フレームワークモジュールの使用

アプリケーションで SF CRYPTO フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

- SF CRYPTO モジュールを使用するには：
  - 構成パラメータ（上記の表で指定されています）をアプリケーションのニーズに合わせて確実に設定します。
  - sf_crypto_api_t::open API を使用して、SCE 共通ドライバーで SF_CRYPTO と SCE HAL モジュール（R_SCE）を初期化します。これは、[Auto Initialize] 設定がデフォルトのままの場合、Synergy コンフィギュレータによって実行されます。
  - リトルエンディアンモードがデフォルトで設定されます。これはロックされており、構成できません。

- open 関数は、モジュールが閉じられるまで再度呼び出すことはできません。
- 暗号化フレームワークサービスと HW SCE を終了するには sf_crypto_api_t::close API を使用します。

アプリケーションでの SF CRYPTO HASH フレームワークモジュールの使用

アプリケーションで SF CRYPTO HASH フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

API を次の順番でコールします。sf_crypto_hash_api_t::open -> sf_crypto_hash_api_t::hashInit -> sf_crypto_hash_api_t::hashUpdate -> sf_crypto_hash_api_t::hashFinal -> sf_crypto_hash_api_t::close

詳細：

- Synergy コンフィギュレータで SF_CRYPTO HASH フレームワークモジュールを追加します。このインスタンスの HASH アルゴリズムは、デフォルトで SHA 256 に設定されます。
- リトルエンディアンモードがデフォルトで設定されます。これはロックされており、構成できません。（モジュールの動作に関する注意事項のエンディアンとデータフォーマットに関する注意を参照してください。）
- SF CRYPTO モジュールを最初に開く必要があります（これは、デフォルトで設定されている [Auto start] オプションがそのままにされていれば、Synergy コンフィギュレータによって実行されます）。
- 選択した HASH アルゴリズム（SF_CRYPTO_HASH フレームワークのコンフィグで指定）をモジュールの open API の構成パラメータとして設定します。
- sf_crypto_hash_api_t::open API を使用して、SCE HASH ドライバーで SF CRYPTO HASH フレームワークモジュールおよび SCE HASH HAL モジュール（R_SCE_HASH）を初期化します（これは、デフォルトで設定されている [Auto init] オプションがそのままにされていれば、Synergy コンフィギュレータによって実行されます）。
- sf_crypto_hash_api_t::open API 関数は、モジュールが閉じられるまで再度呼び出すことはできません。
- sf_crypto_hash_api_t::hashInit API を使用して、選択した HASH アルゴリズムの HASH 操作を初期化します。
  - hashInit を open API、hashUpdate API、hashFinal API の後にコールして、構成した HASH アルゴリズムで新しい操作を初期化することができます。
- sf_crypto_hash_api_t::hashUpdate API を使用して、入力データをハッシュします。すべてのデータが完全に使い尽くされるまで、複数回コールすることができます。
- sf_crypto_hash_api_t::hashFinal API を使用して、sf_crypto_hash_api_t::hashUpdate API. でハッシュされたすべてのデータのメッセージダイジェストを取得します。
  - sf_crypto_hash_api_t::hashFinal API がコールされたら、sf_crypto_hash_api_t::hashInit をコールして、同じ HASH アルゴリズムを使用して次のデータセットの動作を初期化するか、sf_crypto_hash_api_t::close API をコールしてモジュールを閉じることができます。その後、別の HASH アルゴリズムについて再度開くことができます。
- 暗号化 HASH フレームワークサービスを終了するには sf_crypto_hash_api_t::close API を使用します。

注 : 特定のスレッドで、複数のインスタンスを作成せずに SF CRYPTO HASH モジュールの 1 つのインスタンスを別の HASH アルゴリズムに再利用することができます。

アプリケーションでの SF CRYPTO TRNG フレームワークモジュールの使用

アプリケーションで SF CRYPTO TRNG フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

- sf_crypto_trng_api_t::open API を使用して、SF_CRYPTO_TRNG および SCE TRNG HAL モジュール（R_SCE_TRNG）. を初期化します。これは、自動初期化設定がデフォルトのままの場合、ISDE によって実行されます。

- リトルエンディアンモードがデフォルトで設定されます。これはロックされており、構成できません。(モジュールの動作に関する注意事項のエンディアンとデータフォーマットに関する注意を参照してください。)
- open 関数は、モジュールが閉じられるまで再度呼び出すことはできません。
- sf_crypto_trng_api_t::randomNumberGenerate API を使用して、乱数を生成します。要求できる乱数の最小の長さは 1 バイトです。
- TRNG サービスが不要になった場合は、sf_crypto_trng_api_t::close API を使用して暗号化フレームワークサービスと SCE を終了します。

注 : 稀に sf_crypto_trng_api_t::randomNumberGenerate API がエラーを返した場合は、すべての暗号化フレームワークモジュールを閉じ、SF CRYPTO モジュールを再度開く必要があります。

注 :TRNG HAL モジュールのインスタンスは 1 つだけなので、暗号化 TRNG フレームワークの各インスタンスに設定される構成パラメータは、それまでに構成されたインスタンスがあると上書きされます。最後に構成された値がすべてのインスタンスに適用されます。

アプリケーションでの SF CRYPTO Cipher フレームワークモジュールの使用

アプリケーションで SF CRYPTO Cipher フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

API を次の順番でコールします。sf_crypto_cipher_api_t::open -> sf_crypto_cipher_api_t::cipherInit ->

sf_crypto_cipher_api_t::cipherUpdate -> sf_crypto_cipher_api_t::cipherFinal -> sf_crypto_cipher_api_t::close.

詳細 :

- モジュールの動作に関する注意事項のエンディアンとデータフォーマットに関する注意を参照してください。
- 最初に暗号化フレームワークモジュールを開く必要があります。これは、デフォルトで設定されている自動初期化オプションがそのままにされていれば、ISDE によって実行されます。アプリケーションでの暗号化フレームワークモジュールの使用に関するセクションを参照してください。
- アプリケーションのニーズに応じて、モジュールの sf_crypto_cipher_api_t::open API に対して構成パラメータを設定します。
- sf_crypto_cipher_api_t::open API を使用して、SCE ドライバーで暗号化 Cipher フレームワークモジュール (これは、自動開始オプションがデフォルト設定の場合、ISDE によって実行されます) と SCE HAL モジュールを開きます。
- open 関数は、モジュールが閉じられるまで再度呼び出すことはできません。
- sf_crypto_cipher_api_t::cipherInit API を使用して、適切な動作モード、キー、パディング方式オプション、アルゴリズム固有のパラメータを割り当てることでコンテキストを初期化します。
- 暗号化 / 復号化操作用のデータをすべて一度に使用できないが、チャンクとして使用できる場合は、sf_crypto_cipher_api_t::cipherUpdate API を使用します。sf_crypto_cipher_api_t::cipherUpdate API は複数回コールすることができます。

*注 :RSA 操作では、*sf_crypto_cipher_api_t::cipherUpdate API *から出力されるデータはありません。sf_crypto_cipher_api_t::cipherFinal API がコールされるまでメッセージは累積され続けます。*

- 暗号化または復号化操作を完了するには、sf_crypto_cipher_api_t::cipherFinal API をコールします。この API は、受信データの最後のチャンクを受け入れます。すべてのデータが sf_crypto_cipher_api_t::cipherUpdate API を通じて渡された場合、最後のメッセージチャンクの長さを 0 に設定することができます。
- 暗号化 / 復号化するデータをすべて一度に使用できる場合は、sf_crypto_cipher_api_t::cipherUpdate API を使用せずに、sf_crypto_cipher_api_t::cipherFinal API を直接コールできます。

- 暗号化 Cipher フレームワークモジュールサービスを終了するには sf_crypto_cipher_api_t::close API を使用します。

AES GCM 操作固有：

- AES GCM 暗号化 / 復号化操作では、AAD（Additional Authenticated Data）はオプションです。これを使用する場合は、sf_crypto_cipher_api_t::cipherInit API をコールしてから、sf_crypto_cipher_api_t::cipherAadUpdate API または sf_crypto_cipher_api_t::cipherFinal API をコールするまでの間に、sf_crypto_cipher_api_t::cipherUpdate API を通じて暗号化操作に提供する必要があります。
- AES GCM 暗号化操作では、sf_crypto_cipher_api_t::cipherFinal API が実行されるとタグ（認証タグ）が生成されます。コール側は cipherInit API を通じて、タグを保持するバッファを提供する必要があります。
- AES GCM 復号化操作では、cipherUpdate/cipherFinal API を通じて何らかの暗号文データを指定する前に、タグを指定する必要があります。したがって、コール側は cipherInit API を通じてタグを指定する必要があります。
- AES GCM 復号化操作は、cipherUpdate/cipherFinal API を通じてプレーンテキストデータを出力することがありますが、復号化操作がエラーコードを返した場合、このデータは使用できません。これは、データの使用前にタグが確実に検証されるようにするためです。

注 : 特定のスレッドで暗号化 Cipher フレームワークモジュールの 1 つのインスタンスを再利用して、動作モードや関連するパラメータを適切に設定および再設定することで、暗号化操作または復号化操作を交互に実行できます。暗号化 Cipher フレームワークモジュールの 2 つのインスタンスを同時に使用することもできます。

アプリケーションでの SF CRYPTO KEY フレームワークモジュールの使用

アプリケーションで SF CRYPTO KEY フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

API を次の順番でコールします。sf_crypto_key_api_t::open -> sf_crypto_key_api_t::keyGenerate -> sf_crypto_key_api_t::close.

詳細：

- モジュールの動作に関する注意事項のエンディアンとデータフォーマットに関する注意を参照してください。
- 最初に暗号化 Key フレームワークを開く必要があります。これは、自動初期化オプションがデフォルト設定の場合、ISDE によって実行されます。「アプリケーションでの暗号化フレームワークモジュールの使用」セクションを参照してください。
- アプリケーションのニーズに応じて、モジュールの sf_crypto_key_api_t::open API で必要とされる構成パラメータを設定します。
- sf_crypto_key_api_t::open API を使用して、SCE ドライバーで暗号化 Key フレームワークモジュール（これは、自動開始オプションがデフォルト設定の場合、ISDE によって実行されます）と SCE HAL モジュールを初期化します。
- open 関数は、モジュールが閉じられるまで再度呼び出すことはできません。
- sf_crypto_key_api_t::keyGenerate API を使用して、open コール時に構成パラメータによって設定されたタイプとサイズの暗号キーを生成します。
- 暗号化 Key フレームワークモジュールサービスを終了するには sf_crypto_key_api_t::close API を使用します。

注 : 特定のスレッドで、複数のインスタンスを作成せずに暗号化 Key フレームワークモジュールの 1 つのインスタンスを別のキータイプの生成に再利用することができます。

アプリケーションでの SF Crypto Key Installation フレームワークモジュールの使用

アプリケーションで SF Crypto Key Installation フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

API を次の順番でコールします。sf_crypto_key_installation_api_t::open -> sf_crypto_key_installation_api_t::keyInstall2 -> sf_crypto_key_installation_api_t::close.

詳細：

- モジュールの動作に関する注意事項のエンディアンとデータフォーマットに関する注意を参照してください。
- 最初に暗号化フレームワークモジュールを開く必要があります。これは、自動初期化オプションがデフォルト設定の場合、ISDE によって実行されます。「アプリケーションでの暗号化フレームワークモジュールの使用」セクションを参照してください。
- アプリケーションのニーズに応じて、モジュールの sf_crypto_key_installation_api_t::open API に対して構成パラメータを設定します。
- sf_crypto_key_installation_api_t::open API を使用して、SCE ドライバーで暗号化 Key Installation フレームワークモジュール（これは、自動開始オプションがデフォルト設定の場合、ISDE によって実行されます）と SCE HAL モジュールを開きます。
- open 関数は、モジュールが閉じられるまで再度呼び出すことはできません。
- sf_crypto_key_installation_api_t::keyInstall2 API を使用して、ユーザーのキーをインストールします。（sf_crypto_key_installation_api_t::keyInstall API は非推奨です。）
- 暗号化 Key Installation フレームワークモジュールサービスを終了するには sf_crypto_key_installation_api_t::close API を使用します。

アプリケーションでの SF CRYPTO Signature フレームワークモジュールの使用

アプリケーションで SF CRYPTO Signature フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

API を次の順番でコールします。sf_crypto_signature_api_t::open -> sf_crypto_signature_api_t::contextInit -> sf_crypto_signature_api_t::signUpdate -> sf_crypto_signature_api_t::signFinal -> sf_crypto_signature_api_t::close

API を次の順番でコールします。sf_crypto_signature_api_t::open -> sf_crypto_signature_api_t::contextInit -> sf_crypto_signature_api_t::signUpdate -> sf_crypto_signature_api_t::signFinal -> sf_crypto_signature_api_t::close

詳細：

- モジュールの動作に関する注意事項のエンディアンとデータフォーマットに関する注意を参照してください。
- 最初に暗号化フレームワークモジュールを開く必要があります。これは、自動初期化オプションがデフォルト設定の場合、ISDE によって実行されます。「アプリケーションでの暗号化フレームワークモジュールの使用」セクションを参照してください。
- アプリケーションのニーズに応じて、モジュールの sf_crypto_signature_api_t::open API で必要とされる構成パラメータを設定します。
- sf_crypto_signature_api_t::open API を使用して、SCE ドライバーで暗号化 Signature フレームワークモジュール（これは、自動開始オプションがデフォルト設定の場合、ISDE によって実行されます）と SCE HAL モジュールを開きます。
- open 関数は、モジュールが閉じられるまで再度呼び出すことはできません。
- sf_crypto_signature_api_t::contextInit API を使用して、適切な動作モード、キー（検証動作モードでは公開キー、署名動作モードでは秘密キー）、メッセージ構造オプションを割り当てることでコンテキストを初期化します。

- データが署名動作モード用の小さいチャンクで受信される場合は、その後の署名動作のために sf_crypto_signature_api_t::signFinal API を使用してデータを累積するか、検証動作モードでは、その後の署名検証のために sf_crypto_signature_api_t::verifyFinal API を使用してデータを累積します。
- 署名操作または検証操作を完了するためには、それぞれ sf_crypto_signature_api_t::signFinal API または sf_crypto_signature_api_t::verifyFinal API をコールします。これらの API は、受信データの最後のチャンクを受け入れます。すべてのデータが渡された場合、最後のメッセージチャンクに対する update API パラメータの 1 つを NULL に設定することができます。
- 署名または検証するデータをすべて一度に使用できる場合は、いずれの update API も使用せずに、sf_crypto_signature_api_t::signFinal API または sf_crypto_signature_api_t::verifyFinal API を直接コールできます。
- 暗号化 Signature フレームワークモジュールサービスを終了するには sf_crypto_signature_api_t::close API を使用します。

注 : 特定のスレッドで暗号化 Signature フレームワークモジュールの 1 つのインスタンスを再利用して、動作モードや関連するパラメータを適切に設定および再設定することで、署名操作または検証操作を交互に実行できます。署名操作または検証操作を同時に実行するためには、SF Crypto Signature フレームワークモジュールの 2 つのインスタンスを使用する必要があります。

4.1.19 静電容量式タッチフレームワーク

静電容量式タッチフレームワークモジュールは、静電容量式タッチフレームワークモジュールアプリケーション用のハイレベルの ThreadX 対応 API を提供します。静電容量式タッチフレームワークモジュールは、基本的には CTSU HAL ドライバーのフレームワークバージョンとして機能し、追加機能としてスキャンを自律的に実行します。ボタンやスライダなどの他のフレームワークと共に使用できます。また、CTSU ペリフェラルの各チャネルのステータスに関してバイナリデータのみを要求するアプリケーションでも使用できます。

4.1.19.1 静電容量式タッチフレームワークモジュールの特長

CTSU フレームワークインタフェースは、静電容量式タッチパネルのハードウェアスキャンを実行し、構成で指定された周期でパネルを更新するプライベートスレッドを作成します。以下の主要機能を提供します。

- 複数のアプリケーションレイヤがコールバックを登録できます。これにより、スライダやボタンなどの複数のウィジェットがこのレイヤを使用できます。
- 周期的な CTSU スキャンを自律的に実行し、ユーザーが指定したレートでプロセスを更新します。

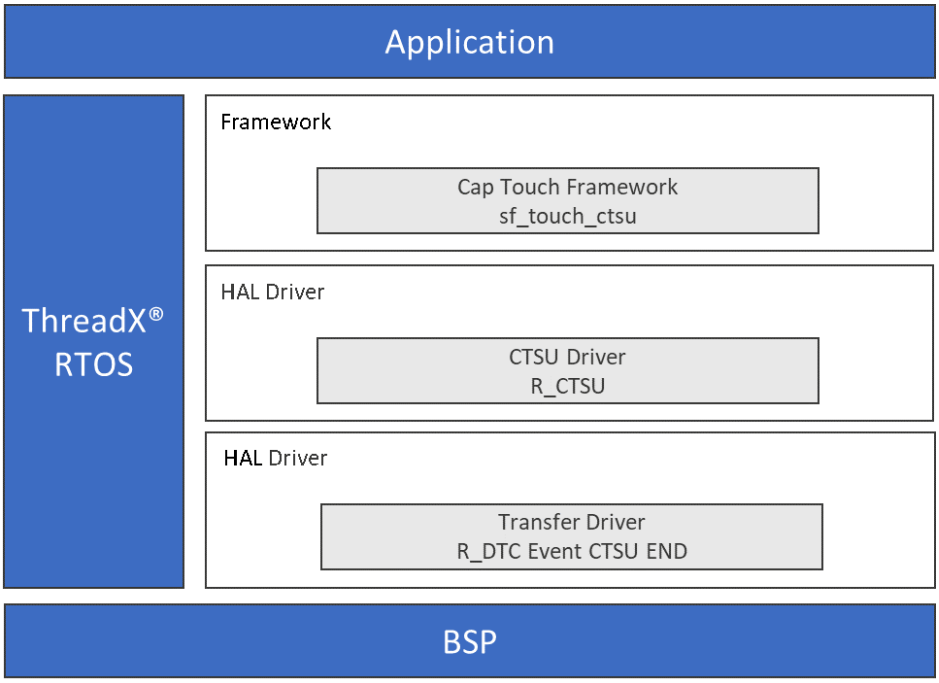


図 76:静電容量式タッチフレームワークモジュールのブロック図

4.1.19.2 静電容量式タッチフレームワークモジュールの API の概要

静電容量式タッチフレームワークは、オープン、リード、クローズの API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

静電容量式タッチフレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_touch_ctsu0.p_api->open(g_sf_touch_ctsu0.p_ctrl, g_sf_touch_ctsu0.p_cfg); Initialize the Touch CTSU Button Framework; configures the lower level hardware and registers callback functions for all the buttons.

Function Name	Example API Call and Description
read	g_sf_touch_ctsu0.p_api->read(g_sf_touch_ctsu0.p_ctrl, p_dest, read_options, channels, count); Reads data from the touch CTSU framework.
close	g_sf_touch_ctsu0.p_api->close(g_sf_touch_ctsu0.p_ctrl); Close the Touch CTSU Framework; Closes the CTSU framework and lower layers if no other modules are using it.
versionGet	g_sf_touch_ctsu0.p_api->versionGet(&p_version); Get version and store it in provided pointer p_version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successful.
SSP_ERR_ASSERTION	Assertion error.
SSP_ERR_IN_USE	The framework has already been initialized.
SSP_ERR_NOT_OPEN	Device not open.
SSP_ERR_INTERNAL	Internal error.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.19.3 静電容量式タッチフレームワークモジュールの動作の概要

CTSU フレームワークインタフェースは、構成された CTSU チャネルのハードウェアスキャンと更新を周期的に実行する内部スレッドを作成します。このフレームワーク モジュールは、HAL レイヤ CTSU ドライバーを使用してスキャン結果を読み取ります。スキャンが完了すると、アプリケーション レイヤーによって登録されたコールバックが呼び出されます。複数の上位レイヤーがこのフレームワークを使用している場合（ボタン、スライダ、ホイールなど）、このレイヤーは初期化を行った順にこれら各レイヤーのコールバックを呼び出します。サポートされるコールバック数は現在 3 に制限されています。

内部スレッド（update_hz）の周波数は、このフレームワークに対して実行される後続の各 sf_touch_ctsu_api_t::open API コールで設定または変更されます。スキャンレートを設定する場合は、ハードウェアスキャンの完了と処理に必要な時間によってスキャンレートが制限されることに注意してください。スキャンの完了にかかる時間は、構成されているチャネルの数と、基礎となる CTSU HAL レイヤーが構成されているモードによって異なります。詳細については、Renesas Synergy ウェブサイトにある静電容量式タッチのアプリケーションノートを参照してください。

静電容量式タッチフレームワークモジュールの動作に関する重要な注意事項と制限事項

静電容量式タッチフレームワークモジュールの動作に関する重要な注意事項

静電容量式タッチフレームワークの構成情報は、Capacitive Touch Workbench for Renesas Synergy ツールによって生成されます。このツールの使用と入手の詳細については、Renesas Synergy ウェブサイトにあるドキュメントを参照してください。

静電容量式タッチフレームワークモジュールの動作に関する制限事項

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.19.4 アプリケーションへの静電容量式タッチフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して静電容量式タッチフレームワークモジュールをアプリケーションに組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

静電容量式タッチフレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（静電容量式タッチフレームワークのデフォルト名は g_sf_touch_ctsu0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

静電容量式タッチフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_touch_ctsu0 Cap Touch Framework on sf_touch_ctsu	Threads	New Stack > Framework > Input > Cap Touch Framework on sf_touch_ctsu

次の図に示すように、sf_touch_ctsu の静電容量式タッチフレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

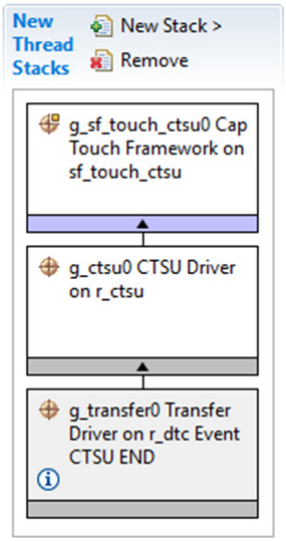


図 77:静電容量式タッチフレームワークモジュールスタック

4.1.19.5 静電容量式タッチフレームワークモジュールの構成

ユーザーは必要な動作に合わせて静電容量式タッチフレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_touch_ctsu 上の静電容量式タッチフレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_sf_touch_ctsu0	Module name.
Thread Priority	3	User determined based on priority of other threads in system. Recommend high priority.
Update Hz	50	Rate at which the CTSU hardware scans should occur. The maximum scan rate should be less than the tick rate set for ThreadX. The total time to complete a hardware scan is determined by the number of channels used. Each channel takes approximately 600 usec to complete a scan. Thus if 10 channels are used for 10 buttons in self-capacitance mode, the total hardware scan time is ~6 milliseconds or 166 Hz. Similarly, if 7channels are used for 5 buttons in mutual-capacitance mode (5x2 layout), the total hardware scan time is ~4.2 milliseconds or 250 Hz.The software processing time is additional and will vary depending on the core clock and the software filter depth used by the CTSU. Thus the maximum scan rate should be lesser than the time required to scan and process all the channels used in the configuration and lesser than the RTOS tick rate.

ISDE Property	Value	Description
Callback	NULL	Name of user callback that will be called when each scan is complete and also when new processed data is available. Can be NULL if not used.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

静電容量式タッチフレームワークモジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_ctsu 上の CTSU HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking Enable	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Offset Adjustment	Enabled, Disabled Default: Enabled	Used to enable or disable offset adjustment. Leave as enabled unless you are tuning the board. (Rate is determined by *Runtime rate of tuning of sensor values*.)

ISDE Property	Value	Description
Drift Compensation	Enabled, Disabled Default: Enabled	Used to enable or disable drift compensation. Leave as enabled unless you are tuning the board. Drift compensation allows the baseline values used to determine the presence or absence of a touch to change over time as the board or surface ages or with changes in temperature and environment. Drift compensation rate is determined by *Steady state drift compensation rate*, *Startup drift compensation rate*, and *Channel release compensation rate*.
Drift Compensation Method (valid only if Drift Compensation is enabled above)	Alternate Method 1	Drift Compensation algorithms provided. Only Alternate 1 is currently supported. Other options may be supported in future releases. Drift compensation method Alternate 1 allows for different rates of drift compensation for the following events: steady state, startup, and channel release.
Steady state drift compensation rate, drift compensation will be applied per n scans	500	Determines the rate of drift compensation (applied every n scans) when the channel is in steady state. (Dependent on the scan rate.)
Startup drift compensation rate (Should be less than the steady state drift compensation)	5	Determines the rate of drift compensation (applied every n scans) when the driver is performing its initialization. (Should be less than the *Steady state drift compensation rate*.)

ISDE Property	Value	Description
Channel release compensation rate (Should be less than the steady state drift compensation rate)	500	Determines the rate of drift compensation (applied every n scans) when a channel transitions from pressed to released. (Should be less than or equal to the *Steady state drift compensation rate*.)
Default filter depth (used in sensor count filter provided by driver)	1	Used in the software sensor count filter provided by driver. This default filter is a modification of the relatively common "leaky integrator" software filter. A depth of 4 equates to an approximate filter constant of 16. When the input is a constant value of 16 or greater, the filter output will reach 68-69% of the constant input value after 16 cycles/iterations of the filter.
Runtime rate of tuning of sensor values (if drift compensation is used this value is overrriden to be twice the rate of the steady state drift compensation)	800	Rate of offset adjustment. (If drift compensation is used this value is set to twice the rate of the steady state drift compensation.)
Perform auto-tune and drift compensation only when all channels are untouched	True, False Default: True	Perform auto-tune and drift compensation only when all channels are untouched.
Max. active channels	1	Specify the maximum number of channels that are to be used by the application. In Self Capacitance Mode, this is the number of channels used. In Mutual Capicitace Mode, this is the multiplication result of the Rx and Tx channels. For example: 4 Rx and 3 Tx channels = 12 Maximum Active Channels.

ISDE Property	Value	Description
Name	g_ctsu0	Module name.
CTSU configuration used	g_ctsu0_config	Name of the configuration structure generated by CTW.
Callback	NULL	The user callback function that will be invoked each time the scan is complete as well as when newly processed data is available. Can be set to NULL if not used.
Data Processing Option	Default Processing (Recommended), No Processing (Tuning only) Default: Default Processing	Can be used to specify if scans should start after the data from the previous one is completed, if auto-calibration should be run between scans etc. Use the Default Processing unless you are tuning.
Write Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Write interrupt priority selection.
Read Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Read interrupt priority selection.
End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	End interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event CTSU END 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Enabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Module name
Mode	Block	Mode selection
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	1	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	1	Number of blocks selection
Activation Source (Must enable IRQ)	Event CTSU END	Activation source selection
Auto Enable	False	Auto enable selection

ISDE Property	Value	Description
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC software event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

静電容量式タッチフレームワークモジュールのクロック構成

CTSU フレームワークは CTSU HAL ドライバーと同じ構成を使用します。CTSU クロック速度を、Capacitive Touch Workbench for Renesas Synergy（CTW）で選択したクロック速度と同じ値に設定します。

静電容量式タッチフレームワークモジュールのピン構成

CTSU フレームワークは CTSU HAL ドライバーと同じ構成を使用します。外部デバイスの要求に応じて、ピンをタッチセンサーピン TSxx として設定し、TSCAP 機能を有効化します。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では I2C ピンの選択例を示します。

注 : 一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号が決まり、それに従って必要な MCU ピンが決定されます。

静電容量式タッチセンシングユニット（CTSU）のピン選択

Resource	ISDE Tab	Pin selection Sequence
CTSU	Pins	Select Peripherals > Input: CTSU > CTSU0

CTSU のピン構成設定

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Enabled Default: Disabled	Select Enabled as the Operation Mode for CTSU

Pin Configuration Property	Value	Description
TS00 to TS11	None, Pn, Pm Default: None	TS Pins
TSCAP	None, Pn, Pm Default: None)	TSCAP Pin

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.1.19.6 アプリケーションでの静電容量式タッチフレームワークモジュールの使用

アプリケーションで静電容量式タッチフレームワークモジュールを使用する際の一般的な手順では、最初に次の操作を行います。

1) ISDE を通じてアプリケーションにフレームワークを追加します。
2) 内部スレッドがチャネルスキャンを開始するレートを構成します。
3) スキャンが完了するたびに呼び出す必要のあるアプリケーションコールバックを指定します。

これらの手順が完了したら、必要に応じてモジュールの API にアクセスできます。

1) sf_touch_ctsu_api_t::open API を使用して CTSU タッチフレームワークを初期化します。
2) コールバックによって新しいスキャンの可用性が通知されたら、sf_touch_ctsu_api_t::read API を使用してスキャンしたデータを読み取ります。
3) sf_touch_ctsu_api_t::close API を使用して CTSU タッチフレームワークを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

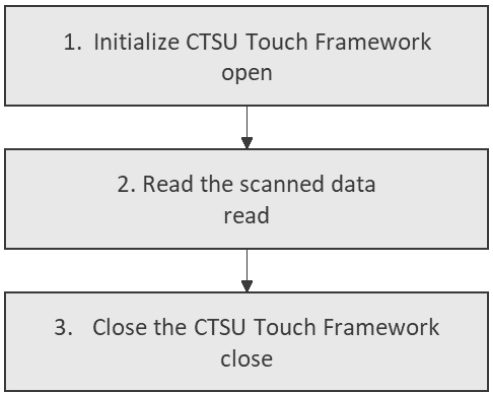


図 78:通常の静電容量式タッチフレームワークモジュールアプリケーションのフロー図

4.1.20 静電容量式タッチボタンフレームワーク

静電容量式タッチボタンフレームワークモジュールは、ThreadX RTOS を使用する静電容量式タッチボタンアプリケーション用のハイレベルの ThreadX 対応 API を提供します。静電容量式タッチボタンフレームワークモジュールは、Synergy MCU 上の CTSU 周辺機能を使用します。各ボタンに対して列挙された順に応答する、ユーザー定義のコールバックを作成できます。

4.1.20.1 静電容量式タッチボタンフレームワークモジュールの特長

静電容量式タッチボタンフレームワークモジュールは、システムに存在するすべてのボタンの CTSU データを解釈するために使用されます。主な特長をいくつか以下に示します。

- 構成データを生成する Capacitive Touch Workbench for Renesas Synergy（CTW）ツールと連携します。
- コールバック関数をイベントに提供します。
  - デバウンシングを実行します
  - Press、Release、LongTouch などの複数のタイプのイベントをサポートします
  - 各ボタンのコールバックをボタン構成テーブルに列挙された順に呼び出します。

静電容量式タッチボタンフレームワークには静電容量式タッチフレームワークが必要です。ほとんどの場合、必要な構成情報はモジュールに自動的に追加されます。ISDE 構成オプションについては、このドキュメントで詳しく説明します。

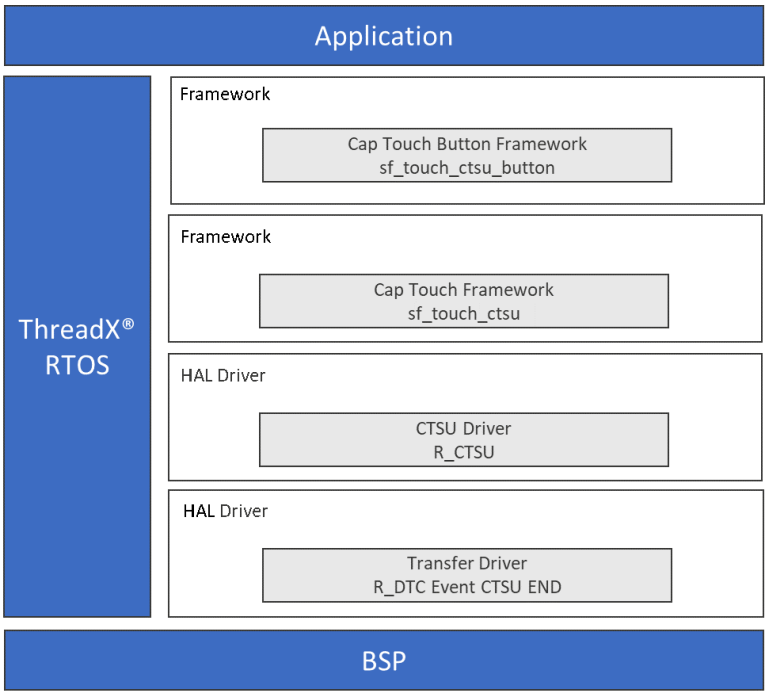


図 79:静電容量式タッチボタンフレームワークモジュールのブロック図

4.1.20.2 静電容量式タッチボタンフレームワークモジュール API の概要

静電容量式タッチボタンフレームワークは、オープン、有効化、無効化、クローズの API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

静電容量式タッチボタンフレームワークモジュール API の要約

Function Name	Example API Call and Description
open	g_sf_touch_button0.p_api->open(g_sf_touch_button0.p_ctrl, g_sf_touch_button0.p_cfg); Initialize the Touch CTSU Button Framework; configures the lower level hardware and registers callback functions for all the buttons.
enable	g_sf_touch_button0.p_api->enable(g_sf_touch_button0.p_ctrl, button_id); Enable callback notification for a configured button.
disable	g_sf_touch_button0.p_api->disable(g_sf_touch_button0.p_ctrl, button_id); Disable callback notification for a configured button.
close	g_sf_touch_button0.p_api->open(g_sf_touch_button0.p_ctrl); Close the Touch CTSU Button Framework; Closes the button framework and lower layers if no other modules are using it.
versionGet	g_sf_touch_button0.p_api->versionGet(&p_version); Get version and store it in provided pointer p_version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successful.
SSP_ERR_ASSERTION	Assertion error.
SSP_ERR_IN_USE	The framework has already been initialized by this particular widget.
SSP_ERR_NOT_OPEN	Device not open.
SSP_ERR_INTERNAL	Internal error.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.20.3 静電容量式タッチボタンフレームワークモジュールの動作の概要

静電容量式タッチボタンフレームワークは、システムに存在するすべてのボタンの CTSU データを解釈するために使用されます。CTSU フレームワークレイヤーも初期化し、さらにハードウェアレイヤーを初期化します。静電容量式タッチボタンフレームワークには、処理されたデータが使用可能になるたびに CTSU フレームワークレイヤを伴って呼び出されるコールバックがあります。静電容量式タッチボタンフレームワークはこの処理済みのデータ（バイナリ）を使用してデバウンシングを実行し、構成されたイベント（Press、Release、LongTouch など）のいずれが各ボタンに対して有効かを判断します。

フレームワークは、各ボタンのコールバックをボタン構成テーブルに列挙された順に呼び出します。ユーザーがスキャン プロセスを停止しない限り、引き続きタイマによってスキャンがトリガされ、データがユーザー バッファに書き込まれます。ユーザー バッファはフレームワークによって循環バッファとして扱われます。バッファの名前と長さは ISDE コンフィギュレータによって指定します。

静電容量式タッチボタンフレームワークモジュールの動作に関する重要な注意事項と制限事項

静電容量式タッチボタンフレームワークは、Workbench 6CTW ツールによって生成された構成データと組み合わせて使用するように設計されています。このツールを使用して構成データを生成する方法の詳細については、Renesas Synergy ウェブサイトにある Workbench 6 Capacitive Touch ユーザーズガイドでアプリの説明を参照してください。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.20.4 アプリケーションへの静電容量式タッチボタンフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して静電容量式タッチボタンフレームワークモジュールをアプリケーションに組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

静電容量式タッチボタンフレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(静電容量式タッチボタンフレームワークのデフォルト名は g_sf_touch_button_ctsu0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

静電容量式タッチボタンフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_touch_button0 Cap Touch Button Framework on sf_touch_ctsu_button	Threads	New Stack > Framework > Input > Cap Touch Button Framework on sf_touch_ctsu_button

次の図に示すように、sf_touch_ctsu_button の静電容量式タッチボタンフレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

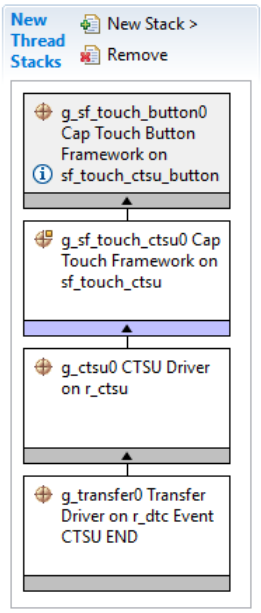


図 80:静電容量式タッチボタンフレームワークモジュールのスタック

4.1.20.5 静電容量式タッチボタンフレームワークモジュールの構成

必要な動作に合わせて静電容量式タッチボタンフレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_touch_ctsu_button 上の静電容量式タッチボタンフレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Controls whether to include code for API parameter checking

ISDE Property	Value	Description
Number of Buttons	1	Number of buttons configured in CTW
Short hold debounce multiplier	1	Specify the multiplier for a button short-hold event
Long hold debouce multiplier	5	Specify the multiplier for a button long-hold event
Stuck in debounce multiplier	10	Multiplier used in debounce function
Multi touch enable	Enabled, Disabled Default: Disabled	Enables or disables multi-touch detection
Enable stuck at condition detection	Enabled, Disabled Default: Disabled	Specify the multiplier for a button stuck-at event
Name	g_sf_touch_button0	Module name
Button Configuration Structure Name	touch_buttons	Name of button configuration structure generated by CTW
Callback	g_button_framework_user_callback	Name of callback function created by user application.
Name of generated initialization function	sf_touch_button_init0	Name of generated initalization function
Auto Initialization	Enabled, Disabled Default: Enabled	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

静電容量式タッチボタンフレームワークモジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

sf_touch_ctsu 上の静電容量式タッチフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_sf_touch_ctsu0	Module name.
Thread Priority	3	User determined based on priority of other threads in system. Recommend high priority.

ISDE Property	Value	Description
Update Hz	50	Rate at which the CTSU hardware scans should occur. The maximum scan rate should be less than the tick rate set for ThreadX. The total time to complete a hardware scan is determined by the number of channels used. Each channel takes approximately 600 usec to complete a scan. Thus if 10 channels are used for 10 buttons in self-capacitance mode, the total hardware scan time is ~6 milliseconds or 166 Hz. Similarly, if 7channels are used for 5 buttons in mutual-capacitance mode (5x2 layout), the total hardware scan time is ~4.2 milliseconds or 250 Hz.The software processing time is additional and will vary depending on the core clock and the software filter depth used by the CTSU. Thus the maximum scan rate should be lesser than the time required to scan and process all the channels used in the configuration and lesser than the RTOS tick rate.
Callback	NULL	Name of user callback that will be called when each scan is complete and also when new processed data is available. Can be NULL if not used.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_ctsu 上の CTSU HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking Enable	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Offset Adjustment	Enabled, Disabled Default: Enabled	Used to enable or disable offset adjustment. Leave as enabled unless you are tuning the board. (Rate is determined by *Runtime rate of tuning of sensor values*.)
Drift Compensation	Enabled, Disabled Default: Enabled	Used to enable or disable drift compensation. Leave as enabled unless you are tuning the board. Drift compensation allows the baseline values used to determine the presence or absence of a touch to change over time as the board or surface ages or with changes in temperature and environment. Drift compensation rate is determined by *Steady state drift compensation rate*, *Startup drift compensation rate*, and *Channel release compensation rate*.
Drift Compensation Method (valid only if Drift Compensation is enabled above)	Alternate Method 1	Drift Compensation algorithms provided. Only Alternate 1 is currently supported. Other options may be supported in future releases. Drift compensation method Alternate 1 allows for different rates of drift compensation for the following events: steady state, startup, and channel release.

ISDE Property	Value	Description
Steady state drift compensation rate, drift compensation will be applied per n scans	500	Determines the rate of drift compensation (applied every n scans) when the channel is in steady state. (Dependent on the scan rate.)
Startup drift compensation rate (Should be less than the steady state drift compensation)	5	Determines the rate of drift compensation (applied every n scans) when the driver is performing its initialization. (Should be less than the *Steady state drift compensation rate*.)
Channel release compensation rate (Should be less than the steady state drift compensation rate)	500	Determines the rate of drift compensation (applied every n scans) when a channel transitions from pressed to released. (Should be less than or equal to the *Steady state drift compensation rate*.)
Default filter depth (used in sensor count filter provided by driver)	1	Used in the software sensor count filter provided by driver. This default filter is a modification of the relatively common "leaky integrator" software filter. A depth of 4 equates to an approximate filter constant of 16. When the input is a constant value of 16 or greater, the filter output will reach 68-69% of the constant input value after 16 cycles/iterations of the filter.
Runtime rate of tuning of sensor values (if drift compensation is used this value is overrriden to be twice the rate of the steady state drift compensation)	800	Rate of offset adjustment. (If drift compensation is used this value is set to twice the rate of the steady state drift compensation.)
Perform auto-tune and drift compensation only when all channels are untouched	True, False Default: True	Perform auto-tune and drift compensation only when all channels are untouched.

ISDE Property	Value	Description
Max. active channels	1	Specify the maximum number of channels that are to be used by the application. In Self Capacitance Mode, this is the number of channels used. In Mutual Capicitace Mode, this is the multiplication result of the Rx and Tx channels. For example: 4 Rx and 3 Tx channels = 12 Maximum Active Channels.
Name	g_ctsu0	Module name.
CTSU configuration used	g_ctsu0_config	Name of the configuration structure generated by CTW.
Callback	NULL	The user callback function that will be invoked each time the scan is complete as well as when newly processed data is available. Can be set to NULL if not used.
Data Processing Option	Default Processing (Recommended), No Processing (Tuning only) Default: Default Processing	Can be used to specify if scans should start after the data from the previous one is completed, if auto-calibration should be run between scans etc. Use the Default Processing unless you are tuning.
Write Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Write interrupt priority selection.
Read Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Read interrupt priority selection.

ISDE Property	Value	Description
End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	End interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event CTSU END 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Enabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Module name
Mode	Block	Mode selection
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection

ISDE Property	Value	Description
Source Pointer	NULL	Source pointer selection
Number of Transfers	1	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	1	Number of blocks selection
Activation Source (Must enable IRQ)	Event CTSU END	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC software event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

静電容量式タッチボタンフレームワークモジュールのクロック構成

CTSU ボタンフレームワークは CTSU HAL ドライバーと同じ構成を使用します。CTSU クロック速度を、Capacitive Touch Workbench for Renesas Synergy（CTW）で選択したクロック速度と同じ値に設定します。

静電容量式タッチボタンフレームワークモジュールのピン構成

CTSU ボタンフレームワークは CTSU HAL ドライバーと同じ構成を使用します。外部デバイスの要求に応じて、ピンをタッチセンサーピン TSxx として設定し、TSCAP 機能を有効化します。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では I2C ピンの選択例を示します。

注 : 一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号が決まり、それに従って必要な MCU ピンが決定されます。

静電容量式タッチセンシングユニット（CTSU）のピン選択

Resource	ISDE Tab	Pin selection Sequence
CTSU	Pins	Select Peripherals > Input: CTSU > CTSU0

CTSU のピン構成設定

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Enabled Default: Disabled	Select Enabled as the Operation Mode for CTSU
TS00 to TS11	None, Pn, Pm Default: None	TS Pins
TSCAP	None, Pn, Pm Default: None)	TSCAP Pin

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.1.20.6 アプリケーションでの静電容量式タッチボタンフレームワークモジュールの使用

静電容量式タッチボタンフレームワークを使用する前に、ISDE のプロジェクトにフレームワークを追加し、CTW の構成データがプロジェクトに追加されていることを確認して、ボタンの数、CTW によって生成された構成構造体の名前、使用するコールバックなどのフレームワークのプロパティを更新します。これを実行した後、アプリケーションで静電容量式タッチボタンフレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_touch_ctsu_button_api_t::open API を使用して静電容量式タッチボタンフレームワークモジュールを初期化します。
2) コールバック関数で、タッチボタンイベントを検出し、処理します。
3) sf_touch_ctsu_button_api_t::disable API を使用してボタンのコールバックを無効化します（オプション）。
4) sf_touch_ctsu_button_api_t::enable API を使用してボタンのコールバックを有効化します（オプション）。
5) sf_touch_ctsu_button_api_t::close API を使用して静電容量式タッチボタンフレームワークモジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

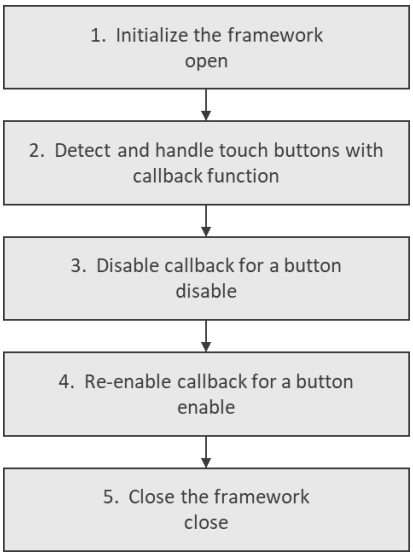


図 81:通常の静電容量式タッチボタンフレームワークモジュールアプリケーションのフロー図

4.1.21 静電容量式タッチスライダフレームワーク

静電容量式タッチスライダフレームワークモジュールは、静電容量式タッチスライダおよびホイールアプリケーション用のハイレベルの ThreadX 対応 API を提供し、Synergy MCU 上の CTSU を使用します。このフレームワークは、Capacitive Touch Workbench for Renesas Synergy（CTW）ツールによって生成された構成データとともに使用するように設計されています。使用するスライダ、ホイール、およびチャネルはツールで構成します。ユーザー定義のコールバックを作成しておくと、そのコールバックがスキャン後にハードウェアから使用可能になったときにデータを処理できます。

4.1.21.1 静電容量式タッチスライダ / ホイールフレームワークモジュールの特長

静電容量式タッチスライダフレームワークモジュールは、システムによって初期化されるすべてのスライダ構成の CTSU データを解釈するために使用されます。主な特長を以下に示します。

- スライダとホイールをサポート
- スライダとホイールの複数インスタンスをサポート
- コールバックを使用してタッチ処理を簡略化
  - CTW から生成された構成データを使用
  - 状態が変化したときにコールバックを生成
  - コールバックは各スライダに関連付けられ、イベントと位置を含む
  - コールバックは構成テーブルに列挙された順に呼び出される
- マルチタッチ検出をサポート（ビルド時にオプションで無効化できる）

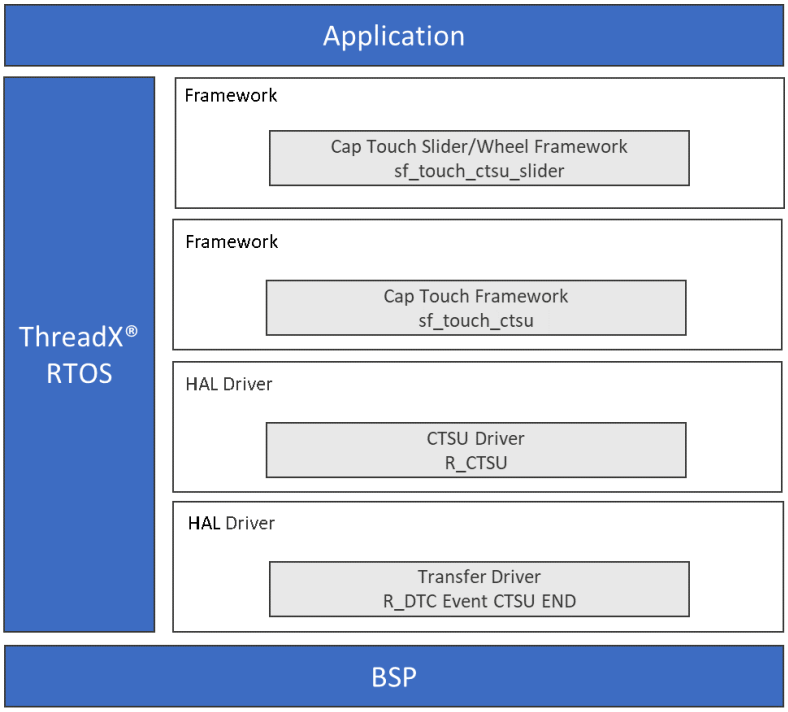


図 82:静電容量式タッチスライダ / ホイールフレームワークモジュールのブロック図

4.1.21.2 静電容量式タッチスライダ / ホイールフレームワークモジュールの API の概要

静電容量式タッチスライダ / ホイールフレームワークは、オープン、有効化、無効化、クローズの API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

静電容量式タッチスライダ / ホイールフレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_touch_slider0.p_api->open(g_sf_touch_slider0.p_ctrl, g_sf_touch_slider0.p_cfg); Initialize the Touch CTSU Button Framework, configure the lower level hardware and registers callback functions for all the buttons.

Function Name	Example API Call and Description
enable	g_sf_touch_slider0.p_api->enable(g_sf_touch_slider0.p_ctrl, slider_id); Enable callback notification for a configured button.
disable	g_sf_touch_slider0.p_api->disable(g_sf_touch_slider0.p_ctrl, slider_id); Disable callback notification for a configured button.
close	g_sf_touch_slider0.p_api->close(g_sf_touch_slider0.p_ctrl); Close the Touch CTSU Button Framework, close the button framework and lower layers if no other modules are using it.
versionGet	g_sf_touch_slider0.p_api->versionGet(&p_version); Get version and store it in provided pointer p_version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successful.
SSP_ERR_ASSERTION	Assertion error.
SSP_ERR_IN_USE	Framework already in use.
SSP_ERR_NOT_OPEN	Device not open.
SSP_ERR_INTERNAL	Internal error.

Name	Description
SSP_ERR_UNSUPPORTED	Device unsupported.
SSP_ERR_INVALID_ARGUMENT	Invalid argument.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.21.3 静電容量式タッチスライダ / ホイールフレームワークモジュールの動作の概要

静電容量式タッチスライダフレームワークモジュールは、システムによって初期化されるすべてのスライダ構成の CTSU データを解釈するために使用されます。また、CTSU フレームワーク レイヤーの初期化も行います。静電容量式タッチスライダフレームワークには、処理されたデータが使用可能になるたびに CTSU フレームワークレイヤを伴ってコールされるコールバックがあります。スライダー フレームワークは、このデータ (未加工値) を使用して、タッチまたは解放が発生したかどうか、また発生した場合はどこで発生したかを判断します。状態に変化があった場合、フレームワークはイベントと位置とともにスライダー構成テーブルに列挙された順に各スライダーのコールバックを呼び出します。スライダフレームワークは、タッチイベントと解放イベント間の更新レート (sf_touch_ctsu_cfg_t::update_hz) でコールバックを実行します。アプリケーション コードは、これらのコールバックを使用してスライダー上の位置を追跡する必要があります。スライダフレームワークは、タッチイベントと解放イベント間の更新レート（update_hz）でコールバックを実行します。この機能は、ビルド時にオプションで無効にできます。

静電容量式タッチスライダ / ホイールフレームワークモジュールの動作に関する重要な注意事項と制限事項

静電容量式タッチスライダ / ホイールフレームワークモジュールの動作に関する重要な注意事項

静電容量式タッチスライダ / ホイールフレームワークは、Workbench 6CTW ツールによって生成された構成データと組み合わせて使用するように設計されています。このツールを使用して構成データを生成する方法の詳細については、Renesas Synergy ウェブサイトにある Workbench 6 Capacitive Touch ユーザーズガイドでアプリの説明を参照してください。

静電容量式タッチスライダ / ホイールフレームワークモジュールの動作に関する制限事項

- CTSU スライダ / ホイールフレームワークは、セルフキャパシタンス動作モードで配置された静電容量式タッチスライダおよびホイールのみをサポートします。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.21.4 アプリケーションへの静電容量式タッチスライダ / ホイールフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して静電容量式タッチスライダ / ホイールフレームワークモジュールをアプリケーションに組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

静電容量式タッチスライダ / ホイールフレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(静電容量式タッチスライダ / ホイールフレームワークのデフォルト名は g_sf_touch_slider_ctsu0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

静電容量式タッチスライダ / ホイールフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_touch_slider0 Cap Touch Slider/Wheel Framework on sf_touch_ctsu_slider	Threads	New Stack> Framework> Input> Cap Touch Slider Framework on sf_touch_ctsu_slider

次の図に示すように、sf_touch_ctsu_slider 上の静電容量式タッチスライダ / ホイールフレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

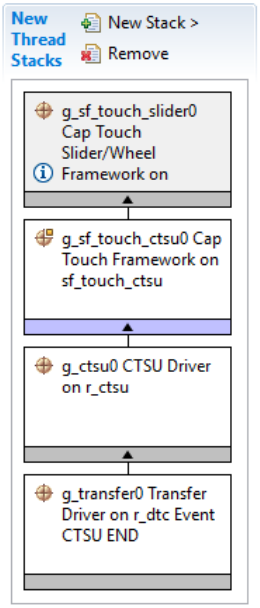


図 83:静電容量式タッチスライダ / ホイールフレームワークモジュールスタック

4.1.21.5 静電容量式タッチスライダ / ホイールフレームワークモジュールの構成

ユーザーは必要な動作に合わせて静電容量式タッチスライダ / ホイールフレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプロー

チにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_touch_ctsu_slider での静電容量式タッチスライダ / ホイールフレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Controls whether to include code for API parameter checking
Number of Sliders/Wheels	1	Number of sliders configured in CTW
Multi touch enable	Enabled, Disabled Default: Enabled	Enables or disables multi-touch detection
Name	g_sf_touch_slider0	Framework name
Slider/Wheel Configuration Structure Name (generated by Workbench)	all_sliders	Name of slider configuration structure generated by CTW
Callback	g_slider_framework_user_callback	Name of callback function created by user application
Name of generated initialization function	sf_touch_slider_init0	Name of generated initialization function
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

静電容量式タッチスライダ / ホイールフレームワークモジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

sf_touch_ctsu 上の静電容量式タッチフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_sf_touch_ctsu0	Module name.
Thread Priority	3	User determined based on priority of other threads in system. Recommend high priority.

ISDE Property	Value	Description
Update Hz	50	Rate at which the CTSU hardware scans should occur. The maximum scan rate should be less than the tick rate set for ThreadX. The total time to complete a hardware scan is determined by the number of channels used. Each channel takes approximately 600 usec to complete a scan. Thus if 10 channels are used for 10 buttons in self-capacitance mode, the total hardware scan time is ~6 milliseconds or 166 Hz. Similarly, if 7channels are used for 5 buttons in mutual-capacitance mode (5x2 layout), the total hardware scan time is ~4.2 milliseconds or 250 Hz.The software processing time is additional and will vary depending on the core clock and the software filter depth used by the CTSU. Thus the maximum scan rate should be lesser than the time required to scan and process all the channels used in the configuration and lesser than the RTOS tick rate.
Callback	NULL	Name of user callback that will be called when each scan is complete and also when new processed data is available. Can be NULL if not used.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_ctsu 上の CTSU HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking Enable	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Offset Adjustment	Enabled, Disabled Default: Enabled	Used to enable or disable offset adjustment. Leave as enabled unless you are tuning the board. (Rate is determined by *Runtime rate of tuning of sensor values*.)
Drift Compensation	Enabled, Disabled Default: Enabled	Used to enable or disable drift compensation. Leave as enabled unless you are tuning the board. Drift compensation allows the baseline values used to determine the presence or absence of a touch to change over time as the board or surface ages or with changes in temperature and environment. Drift compensation rate is determined by *Steady state drift compensation rate*, *Startup drift compensation rate*, and *Channel release compensation rate*.
Drift Compensation Method (valid only if Drift Compensation is enabled above)	Alternate Method 1	Drift Compensation algorithms provided. Only Alternate 1 is currently supported. Other options may be supported in future releases. Drift compensation method Alternate 1 allows for different rates of drift compensation for the following events: steady state, startup, and channel release.

ISDE Property	Value	Description
Steady state drift compensation rate, drift compensation will be applied per n scans	500	Determines the rate of drift compensation (applied every n scans) when the channel is in steady state. (Dependent on the scan rate.)
Startup drift compensation rate (Should be less than the steady state drift compensation)	5	Determines the rate of drift compensation (applied every n scans) when the driver is performing its initialization. (Should be less than the *Steady state drift compensation rate*.)
Channel release compensation rate (Should be less than the steady state drift compensation rate)	500	Determines the rate of drift compensation (applied every n scans) when a channel transitions from pressed to released. (Should be less than or equal to the *Steady state drift compensation rate*.)
Default filter depth (used in sensor count filter provided by driver)	1	Used in the software sensor count filter provided by driver. This default filter is a modification of the relatively common "leaky integrator" software filter. A depth of 4 equates to an approximate filter constant of 16. When the input is a constant value of 16 or greater, the filter output will reach 68-69% of the constant input value after 16 cycles/iterations of the filter.
Runtime rate of tuning of sensor values (if drift compensation is used this value is overrriden to be twice the rate of the steady state drift compensation)	800	Rate of offset adjustment. (If drift compensation is used this value is set to twice the rate of the steady state drift compensation.)
Perform auto-tune and drift compensation only when all channels are untouched	True, False Default: True	Perform auto-tune and drift compensation only when all channels are untouched.

ISDE Property	Value	Description
Max. active channels	1	Specify the maximum number of channels that are to be used by the application. In Self Capacitance Mode, this is the number of channels used. In Mutual Capicitace Mode, this is the multiplication result of the Rx and Tx channels. For example: 4 Rx and 3 Tx channels = 12 Maximum Active Channels.
Name	g_ctsu0	Module name.
CTSU configuration used	g_ctsu0_config	Name of the configuration structure generated by CTW.
Callback	NULL	The user callback function that will be invoked each time the scan is complete as well as when newly processed data is available. Can be set to NULL if not used.
Data Processing Option	Default Processing (Recommended), No Processing (Tuning only) Default: Default Processing	Can be used to specify if scans should start after the data from the previous one is completed, if auto-calibration should be run between scans etc. Use the Default Processing unless you are tuning.
Write Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Write interrupt priority selection.
Read Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Read interrupt priority selection.

ISDE Property	Value	Description
End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	End interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event CTSU END 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Enabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Module name
Mode	Block	Mode selection
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection

ISDE Property	Value	Description
Source Pointer	NULL	Source pointer selection
Number of Transfers	1	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	1	Number of blocks selection
Activation Source (Must enable IRQ)	Event CTSU END	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC software event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

静電容量式タッチスライダ / ホイールフレームワークモジュールのクロック構成

CTSU スライダ / ホイールフレームワークは CTSU HAL ドライバーと同じ構成を使用します。CTSU クロック速度を、Capacitive Touch Workbench for Renesas Synergy（CTW）で選択したクロック速度と同じ値に設定します。

静電容量式タッチスライダ / ホイールフレームワークモジュールのピン構成

CTSU スライダ / ホイールフレームワークは CTSU HAL ドライバーと同じ構成を使用します。外部デバイスの要求に応じて、ピンをタッチセンサーピン TSxx として設定し、TSCAP 機能を有効化します。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では I2C ピンの選択例を示します。

注 : 一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号が決まり、それに従って必要な MCU ピンが決定されます。

静電容量式タッチセンシングユニット（CTSU）のピン選択

Resource	ISDE Tab	Pin selection Sequence
CTSU	Pins	Select Peripherals > Input: CTSU > CTSU0

CTSU のピン構成設定

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Enabled Default: Disabled	Select Enabled as the Operation Mode for CTSU
TS00 to TS11	None, Pn, Pm Default: None	TS Pins
TSCAP	None, Pn, Pm Default: None)	TSCAP Pin

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.1.21.6 アプリケーションでの静電容量式タッチスライダ / ホイールフレームワークモジュールの使用

静電容量式タッチスライダ / ホイールフレームワークを使用する前に、ISDE のプロジェクトにフレームワークを追加し、CTW の構成データがプロジェクトに追加されていることを確認して、ボタンの数、CTW によって生成された構成構造体の名前、使用するコールバックなどのフレームワークのプロパティを更新します。これを実行した後、アプリケーションで静電容量式タッチスライダ / ホイールフレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_touch_ctsu_slider_api_t::open API を使用して静電容量式タッチスライダ / ホイールフレームワークモジュールを初期化します。
2) コールバック関数で、タッチボタンイベントを検出し、処理します。
3) sf_touch_ctsu_slider_api_t::disable API を使用してボタンのコールバックを無効化します（オプション）。
4) sf_touch_ctsu_slider_api_t::enable API を使用してスライダ / ホイールのコールバックを有効化します（オプション）。
5) sf_touch_ctsu_slider_api_t::close API を使用して静電容量式タッチスライダ / ホイールフレームワークモジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

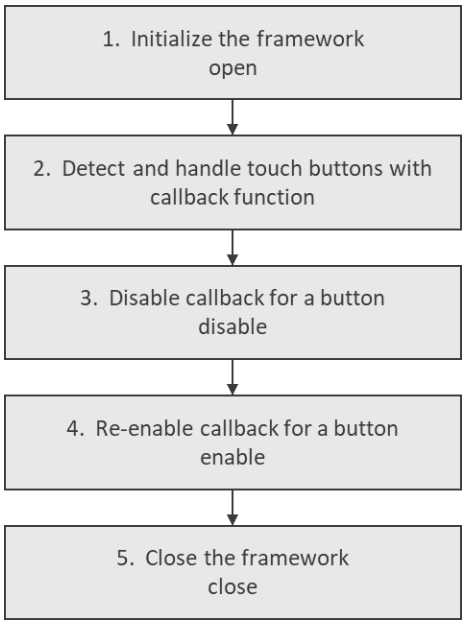


図 84:通常の静電容量式タッチスライダ / ホイールフレームワークモジュールアプリケーションのフロー図

4.1.22 外部 IRQ フレームワーク

外部 IRQ フレームワークは、ThreadX RTOS による外部ピン割り込みを使用するアプリケーションにハイレベル API を提供し、Synergy マイクロコントローラ上の外部 IRQ ピンをサポートします。IRQn がトリガされるたびに割り込みサービスルーチン（ISR）からコールされるコールバック関数（sf_external_irq_callback）が使用可能です。

4.1.22.1 外部 IRQ フレームワークモジュールの特長

- 外部割り込み入力に応答します
- RTOS はスレッド同期に内部セマフォを使用して実装を認識します
  - 内部スレッドに信号を送信できます
  - イベントリンクコントローラ（ELC）を介して転送をトリガできます
- Synergy MCU で使用可能なポートピンを使用します
  - ピンは MCU 間で異なる可能性があるため、詳細については MCU のユーザーズマニュアルを参照してください
- 以下のような複数のハードウェア機能をサポートします
  - チャネルの選択
  - トリガ条件
  - デジタルフィルタリング

– オートスタート

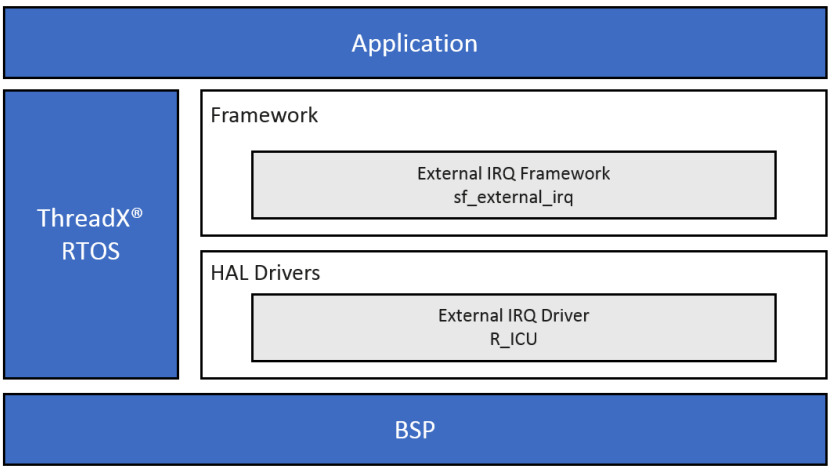


図 85:外部 IRQ フレームワークモジュールのブロック図

4.1.22.2 外部 IRQ フレームワークモジュール API の概要

外部 IRQ フレームワークモジュールは、モジュールのオープン、待機、クローズの API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

外部 IRQ フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_external_irq.p_api->open(g_sf_external_irq.p_ctrl, g_sf_external_irq.p_cfg); Acquire mutex, then handle driver initialization at the HAL layer.
wait	g_sf_external_irq.p_api->wait(g_sf_external_irq.p_ctrl, TX_WAIT_FOREVER); Wait for the next external interrupt expiration, then return.

Function Name	Example API Call and Description
versionGet	g_sf_external_irq.p_api->versionGet(&version); Retrieve the API version and store it in the version pointer.
close	g_sf_external_irq.p_api->close(g_sf_external_irq.p_ctrl); Release channel mutex and close channel at HAL layer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successful.
SSP_ERR_ASSERTION	Assertion error.
SSP_ERR_IN_USE	Device in use.
SSP_ERR_NOT_OPEN	Device unopened.
SSP_ERR_TIMEOUT	Timeout error.
SSP_ERR_WAIT_ABORTED	Suspension aborted.
SSP_ERR_UNSUPPORTED	Function unsupported by the HAL driver.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.22.3 外部 IRQ フレームワークモジュールの動作の概要

外部 IRQ フレームワークは、ThreadX 対応の一連のフレームワーク API です。外部 IRQ フレームワークの外部入力は、内部セマフォを通じてスレッドに信号を送信したり、イベントリンクコントローラ（ELC）を通じて転送をトリガしたりすることができます。適切な動作のために、外部 IRQ フレームワークモジュールと外部 IRQ HAL モジュールの両方を構成する必要があります。HAL 構成設定では、トリガレベルやデジタルフィルタリング設定などのハードウェアオプションを制御できます。

外部 IRQ フレームワークモジュールの動作に関する重要な注意事項と制限事項

- 外部割り込み機能をサポートするポート ピンを調べるときや、特定のポート ピンの外部 IRQ 番号が知りたいときには、プログラムする Synergy デバイスのデータシートを参照してください。
- 外部 IRQ 番号は、ISDE で外部 IRQ ドライバーの [Properties] ウィンドウに示されるチャネルの設定に対応します。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.22.4 アプリケーションへの外部 IRQ フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して外部 IRQ フレームワークモジュールをアプリケーションに組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

外部 IRQ フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(外部 IRQ フレームワークのデフォルト名は g_sf_external_irq0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

外部 IRQ フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_external_irq0 External IRQ Framework on sf_external_irq	Threads	New Stack> Framework> Input> External IRQ Framework on sf_external_irq

次の図に示すように sf_external_irq 上の外部 IRQ フレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

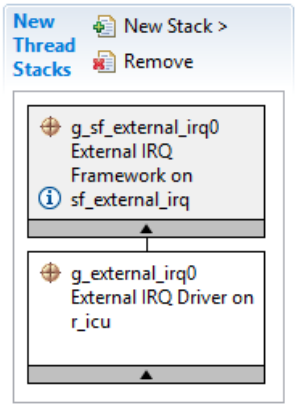


図 86:外部 IRQ フレームワークモジュールスタック

4.1.22.5 外部 IRQ フレームワーク モジュールの設定

ユーザーは必要な動作に合わせて外部 IRQ フレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_external_irq での外部 IRQ フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Controls whether to include code for API parameter checking.
Name	g_sf_external_irq0	Framework name.
Event	None, Semaphore Put Default: Semaphore Put	Event selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

外部 IRQ フレームワークモジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_icu での外部 IRQ HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter checking setting enables or disables the addition of parameter checking code.
Name	g_external_irq0	Module name.
Channel	0	Specifies the hardware IRQ channel used.
Trigger	Falling, Rising, Both Edges, Low Level Default: Rising	Configures edge or level triggering.
Digital Filtering	Enabled, Disabled Default: Disabled	Digital filter enable/disable.
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCLK/1, PLCK/8, PLCK/32, PCLK/64 Default: PCKL/64	Sets noise filter sampling period.
Interrupt enabled after initialization	True, False Default: True	Determines if the interrupt is enabled immediately after initialization.

ISDE Property	Value	Description
Callback	NULL	A user callback function can be registered in external_irq_api_t::open. If this callback function is provided, it is called from the interrupt service routine (ISR) each time the IRQn triggers. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	An Interrupt priority can be registered in external_irq_cfg_t::irq_ipl.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

外部 IRQ フレームワークモジュールのクロック構成

外部 IRQ フレームワークには、クロック構成は不要です。

外部 IRQ フレームワークモジュールのピン構成

外部 IRQ フレームワークには、ピン構成は不要です。

4.1.22.6 アプリケーションでの外部 IRQ フレームワークモジュールの使用

一般的なアプリケーションで外部 IRQ フレームワークモジュールを使用する際の手順は次のとおりです。

1) sf_external_irq_api_t::open API で外部 IRQ フレームワークモジュールを開きます。
2) sf_external_irq_api_t::wait API を使用して割り込みを待ちます。
3) 外部 IRQ イベントを処理します
4) sf_external_irq_api_t::close API を使用して、モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

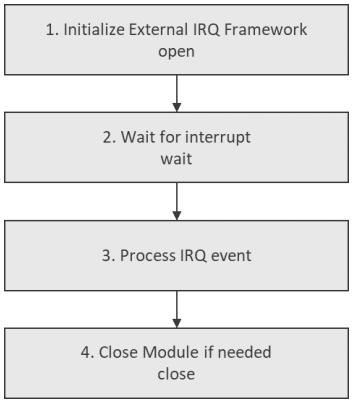


図 87:通常の外部 IRQ フレームワークモジュールアプリケーションのフロー図

4.1.23 I2C フレームワーク

I2C フレームワークモジュールは業界標準の I2C シリアルデバイス通信に ThreadX 対応のハイレベル API を提供し、I2C ペリフェラルを構成することで、フレームワークで使用されるシリアル通信を有効化します。I2C フレームワークモジュールは、Synergy MCU 上の I2C および SCI ペリフェラルを使用します。

4.1.23.1 I2C フレームワークモジュールの特長

- ThreadX 対応フレームワーク
- I2C バス上の複数の I2C 周辺機器の統合と同期を処理します
- SCI I2C ドライバーと RIIC ドライバーの両方にアクセスするための単一のインタフェースを提供します
- I2C フレームワークモジュールはマスタモードで I2C 通信を構成します
- I2C フレームワークモジュールは 100kHz、400kHz、1MHz の 3 つのデータレートをサポートします
- I2C フレームワークモジュールは 7 ビットアドレッシングと 10 ビットアドレッシングの両方をサポートします
- I2C　フレームワークモジュールは、内部的なコールバックのサポートも提供します。ユーザー定義のコールバックは使用されません。コールバック関数は、次のイベントで呼び出されます。
  - 転送中断
  - 転送完了
  - 受信完了
- コールバック構造体 i2c_callback_args_t には、送信または受信したバイト数も設定されます。
- 以下によって実装されます。
  - SCI 上の簡易 I2C
  - RIIC

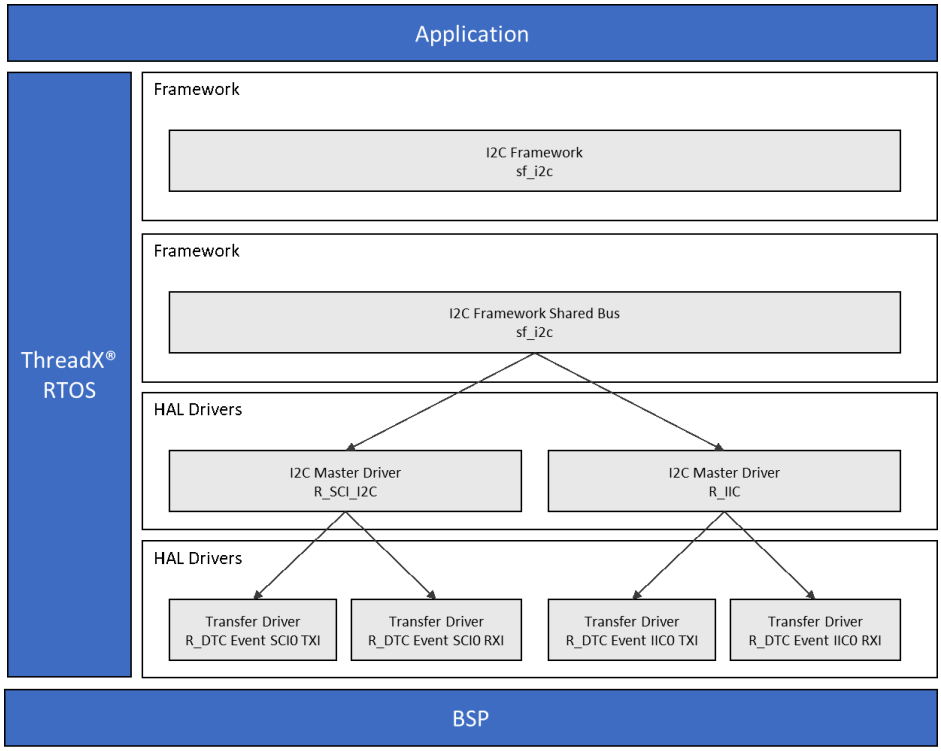


図 88:I2C フレームワークモジュールのブロック図

4.1.23.2 I2C フレームワークモジュール API の概要

I2C フレームワークインタフェースは、I2C フレームワークを使用したバスのオープン、クローズ、リード、ライト、ロック、ロック解除、リセットの API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

I2C フレームワークモジュール API の要約

Function Name	Example API Call and Description
open	g_sf_i2c_device.p_api->open(g_sf_i2c_device.p_ctrl, g_sf_i2c_device.p_cfg); Opens a designated I2C device on the bus.

Function Name	Example API Call and Description
close	g_sf_i2c_device.p_api->close (g_sf_i2c_device.p_ctrl); Disables I2C device designated by control handle. Closes the RTOS services used by the bus if no devices are connected to the bus.
read	g_sf_i2c_device.p_api->read (g_sf_i2c_device.p_ctrl, &reg, no_of_bytes_to_read, 0, TX_WAIT_FOREVER); Receives data from I2C device.
write	g_sf_i2c_device.p_api->write (g_sf_i2c_device.p_ctrl, command, no_of_bytes_to_write , false, TX_WAIT_FOREVER); Transmits data to I2C device.
lock	g_sf_i2c_device.p_api->lock (g_sf_i2c_device.p_ctrl); Locks the bus for a device. Locking reserves the bus until unlocking and allows several reads and writes without interrupt.
unlock	g_sf_i2c_device.p_api->unlock (g_sf_i2c_device.p_ctrl); Unlocks the bus from a particular device and makes it available for other devices.
reset	g_sf_i2c_device.p_api->reset (g_sf_i2c_device.p_ctrl, TX_NO_WAIT); Aborts any in-progress transfer and forces the I2C peripheral into ready state.

Function Name	Example API Call and Description
version	g_sf_i2c_device.p_api->version(version); Retrieves the version information using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	I2C function performed successfully
SSP_ERR_INVALID_MODE	Illegal I2C mode is specified
SSP_ERR_INVALID_CHANNEL	Omitted I2C channel is specified
SSP_ERR_IN_USE	I2C channel has already been opened
SSP_ERR_INVALID_ARGUMENT	Argument is not one of the predefined values
SSP_ERR_INVALID_POINTER	Null pointer(s) is (are) given
SSP_ERR_INTERNAL	Internal error has occurred
SSP_ERR_ASSERTION	A critical assertion has failed
SSP_ERR_NOT_OPEN	Device instance not opened
SSP_ERR_TRANSFER_ABORTED	The data transfer was aborted
SSP_ERR_INVALID_RATE	The requested rate cannot be set
SSP_ERR_TIMEOUT	Timeout error occurs.

注：ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.23.3 I2C フレームワークモジュールの動作の概要

I2C フレームワークモジュールは、SSP の階層化されたドライバーアーキテクチャに準拠しています。ローレベルの I2C HAL モジュールを使用して I2C ペリフェラルと通信し、(ユーザーの構成に従って) Synergy マイクロコントロー

ラ上の I2C 対応ペリフェラルを制御します。I2C フレームワークモジュールを使用すると、1 つ以上の I2C バスを作成し、複数のI2C周辺機器をそれぞれのI2Cバスに接続できます。I2CフレームワークモジュールAPIは、ThreadX-Mutexを使用して、I2C スレーブデバイスの共有バスを取得および解放します。取得とリリースは、それぞれ I2C フレームワークモジュールの sf_i2c_api_t::lock および sf_i2c_api_t::unlock API によって実装されます。

I2C フレームワークモジュールはマスタモードで I2C 通信を構成するため、ユーザーは次の操作を行うことができます。

- スレーブ デバイスからの読み取り
- スレーブ デバイスへの書き込み
- I2C 周辺機器のリセット
- MCU I2C ペリフェラルのリセット
- I2C バスのロック
- I2C バスのロック解除

I2C フレームワークモジュールは、Synergy MCU I2C ハードウェアモジュール（RIIC および SCI HAL モジュール）と連携します。両方の I2C モジュールが、最大ビットレート 400kHz の I2C 高速モードをサポートしています。IIC ペリフェラルと RIIC HAL モジュールは、ビットレートが 1MHz の高速モードプラスをサポートしています。このモジュールは、両方の実装について、マスタモードのみをサポートしています。

同じバス上での複数のスレーブデバイスの接続

I2C フレームワークモジュールは、バスとバス上のデバイスによるアーキテクチャを使用します。複数のスレーブが I2C バスに接続されている場合は、各スレーブが、関連付けられた個別の SF_I2C モジュールインスタンスと通信します。各 SF_I2C インスタンスは、別のスレッドで作成されます。どのスレーブデバイスも、接続先のバスに関連付けられており、他のすべてのスレーブデバイスとバスを共有します。ユーザーは、フレームワーク共有バスと、バスに接続されている各フレームワークデバイスのローレベル I2C HAL レイヤを構成する必要があります。ユーザーは、複数のデバイスを同じバス上で構成するときに既存のフレームワーク共有バスを追加することができます。共通の開始および停止手順が、すべての I2C データ転送操作に使用されます。ローレベルに対してはデバイスを 1 つだけ構成します。残りのデバイスは、フレームワーク内のデバイスアドレスを切り替えることによって、リードまたはライト動作を実行します。

同じバス上のすべての I2C フレームワークデバイスは、同じローレベル構成設定（I2C HAL モジュールなど）を使用しますが、スレーブアドレスとアドレシングモードは例外です。フレームワークは、アプリケーションで最初に開かれるデバイスの構成を使用して、バスを構成します。つまり、同じバス上のすべての I2C フレームワークデバイスは、ローレベル構成設定が同じでなければなりません（スレーブアドレスとアドレシングモードを除く）。異なる構成が使用されると、適切な動作は保証されません。

バスロック

I2C フレームワークはバスロック機能をサポートするため、特定のスレーブペリフェラルに対してバスをロックできます。ロックすると、ロックコマンドからロック解除コマンドまでの期間、デバイスがバスを予約できるようになります。これにより、（一部の状況に対応して）デバイスが複数のリードおよびライトを中断することなく完了できるようになります。

sf_i2c_api_t::lock API がコールされると、I2C バスはロックされます。この API は、I2C フレームワークデバイスを使用するスレッド用にミューテックスを取得することで、I2C バスをロックします。ロックされると、I2C バスは関連付けられたデバイスによってのみ利用可能となります。他のスレッドからアクセスする他の I2C フレームワークデバイスはミューテックスを取得できないため、バスにアクセスすることはできません。バスをロックした sf_i2c デバイスが sf_i2c_api_t::unlock API をコールし、バスがロック解除されると、ミューテックスは解放され、バスは他の sf_i2c デバイスから利用できるようになります。

I2C フレームワークモジュールの動作に関する重要な注意事項と制限事項

- 現在の PCLKB 設定で達成可能な最も近いボー レート (要求されたレートより小さいか等しいもの) が計算されて使用されます。有効なクロック レートを計算できなかった場合は、エラーが返されます。
- I2C は、ELC から使用可能な他の周辺機器の起動をトリガできます。詳細については、『ELC Module Guide』を参照してください。

- すべてのデバイスに対しクロックレートが同じに保たれる場合、I2C フレームワークは、同一バス上の複数の I2C デバイスをサポートできます。つまり、クロックレートが同じであれば、同一バス内で複数のデバイスを開くことができます。デバイスのクロックレートが異なる場合、一度に開くことのできるデバイスは 1 つのみです。
- SCI で I2C を使用する場合、SDA および SCL 出力ピンタイプは、n チャネルオープンドレインである必要があります。
- I2C フレームワーク構成では、このバスに指定されたチャネル番号により HAL モジュールで指定されたチャネル番号が上書きされます。
- 共有バスは、それぞれの構成を持つ複数のスレーブデバイスで使用できます。複数のデバイスが同じ I2C チャネルを使用している場合、フレームワークは lock および unlock API で相互排除も処理します。
- 複数の I2C デバイスを同じバス上で構成するには、バスに接続する各デバイスについて、以下のモジュールを追加して構成します。
  - I2C フレームワークデバイスモジュール
  - 最初のデバイスの構成で I2C 共有バスモジュールを構成してから、残りのデバイスに対して同じバスを使用します。
  - I2C HAL モジュール
  - DTC module(Optional)
- ロック機能は、異なるスレッドのデバイスに対して有効になります。バスに接続されている複数のデバイスが同じスレッドからのものである場合、I2C バスはそのスレッドのすべてのデバイスに対してロックされます。このような場合、バスがロックされたとしても、同じスレッドのすべてのデバイスがバスにアクセスできます。

注 : 各 I2C フレームワークデバイスは、ISDE コンフィギュレータで一意の名前を使用して構成する必要があります。
注 : 同じバス上で接続されているすべてのデバイスに同じ構成設定を指定する必要があります（スレーブアドレスとアドレシングモードは例外です。）

I2C フレームワークモジュールでは現在、以下の機能はサポートされていません。

- GPT 以外のタイマの使用
- DMA の使用

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.23.4 アプリケーションへの I2C フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して I2C フレームワークモジュールをアプリケーションに組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

I2C フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。(I2C フレームワークモジュールのデフォルト名は g_sf_ i2c_device0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

I2C フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_i2c_device0 I2C Framework on sf_i2c	Threads	New Stack> Framework> Connectivity> I2C Framework on sf_i2c

次の図に示すように、sf_i2c の I2C フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

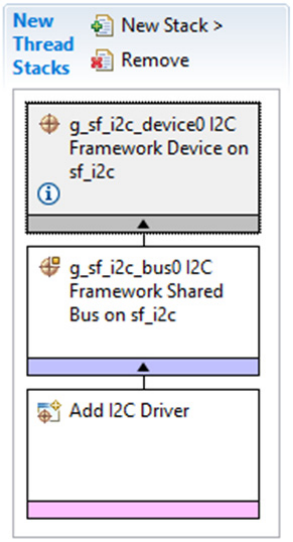


図 89:I2C フレームワークモジュールのスタック

4.1.23.5 I2C フレームワーク モジュールの設定

ユーザーは必要な動作に合わせて I2C フレームワークモジュールを構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成テーブルの設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_ i2c の I2C フレームワークデバイスモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: Enabled	Selects if code for parameter checking is to be included in the build.
Name	g_sf_i2c_device0	Give a name to identify the I2C Framework device. API, Config and Control instances will be created based on this name.
Slave Address	0x00	Specify the address of the I2C slave device.
Address Mode	7-Bit, 10-Bit Default: 7-Bit	Select the I2C address mode.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

デフォルト以外の設定が望ましい場合もあります。たとえば、異なるバイト順序やピクセルカラーフォーマットの選択が役立つ場合です。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

I2C フレームワークのローレベルモジュールの構成

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、ローレベルモジュールの [properties] セクションのすべての設定を示します。

sf_i2c 上の I2C フレームワーク共有バスの構成設定

ISDE Property	Value	Description
Name	g_sf_i2c_bus0	Module name.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_riic 上の I2C マスタ ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Name	g_i2c0	Module name.
Channel	0	Specify the IIC channel to be used with this configuration.
Rate	Standard, Fast-mode, Fast-mode Plus Default: Standard	Standard, Fast, and Fast-plus. (See IIC Rate Calculation.)
Slave Address	0x00	Set the address of the slave device the I2C master will be communicating with.
Address Mode	7-Bit, 10-Bit Default: 7-Bit	Only 7-bit addresses are currently supported.
Timeout Mode	Short Mode, Long Mode Default: Short Mode	Select the timeout mode.

ISDE Property	Value	Description
Callback	NULL	A user callback function can be registered in i2c_api_master_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) for each of the conditions defined in i2c_event_t. Warning: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Default: Priority 12	Select the receive interrupt priority.
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Default: Priority 12	Select the transmit interrupt priority.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Default: Priority 12	Select the transmit end interrupt priority.
Error Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Default: Priority 12	Select the error interrupt priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event IIC0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Software Start	Enabled, Disabled Default: Disabled	Include code for software start in the build.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Section to place the DTC vector table.
Name	g_transfer0	Module name.
Mode	Normal	Specify the hardware channel.
Transfer Size	1 Byte	Select the transfer mode.
Destination Address Mode	Fixed	Select the transfer size.
Source Address Mode	Incremented	Select the address mode for the destination.
Repeat Area (Unused in Normal Mode)	Source	Select the address mode for the source.
Interrupt Frequency	After all transfers have completed	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.
Number of Transfers	0	Specify the number of transfers.

ISDE Property	Value	Description
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source (Must enable IRQ)	Event IIC0 TXI	Select the DTC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback (Only valid with Software start)	NULL	A user callback that is called at the end of the transfer.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event IIC0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Software Start	Enabled, Disabled Default: Disabled	Include code for software start in the build.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Section to place the DTC vector table.
Name	g_transfer1	Module name.
Mode	Normal	Specify the hardware channel.

ISDE Property	Value	Description
Transfer Size	1 Byte	Select the transfer mode.
Destination Address Mode	Incremented	Select the transfer size.
Source Address Mode	Fixed	Select the address mode for the destination.
Repeat Area (Unused in Normal Mode	Destination	Select the address mode for the source.
Interrupt Frequency	After all transfers have completed	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.
Number of Transfers	0	Specify the number of transfers.
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source (Must enable IRQ)	Event IIC0 RXI	Select the DTC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback (Only valid with Software start)	NULL	A user callback that is called at the end of the transfer.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sci_i2c 上の I2C マスタドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Name	g_i2c0	Module name.
Channel	0 to 9	Specify the SCI channel to be used with this configuration. SCI has channels as follows: Series S7: 0 1 2 3 4 5 6 7 8 9; Series S3 : 0 1 2 3 4 - - - - 9; Series S1 : 0 1 - - - - - - - 9 .
Rate	Standard, Fast-mode, Fast-mode plus Default: Standard	Select the I2C data rate.
Slave Address	0x00	Specify the slave address.
Address Mode	7-Bit, 10-Bit Default: 7-Bit	Only 7-bit addresses are currently supported.
SDA Output Delay (nano seconds)	300	SDA output delay in nanoseconds.
Bit Rate Modulation Enable	Enable, Disable Default: Enable	Enables bitrate modulation function.

ISDE Property	Value	Description
Callback	NULL	A user callback function can be registered in i2c_api_master_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) for each of the conditions defined in i2c_event_t. Warning: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Select the receive interrupt priority.
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Select the transmit interrupt priority.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Select the transmit end interrupt priority.
Error Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Select the error interrupt priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Software Start	Enabled, Disabled Default: Disabled	Include code for software start in the build.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Section to place the DTC vector table.
Name	g_transfer0	Module name.
Mode	Block	Specify the hardware channel.
Transfer Size	1 Byte	Select the transfer mode.
Destination Address Mode	Fixed	Select the transfer size.
Source Address Mode	Incremented	Select the address mode for the destination.
Repeat Area (Unused in Normal Mode	Source	Select the address mode for the source.
Interrupt Frequency	After all transfers have completed	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.
Number of Transfers	0	Specify the number of transfers.

ISDE Property	Value	Description
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source (Must enable IRQ)	Event SCI0 TXI	Select the DTC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback (Only valid with Software start)	NULL	A user callback that is called at the end of the transfer.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Software Start	Enabled, Disabled Default: Disabled	Include code for software start in the build.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Section to place the DTC vector table.
Name	g_transfer1	Module name.
Mode	Normal	Specify the hardware channel.

ISDE Property	Value	Description
Transfer Size	1 Byte	Select the transfer mode.
Destination Address Mode	Incremented	Select the transfer size.
Source Address Mode	Fixed	Select the address mode for the destination.
Repeat Area (Unused in Normal Mode	Destination	Select the address mode for the source.
Interrupt Frequency	After all transfers have completed	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.
Number of Transfers	0	Specify the number of transfers.
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source (Must enable IRQ)	Event SCI0 RXI	Select the DTC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback (Only valid with Software start)	NULL	A user callback that is called at the end of the transfer.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

I2C フレームワークモジュールのクロック構成

SCI ペリフェラルモジュールは PCLKB をそのクロックソースとして使用します。PCLKB 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。構成時に、I2C 転送レートは、ユーザー選択 PCLB レートとユーザー選択転送レートに基づいて、ドライバーにより内部で計算および設定されます。ユーザー選択レートを実現できないような構成が PCLKB で行われた場合は、ドライバーを初期化するときにエラーが返されます。

I2C フレームワークモジュールのピン構成

SCI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

注 : 一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号が決まり、それに従って必要な MCU ピンが決定されます。

I2C フレームワークモジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SCI	Pins	Select Peripherals > SCI1_3_5_7_9 > SCI1

注 : 選択シーケンスでは、SCI1 がドライバーに必要なハードウェアターゲットであることを想定しています。

I2C フレームワークモジュールのピン構成設定

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Asynchronous UART, Synchronous UART, Simple I2C, Simple SPI, SmartCard Default: Disabled	Select Simple I2C as the Operation Mode for SPI on SCI
RXD1_SCL1_MISO1	None, P212, P708 Default: None	SCL Pin

Pin Configuration Property	Value	Description
TXD1_SDA1_MOSI1	None, P213, P709 Default: None	SDA Pin

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

I2C フレームワークモジュールの追加設定

SCL ピンと SDA ピンに加えて、I2C スレーブデバイスをリセットするために I2C RESET 信号が必要な場合があります。この場合、RESET 信号は GPIO を使用して追加でき、アプリケーションプログラムで直接制御する必要があります。外部デバイスのリセット機能は、r_sci_i2c モジュール内でサポートされていません。

4.1.23.6 アプリケーションでの I2C フレームワークモジュールの使用

I2C フレームワークモジュールの共通アプリケーションには、単一バス上に複数のスレーブデバイスが必要です。この共通アプリケーションの実装については、以下で説明しています。(複数のバスが必要なアプリケーションについては、必要な数のバスそれぞれについて、単一バス向けの手順を繰り返してください。)

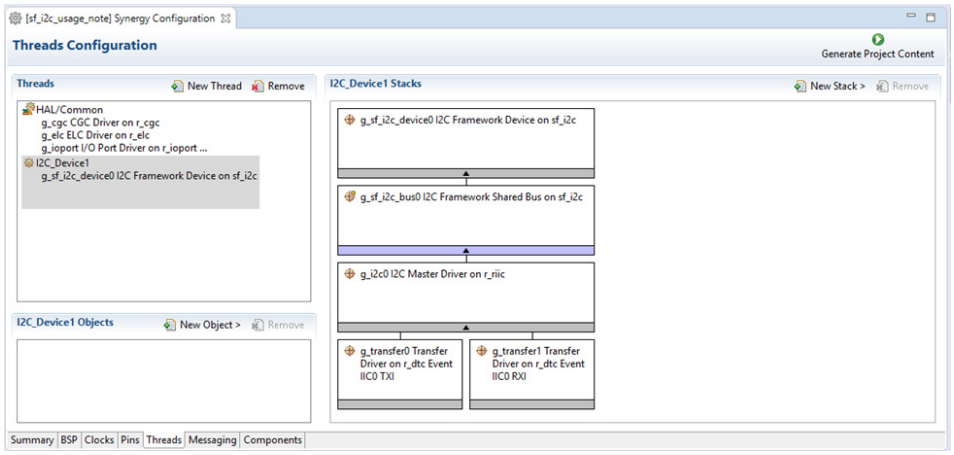
同じ共有バス上の 2 つのスレーブデバイスに対する実装手順

I2C フレームワークモジュールを使用して複数のスレーブデバイスを接続した単一のバスを作成するには、2 つのスレッドスタックを作成し、それぞれが I2C フレームワークインスタンスを使用するようにします。これらのインスタンスは、どちらも同じ共有バスインスタンスを使用します。以下の手順に従って、SSP コンフィギュレータでこの操作を行う方法を確認してください。

注 : 以下の例では、両方の sf_i2c モジュールインスタンスを同じスレッドに追加しています。バスのロック機能が必要な場合は、sf_i2C モジュールを別々のスレッドに追加する必要があります。ロックは、ロックされたスレッドのすべてのデバイスに適用されます。注 : 以下の手順は、SSP 開発環境の使用について、ある程度習熟していることを前提としています。以下の手順でわかりにくい点がある場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を読んで、SSP 開発環境に習熟するようにしてください。

1) 最初の I2C フレームワークデバイスを新しいスレッドまたは既存のスレッドに追加します。これにより、I2C マスタスタックが作成されます。sf_i2c 上の共有バスが、I2C ドライバーとともに追加されます。I2C ドライバーは、r_riic または r_sci_i2c. 上での実装のために選択できます。デフォルトで DTC 転送ドライバーも追加されます。このドライバーは、代わりに CPU 転送モードが必要な場合には削除できます。

結果のモジュールスタックを次の図に示します。図の下の表に、構成設定の例を示します。



以下に、最初のスレッドスタックの構成設定の例を示します。

sf_i2c 上の I2C フレームワークデバイスの構成設定（スレーブ #1）

Property	Value	Description
Parameter Checking	Default(BSP)	Enable Or Disable Parameter Checking.
Name	g_sf_i2c_device0	Give a name to identify the I2C Framework device. API, Config and Control instances will be created based on this name.
Slave Address	0x21	Specify the address of I2C slave 1.
Address Mode	7-bit	Select the I2C address mode.

sf_i2c 上の I2C フレームワーク共有バスの構成設定

Property	Value	Description
Name	g_sf_i2c_bus0	Give a name to identify the I2C Framework shared bus. This shared bus will be shared by multiple I2C Framework Devices.

r_riic 上の I2C マスタ ドライバーの構成設定

Property	Value	Description
Parameter Checking	Default(BSP)	Enable Or Disable Parameter Checking.
Name	g_i2c0	Give a name to identify the I2C Driver device. This will be used internally by Framework.
Channel	0	Specify the I2C channel.
Rate	Standard	Select the speed of the I2C bus.
Slave Address	0	This field will be locked as slave address already configured in the I2C Framework Device on sf_i2c.
Address Mode	7-bit	This field will be locked as address mode already configured in the I2C Framework Device on sf_i2c.
Timeout Mode	Short Mode	Select Timeout mode: Short mode or Long mode
Callback	NULL	This field will be locked as Framework does not provide callback handling to the user.
Receive Interrupt Priority	Priority 2	Receive interrupt priority selection.
Transmit Interrupt Priority	Priority 2	Transmit interrupt priority selection.
Transmit End Interrupt Priority	Priority 2	Transmit end interrupt priority selection.
Error Interrupt Priority	Priority 2	Error interrupt priority selection.

r_sci_i2c 上の I2C マスタドライバーの構成設定

Property	Value	Description
Parameter Checking	Default(BSP)	Enable Or Disable Parameter Checking.
Name	g_i2c0	Give a name to identify the I2C Driver device. This will be used by Framework internally.
Channel	0	Specify the address of I2C slave.
Rate	Standard	Select the speed of the I2C bus.
Slave Address	0	This field will be locked as slave address already configured in the I2C Framework Device on sf_i2c.
Address Mode	7-bit	This field will be locked as address mode already configured in the I2C Framework Device on sf_i2c.
Slave Output Delay	300	SDA output delay in nanoseconds.
Bit Rate Modulation Enable	Enable	Enables bitrate modulation function.
Callback	NULL	This field will be locked as Framework does not provide callback handling to the user.
Receive Interrupt Priority	Priority 2	Receive interrupt priority selection.
Transmit Interrupt Priority	Priority 2	Transmit interrupt priority selection.
Transmit End Interrupt Priority	Priority 2	Transmit end interrupt priority selection.

r_dtc Event IIC0 TXI 時の転送ドライバーの構成設定

Property	Value	Description
Parameter Checking	Default(BSP)	Enable Or Disable Parameter Checking.
Software Start	Disabled	
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Module name.
Mode	Normal	Mode selection. This field is locked.
Transfer Size	1 Byte	Transfer size selection. This field is locked.
Destination Address Mode	Fixed	Destination address mode selection. This field is locked.
Source Address Mode	Incremented	Source address mode selection. This field is locked.
Repeat Area (Unused in Normal Mode	Source	Repeat area selection. This field is locked.
Interrupt Frequency	After all transfers have completed	This field is locked.
Destination Pointer	NULL	Destination pointer selection. This field is locked.
Source Pointer	NULL	Source pointer selection. This field is locked.
Number of Transfers	0	Number of transfer selection. This field is locked.
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection. This field is locked.
Activation Source (Must enable IRQ)	Event IIC0 TXI	Activation source selection. This field is locked.

Property	Value	Description
Auto Enable	False	Auto enable selection. This field is locked.
Callback (Only valid with Software start)	NULL	Callback selection. This field is locked.
ELC Software Event Interrupt Priority	Disabled	ELC software event interrupt priority selection.

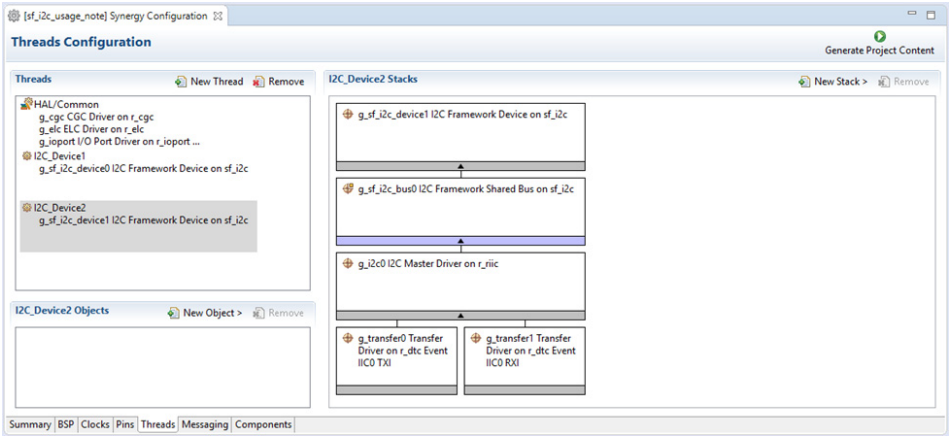
r_dtc Event IIC0 RXI 時の転送ドライバーの構成設定

Property	Value	Description
Parameter Checking	Default(BSP)	Enable or Disable Parameter Checking.
Software Start	Disabled	Software Start Selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Module name.
Mode	Normal	Mode selection. This field is locked.
Transfer Size	1 Byte	Transfer size selection. This field is locked.
Destination Address Mode	Incremented	Destination address mode selection. This field is locked.
Source Address Mode	Fixed	Source address mode selection. This field is locked.
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection. This field is locked.
Interrupt Frequency	After all transfers have completed	This field is locked.
Destination Pointer	NULL	Destination pointer selection. This field is locked.

Property	Value	Description
Source Pointer	NULL	Source pointer selection. This field is locked.
Number of Transfers	0	Number of transfer selection. This field is locked.
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection. This field is locked.
Activation Source (Must enable IRQ)	Event IIC0 RXI	Activation source selection. This field is locked.
Auto Enable	False	Auto enable selection. This field is locked.
Callback (Only valid with Software start)	NULL	Callback selection. This field is locked.
ELC Software Event Interrupt Priority	Disabled	ELC software event interrupt priority selection.

2）2 番目の I2C フレームワークデバイスを同じスレッドに追加します。sf_i2c 上の I2C フレームワーク共有バスは自動的に追加されません。既存の共有バスを使用するオプションを選択します。コンフィギュレータによって、sf_i2c 上の I2C フレームワーク共有バスと残りのモジュールが自動的に追加されます。最初の I2C フレームワークインスタンスの以前定義した設定と整合するよう、ローレベルモジュールが自動的に追加、構成されます。実際に、スタックでローレベルの設定が変更されると、その設定は他のスタックで自動的に更新されます。

結果のモジュールスタックを次の図に示します。



2 番目のスレッドスタック（スレーブデバイス #2）の構成設定の例は、以下のとおりです。

sf_i2c 上の I2C フレームワークデバイスの構成設定（スレーブ #2）

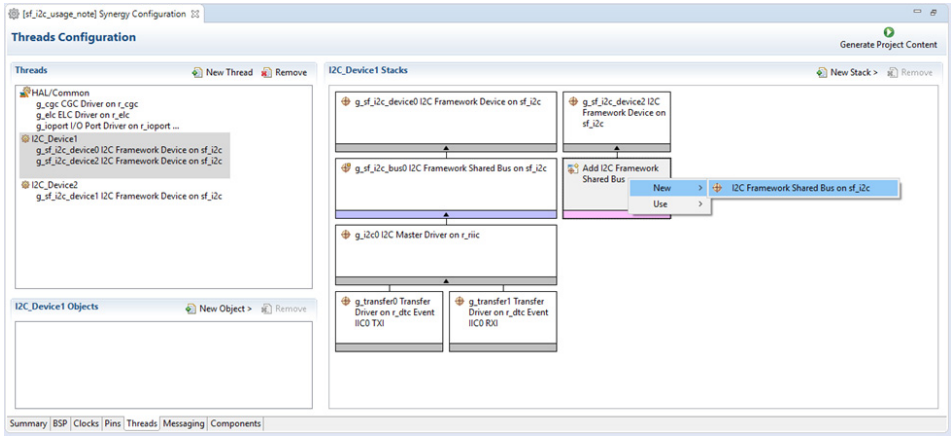
Property	Value	Description
Parameter Checking	Default(BSP)	Enable Or Disable Parameter Checking
Name	g_sf_i2c_device1	Give a name to identify the I2C Framework device. API, Config and Control instances will be created based on this name.
Slave Address	0x28	Specify the address of I2C slave2
Address Mode	7-bit	Select the I2C address mode.

スレーブデバイスが同じローレベル設定を共有している場合は、必要に応じて上記の手順を繰り返し、さらにスレーブデバイスを追加できます。

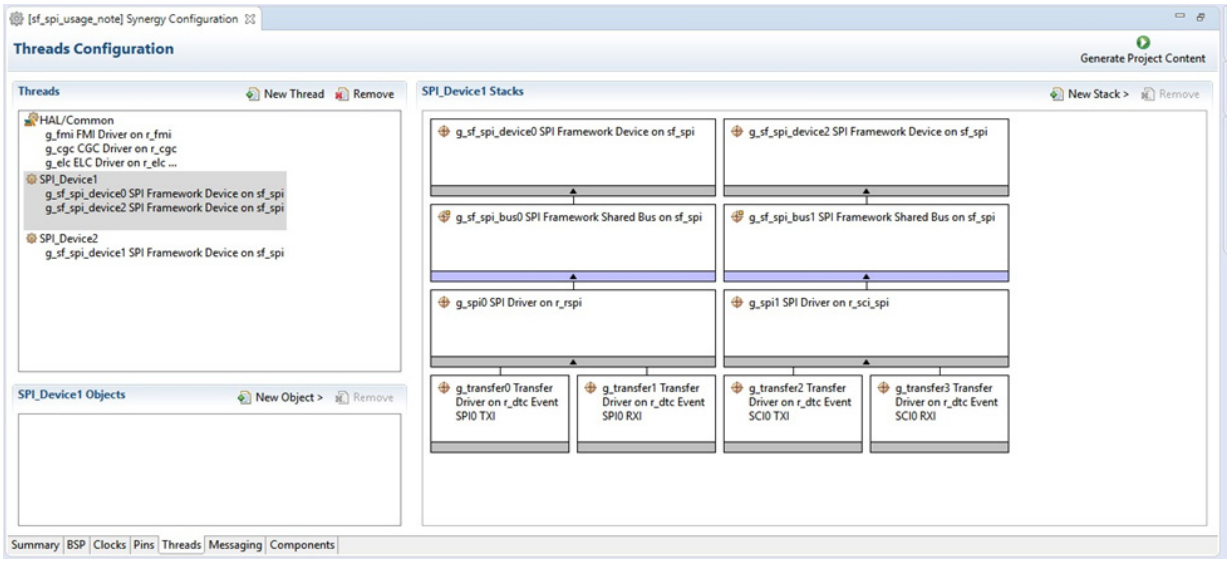
別のセットのスレーブデバイスとはローレベル設定が異なるスレーブデバイスのセットは、異なるバスを使用する必要があり、上記で示した 2 つの手順を繰り返して実装することができます。つまり、このスレーブデバイスのセットには異なるバスと異なるローレベル特性を定義します。

別の共有バスの追加

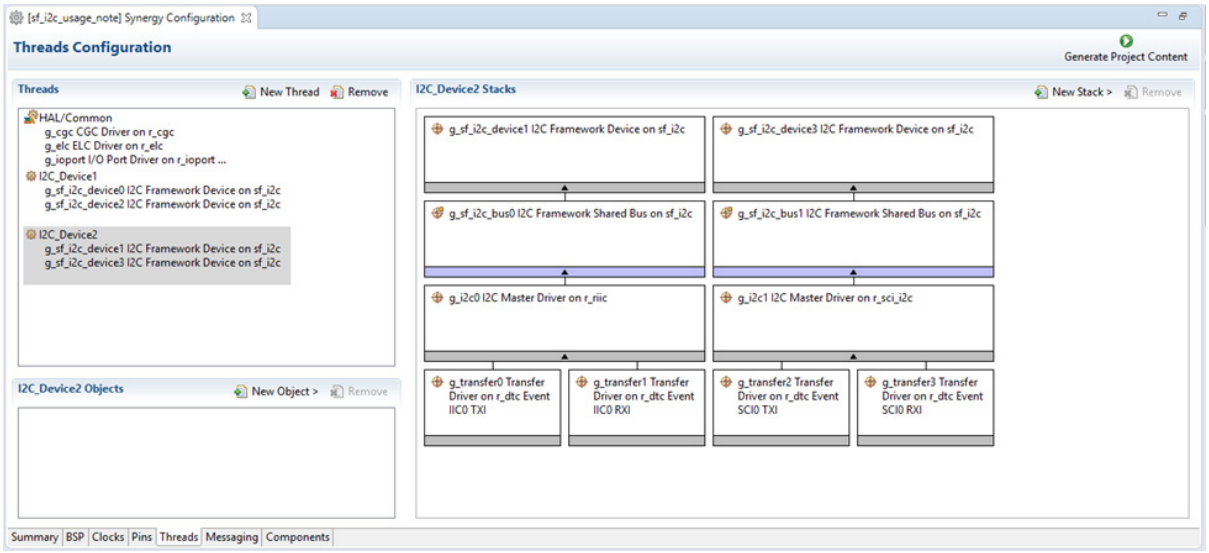
2 番目の共有バスを使用する I2C フレームワークモジュールは、任意のスレッドに追加できます。前の例のモジュールから順に I2C_Device1 スレッドに追加すると、モジュールスタックは以下のように表示されます。共有バスで使用できるオプションは [New] または [Use] です。



2）[New] を選択し、sf_i2c モジュール上の別の I2C フレームワーク共有バスを追加します。必要に応じて、アプリケーションに対して共有バスプロパティを設定します。必要なローレベル I2C ドライバーを選択します。g_i2c1 I2C ドライバーモジュールのチャネル番号は、g_i2c0 I2C ドライバーモジュールのチャネル番号とは異なる必要があります。以下に結果のスレッドスタックを示します。



3）前述の手順と同じ手順を使用して、2 番目のデバイスを I2C_Device2 スレッドに追加できます。以下に結果のスレッドスタックを示します。



アプリケーションで I2C フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_i2c_api_t::open API を使用して I2C フレームワークモジュールを初期化します。各 I2C フレームワークモジュールは、バスに対して操作を実行する前に sf_i2c_api_t::open API を 1 回以上コールする必要があります。
2) sf_i2c_api_t::reset API を使用して I2C MCU ペリフェラルをリセットします（必要な場合）。
3) 特定のフレームワークモジュール用に、sf_i2c_api_t::lock API を使用してバスをロックします。I2C フレームワークモジュールによってバスがロックされると、同じバス上の他の I2C フレームワークモジュールがそのバスを使用することはできません。このことによって、バスの所有権は、この I2C フ

レームワークモジュールがバスをロック解除するまで、このモジュールに残ります。バスがロックされている間、このバス上の他の I2C フレームワークモジュールからの操作は失敗します。バスに対するリード / ライト動作の前に、バスをロックする必要はありません。このことはオプションです。(必要な場合)

4) sf_i2c_api_t::write API を使用して、スレーブにデータを書き込みます。他の I2C フレームワークモジュールによってバスが既にロックされている場合は、ライト動作は失敗します。
5) sf_i2c_api_t::read API を使用して、スレーブからデータを読み取ります。他の I2C フレームワークモジュールによってバスが既にロックされている場合は、リード動作は失敗します。
6) 同じ I2C フレームワークモジュールによってバスが既にロックされている場合は、sf_i2c_api_t::unlock API を使用してバスをロック解除します。バスがロック解除されると、他の I2C フレームワークモジュールがバスを使用できます。保護されたリード動作またはライト動作の終了後は、ロックされたバスをロック解除する必要はありません。(必要な場合)
7) sf_i2c_api_t::close API を使用して I2C フレームワークモジュールを閉じます。各 I2C フレームワークモジュールは、バスに対するすべてのリードおよびライト動作が完了すると、spi_api_t::close API をコールできます。(必要な場合)

これらの一般的な手順を、次の図の通常の動作フローに示します。

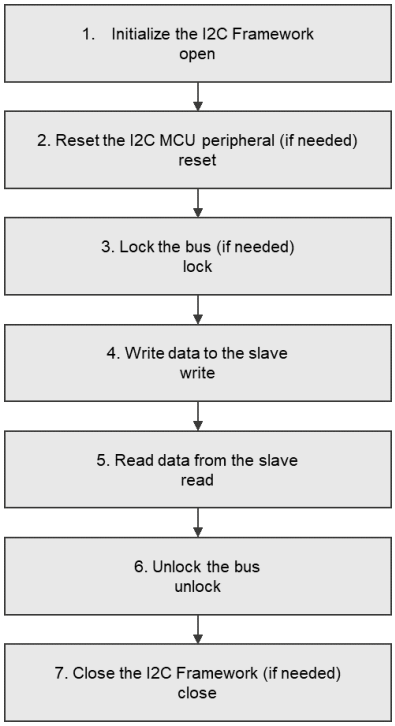


図 90:通常の I2C フレームワークモジュールアプリケーションのフロー図

4.1.24 JPEG デコード フレームワーク

JPEG デコード HAL モジュールは、業界標準の JPEG イメージデコード処理用にハイレベル API を提供し、Synergy ™ JPEG コーデックペリフェラルをサポートします。JPEG デコードフレームワークモジュールは ThreadX® 対応の実装であり、Synergy MCU 上の Synergy JPEG ハードウェアへのスレッドセーフなアクセスを提供します。ユーザー定義のコールバックを作成し、ハードウェアサポートイベントを検出できます。

4.1.24.1 JPEG デコードフレームワークモジュールの特長

- Synergy JPEG ハードウェアへのスレッドセーフなアクセスを提供。
- JPEG デコード HAL モジュールを使用した JPEG 解凍をサポート。
- JPEG　デコーダが完了するまでアプリケーションが待機できるようにするポーリングモードをサポート。
- ユーザーが指定したコールバック関数を使用した割り込みモードをサポート。
- 水平および垂直サブサンプル値、水平ストライド、デコードされたピクセル フォーマット、入力および出力データフォーマット、色空間などのパラメータを設定。
- 画像をデコードする前にそのサイズを取得。
- デコードされた画像フレームの保管のため、コード化されたデータを入力バッファおよび出力バッファに格納する操作をサポート。
- JPEG　デコーダモジュールへのコード化されたデータのストリーム転送をサポート。この機能により、アプリケーションは、ファイルやネットワークからのコード化された JPEG イメージを、イメージ全体をバッファリングすることなく読み取ることができます。
- デコードするイメージ行数を設定。この機能により、アプリケーションは、デコードされたイメージの処理を、フレーム全体をバッファーリングすることなくオンザフライで実行可能。
- 入力値のデコードされたフォーマットとして YCbCr444、YCbCr422、YCbCr420、YCbCr411 をサポート。
- 出力値のデコードされたフォーマットとして ARGB8888 と RGB565 をサポート。
- JPEG 画像のサイズ、高さ、幅が要件を満たさない場合にエラーを返すことが可能。
- JPEG　ハードウェアサポートイベントと同期するためのスレッドをサスペンドまたは再開するための sf_jpeg_decode_api_t::wait API 関数をサポート。

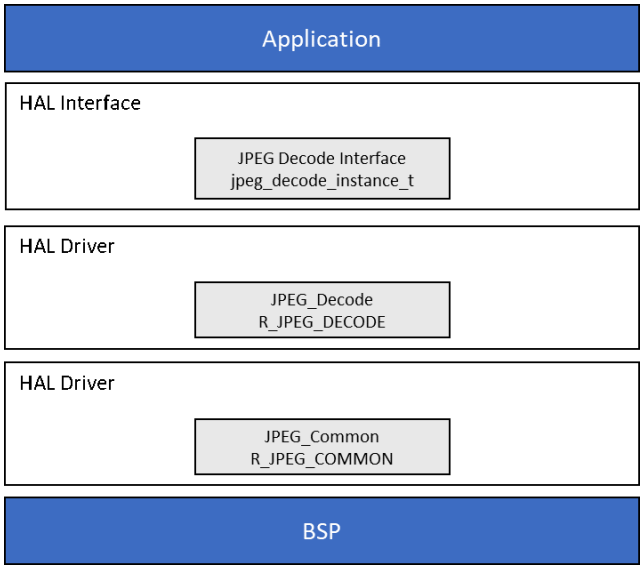


図 91:JPEG デコードフレームワークモジュールのブロック図

4.1.24.2 JPEG デコードフレームワークモジュール API の概要

JPEG デコードフレームワークモジュールは、オープン、クローズ、アラーム設定、RTC 操作の開始と停止の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

JPEG デコードフレームワークモジュール API の要約

Function Name	Example API Call and Description
open	g_sf_jpeg_decode0.p_api->open(g_sf_jpeg_decode0.p_ctrl, g_sf_jpeg_decode0.p_cfg); Open the JPEG Decode Framework.
inputBufferSet	g_sf_jpeg_decode0.p_api->close(g_sf_jpeg_decode0.p_ctrl); Close the JPEG Decode Framework.

Function Name	Example API Call and Description
outputBufferSet	g_sf_jpeg_decode0.p_api->outputBufferSet(g_sf_jpeg_decode0.p_ctrl, p_buffer, buffer_size); Assign output buffer to JPEG codec for storing output data.
linesDecodedGet	g_sf_jpeg_decode0.p_api->linesDecodedGet g_sf_jpeg_decode0.p_ctrl, p_lines); Return the number of lines decoded into the output buffer.
horizontalStrideSet	g_sf_jpeg_decode0.p_api->horizontalStrideSet(g_sf_jpeg_decode0.p_ctrl, stride); Configure the horizontal stride value.
imageSubsampleSet	g_sf_jpeg_decode0.p_api->imageSubsampleSet(g_sf_jpeg_decode0.p_ctrl, horizontal, vertical); Configure the horizontal and vertical subsample settings.
wait	g_sf_jpeg_decode0.p_api->wait(g_sf_jpeg_decode0.p_ctrl, p_status, timeout); Wait for the current JPEG codec operation to finish with a timeout value given in ThreadX ticks.
statusGet	g_sf_jpeg_decode0.p_api->statusGet(g_sf_jpeg_decode0.p_ctrl, p_status); Retrieve current status of the JPEG codec module.

Function Name	Example API Call and Description
imageSizeGet	g_sf_jpeg_decode0.p_api->imageSizeGet(g_sf_jpeg_decode0.p_ctrl, p_horizontal, p_vertical); Retrieve image size during decoding operation.
pixelFormatGet	g_sf_jpeg_decode0.p_api->pixelFormatGet(g_sf_jpeg_decode0.p_ctrl, p_color_space); Get the input pixel format.
close	g_sf_jpeg_decode0.p_api->close(g_sf_jpeg_decode0.p_ctrl); Cancel an outstanding operation.
versionGet	g_sf_jpeg_decode0.p_api->versionGet(&version); Get version and store it in provided pointer p_version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	JPEG Decode driver is successfully opened.
SSP_ERR_ASSERTION	Assertion error.
SSP_ERR_IN_USE	Module already in use.
SSP_ERR_TIMEOUT	The wait operation times out, the underlying driver did not respond in time.
SSP_ERR_WAIT_ABORTED	System internal error occurred.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.24.3 JPEG デコードフレームワークモジュールの動作の概要

JPEG デコードフレームワークモジュールは、標準 JPEG デコード操作を実装します。入力バッファのデータを取得し、定義済みの JPEG デコードアルゴリズムをバッファに適用します。出力は、定義済みの出力バッファの場所に配信されます。JPEG ハードウェアサポートイベントと同期するためのスレッドのサスペンドや再開には wait API 関数を使用できます。

JPEG デコードフレームワークモジュールの動作に関する重要な注意事項と制限事項

- JPEG エンコードされたデータのデコードは、sf_jpeg_decode_api_t::open API をコールすることで開始できます。モジュールを開くには、JPEG デコードフレームワークモジュールインスタンスを使用します。このインスタンスには、API 関数のポインタ、制御ブロックへのポインタ、ISDE の e^2 studio の Synergy プロジェクトコンフィギュレータを通じて生成される静的構成が含まれています。
- JPEG デコードフレームワークモジュールを停止するには、sf_jpeg_decode_api_t::close API をコールします。
- 入力バッファストリーミングモードは、入力中心の機能が必要な場合に使用できます。
- 出力バッファストリーミングモードは、出力中心の機能が必要な場合に使用できます。
- 出力データのカラーフォーマットとして RGB565 と ARGBB888 をサポートします。
- JPEG デコードフレームワークモジュールの制御ブロックにはステータスフラグがあり、sf_jpeg_decode_api_t::statusGet API を通じてモジュールの現在のステータスが示されます。また、モジュールで特定のイベントが発生すると、ユーザーコールバック関数を通じてステータスがレポートされます。
- JPEG デコードフレームワークモジュールは、入力バッファがソースイメージファイルよりも小さい場合に入力バッファのバッファストリーミングモードをサポートします。ハードウェアで生成された INPUT_PAUSE 割り込みが発生するたびに、次の入力フレームが入力バッファとして設定されます。
- JPEG デコードフレームワークモジュールは、結果のイメージが出力バッファサイズよりも大きい場合に出力バッファのバッファストリーミングモードをサポートします。ハードウェアで生成された OUTPUT_PAUSE 割り込みが発生するたびに、次のデータ用のスペースを空けるために出力バッファからデータが読み取られ、格納されます。
- JPEG デコードフレームワークモジュールが正常に動作するためには、入力および出力バッファが 8 バイトでアラインされている必要があります。そうでないと、API 関数は実行に失敗したことを示すエラーコードを返します。
- JPEG デコードフレームワークモジュールは、JPEG エンコード処理をサポートしていません。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.24.4 アプリケーションへの JPEG デコードフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに JPEG デコードフレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

JPEG デコードフレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(JPEG デコードフレームワークのデフォルト名は g_sf_jpeg_decode0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

JPEG デコードフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_jpeg_decode0 JPEG Framework	Threads	New Stack> Framework> Graphics> JPEG Decode Framework on sf_jpeg_decode

次の図に示すように、sf_jpeg_decode 上の JPEG デコードフレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

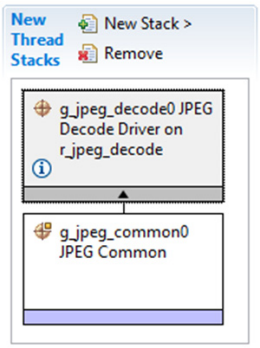


図 92:JPEG デコードフレームワークモジュールのスタック

解凍プロセス割り込み (JEDI)

JPEG 解凍プロセス割り込みは、次のイベントが検出されたときに発生します。

- 現在の解凍プロセスが正常に完了した。
- 解凍プロセス中にエラーが発生した。
- イメージのサイズとピクセル フォーマットが正常に読み出された。

データ転送インタフェース (JDTI)

JPEG データ転送割り込みは、次のイベントが検出されたときに発生します。

- JPEG コードされたデータがすべて正常に完了した。
- sf_jpeg_decode_api_t::linesDecodedGet で指定された出力イメージデータ行数が転送された。

sf_jpeg_decode_api_t::inputBufferSet で指定された入力イメージデータ行数が転送された。

4.1.24.5 JPEG デコードフレームワークモジュールの構成

ユーザーは必要な動作に合わせて JPEG デコードフレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_jpeg_decode での JPEG デコードフレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_jpeg_decode0	The name to be used for a JPEG Decode Framework module instance.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

JPEG デコードフレームワークモジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_jpeg の JPEG HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_jpeg_decode0	The name to be used for a JPEG Decode module instance.
Byte Order for Input Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2) Default: Normal Byte order	Specify the byte order for input data. The order is swapped as specified in every 8-byte.
Byte Order for Output Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2) Default: Normal Byte order	Specify the byte order for output data. The order is swapped as specified in every 8-byte.

ISDE Property	Value	Description
Output Data Color Format	Pixel Data RGB565 format, Pixel Data ARGBB888 format Default: Pixel Data RGB565 format	Specify the output data format.
Alpha value to be applied to decoded pixel data (only valid for ARGB8888 format)	255	Specify the alpha value for the output data format (only valid for ARGB8888 format).
Name of user callback function	NULL	Specify the name of user callback function.
Decompression Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Decompression interrupt priority selection.
Data Transfer Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Data transfer interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

JPEG 共通モジュールの構成設定

ISDE Property	Value	Description
Name	g_jpeg_common0	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

JPEG デコードフレームワークモジュールのクロック構成

JPEG フレームワークモジュールは、周辺モジュールクロック A（PCLKA）を使用して内部論理を実行します。

JPEG デコードフレームワークモジュールの割り込み構成

割り込みを有効化するには、ISDE の JPEG デコードフレームワークモジュールの [Properties] ウィンドウで解凍割り込みとデータ転送割り込みのプライオリティを設定します。

JPEG デコードフレームワークモジュールのピン構成

JPEG デコードフレームワークモジュールはピンを使用しません。

4.1.24.6 アプリケーションでの JPEG デコードフレームワークモジュールの使用

一般的なアプリケーションで JPEG デコードフレームワークモジュールを使用する際の手順は次のとおりです。

1) sf_jpeg_decode_api_t::open API を使用して JPEG デコードペリフェラルを初期化します。
2) sf_jpeg_decode_api_t::imageSubsampleSet API を使用してイメージサブサンプルを設定します。
3) sf_jpeg_decode_api_t::horizontalStrideSet API を使用して水平ストライドを設定します。
4) sf_jpeg_decode_api_t::outputBufferSet API を使用して出力バッファを設定します。
5) sf_jpeg_decode_api_t::inputBufferSet API を使用して入力バッファを設定します。
6) sf_jpeg_decode_api_t::wait API を使用してデコードの完了を待ちます。
7) sf_jpeg_decode_api_t::statusGet API を使用してデコードステータスを確認します。
8) sf_jpeg_decode_api_t::close API を使用して、インスタンスを閉じます（必要な場合）。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

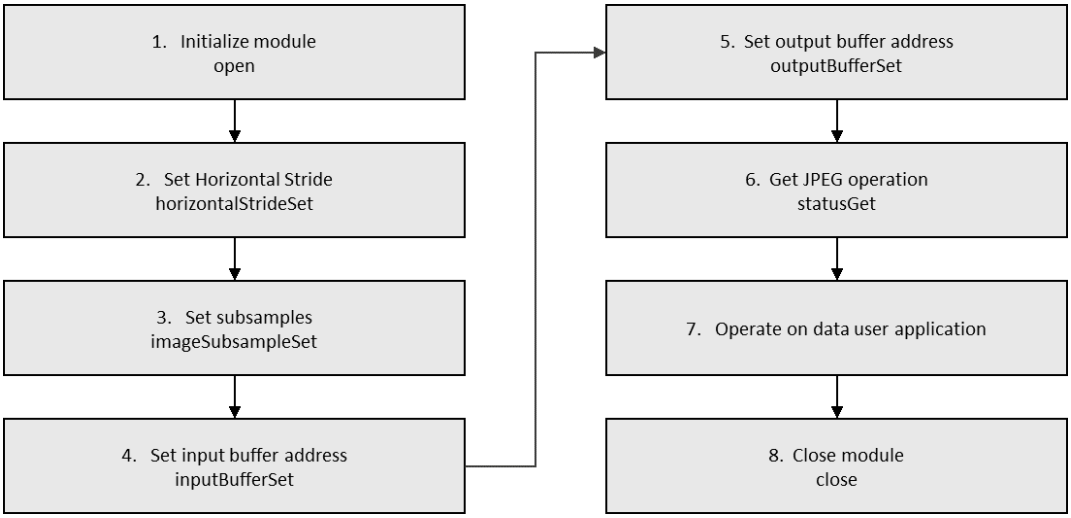


図 93:通常の JPEG デコードフレームワークモジュールアプリケーションのフロー図

4.1.25 sf_memory_qspi_nor 上のメモリフレームワーク

sf_memory_qspi_nor フレームワークモジュール上のメモリフレームワークは、QSPI NOR メモリデバイスとのインタフェースを可能にするためにハイレベル API を提供します。QSPI NOR フラッシュメモリに対するデータのリード、ライト、消去のための API 関数を提供します。sf_memory_qspi_nor モジュール上のメモリフレームワークは、LevelX

ウェアレベリングのサポートが必要な場合に、sf_el_lx_nor フレームワークモジュール上のハイレベルの Port LevelX フレームワークでも使用されます。

4.1.25.1 メモリフレームワークモジュールの特長

- QSPI NOR フラッシュメモリデバイス用のメモリインタフェースをサポート。
- QSPI NOR フラッシュメモリデバイスでの I/O 操作をサポート。
  - 読み取り
  - 書き込み
  - 消去

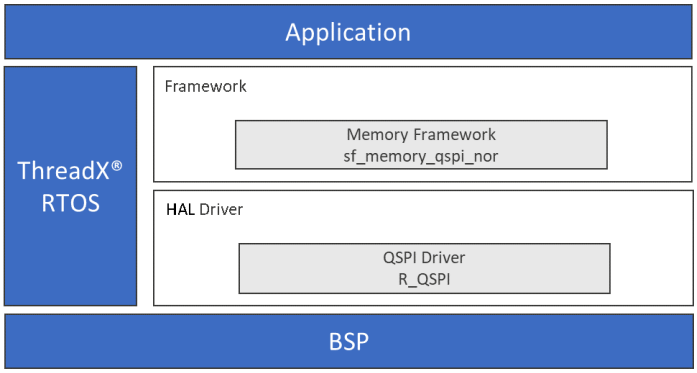


図 94: メモリフレームワークモジュールのブロック図

4.1.25.2 メモリフレームワークモジュールの API の概要

メモリフレームワークモジュールは、QSPI NOR フラッシュメモリデバイスのリード、ライト、消去を行うための API 関数を定義します。次の表には、使用可能なすべての API 関数のリスト、API コールの例、各 API の簡単な説明が記載されています。ステータス戻り値の表は API 要約表の後にあります。

メモリフレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_memory_qspi_nor0.p_api->open(g_sf_memory_qspi_nor0.p_ctrl, g_sf_memory_qspi_nor0.p_cfg); Open the SF Memory QSPI NOR driver module for the purposes of reading and writing flash memory.

Function Name	Example API Call and Description
close	g_sf_memory_qspi_nor0.p_api->close(g_sf_memory_qspi_nor0.p_ctrl); Close the Memory QSPI NOR driver module.
read	g_sf_memory_qspi_nor0.p_api->read(g_sf_memory_qspi_nor0.p_ctrl, p_dest_address, memory_address, num_bytes); Read specified number of bytes of data from a particular address on the QSPI flash device.
write	g_sf_memory_qspi_nor0.p_api->write(g_sf_memory_qspi_nor0.p_ctrl, p_src_address, memory_address, num_bytes); Program data to the flash.
flush	g_sf_memory_qspi_nor0.p_api->flush(g_sf_memory_qspi_nor0.p_ctrl); Flush any pending data to the disk. This is not required for QSPI NOR flash.
erase	g_sf_memory_qspi_nor0.p_api->erase(g_sf_memory_qspi_nor0.p_ctrl, memory_address, num_bytes); Erase a number of bytes from the flash.
infoGet	g_sf_memory_qspi_nor0.p_api->infoGet(g_sf_memory_qspi_nor0.p_ctrl, p_info); Returns the information about the flash.
versionGet	g_sf_memory_qspi_nor0.p_api->versionGet(&version); Get the driver version based on compile time macros.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Configuration was successful.
SSP_ERR_ASSERTION	The parameter p_ctrl or p_cfg is NULL.
SSP_ERR_ALREADY_OPEN	Driver is already open.
SSP_ERR_NOT_OPEN	Driver is not opened.
SSP_ERR_INVALID_ARGUMENT	Number of bytes requested is invalid.
SSP_ERR_TIMEOUT	Wait timed out.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.25.3 メモリフレームワークモジュールの動作の概要

メモリフレームワークモジュールは、QSPI メモリ用のメモリデータ転送をサポートします。モジュールのオープン、リード、ライト、消去に、ハイレベルの API 関数が使用可能です。LevelX のサポートが必要な場合に、このモジュールと組み合わせて、sf_el_lx_nor フレームワークモジュール上のハイレベルの Port LevelX フレームワークを使用できます。

メモリフレームワークモジュールは、他の SSP メディアモジュールに共通の標準インタフェースを使用します。たとえば、SDMMC、SPI フラッシュ、および SDRAM/RAM メモリをサポートするモジュールは同じ API コールを使用するため、プログラミングインタフェースは、すべてのメディアドライバーで同じになります。これらのモジュールは相互に簡単に交換できます。デバイス適応ドライバー（r_qspi, など）にはメモリフレームワークインタフェースを使用してアクセスします。これらのドライバーは、メディア I/O 操作の実行に必要なデバイス固有のコードを提供します。メモリインタフェース関数コールによって渡される構成構造体と制御構造体も、通常はデバイス固有です。

メモリフレームワークモジュールの動作に関する重要な注意事項と制限事項

I/O 操作に sf_memory_qspi_nor を使用する前に、メディアを消去する必要があります。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.25.4 アプリケーションへのメモリフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにメモリフレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユー

ザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

メモリフレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(メモリフレームワークのデフォルト名は g_sf_memory_qspi_nor0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

メモリフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_memory_qspi_nor0 Memory Framework on sf_memory_qspi_nor	Threads	New Stack> Framework> Memory> Memory Framework on sf_memory_qspi_nor

次の図に示すように、sf_memory_qspi_nor 上のメモリフレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

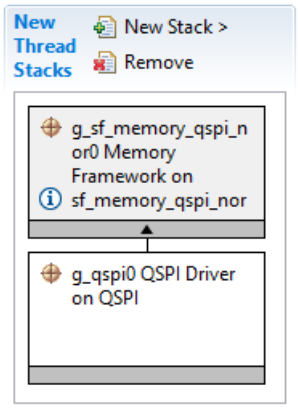


図 95: メモリフレームワークモジュールのスタック

4.1.25.5 メモリフレームワークモジュールの構成

ユーザーは必要な動作に合わせてメモリフレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略

化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_memory_qspi_nor でのメモリフレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_sf_memory_qspi_nor0	Module name.
Write of Erase Timeout (in ticks)	30000	Timeout ticks for waiting on write or erase to complete.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

メモリフレームワークのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_qspi での QSPI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_qspi0	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

メモリフレームワークモジュールのクロック構成

メモリフレームワークモジュールは、PCLKA をクロックソースとして使用する QSPI ペリフェラルを使用します。

実行時にクロック周波数を変更するには、CGC インタフェースを使用します。

メモリフレームワークモジュールのピン構成

メモリフレームワークモジュールを使用するには、必要に応じて QSPI ペリフェラルのポートピンを設定する必要があります。次の表では、ISDE [Configuration] ウィンドウでのピンの選択方法を示します。

sf_memory_qspi_nor 上のメモリフレームワークモジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
QSPI	Pins	Select Peripherals > Storage:QSPI QSPI0

4.1.25.6 アプリケーションでのメモリフレームワークモジュールの使用

アプリケーションでメモリフレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_memory_api_t::open API 関数を使用してインスタンスを初期化します。
2) sf_memory_api_t::write API 関数を使用して、QSPI フラッシュにデータを書き込みます。
3) sf_memory_api_t::read API 関数を使用して、QSPI フラッシュからデータを読み取ります。
4) sf_memory_api_t::erase API 関数を使用して、QSPI フラッシュからデータを消去します。
5) sf_memory_api_t::infoGet API 関数を使用して、QSPI フラッシュ情報を読み取ります。
6) sf_memory_api_t::close API 関数を使用してインスタンスを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

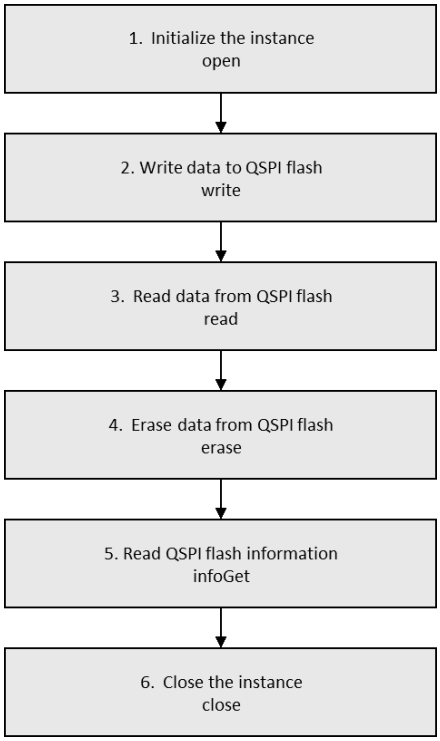


図 96:通常のメモリフレームワークモジュールアプリケーションのフロー図

4.1.26 メッセージング フレームワーク

メッセージフレームワークは、スレッド間のメッセージ送受信に使用する、イベント駆動型の API を実装します。メッセージフレームワークモジュールにより、アプリケーションは 2 つ以上のスレッド間でメッセージを送受信できるようになります。このフレームワークは、ThreadX のメッセージキュープリミティブを使用してメッセージを送受信し、ThreadX RTOS メッセージキューサービス単独の場合よりも多くの利点が得られます。メッセージフレームワーク API は純粋なソフトウェア API であり、ハードウェア周辺機能にはアクセスしません。メッセージフレームワークコールバックを使用すると、イベント作成元スレッドとメッセージサブスクライバスレッドは、メッセージの送受信後にハンドシェイクを行うことができます。

[Messaging] タブを使用して、メッセージフレームワークモジュール用のカスタムイベントクラスやイベント、サブスクライバを作成したり、タッチパネルフレームワークモジュールによって使用されるタッチイベントなどの事前設定されたイベントをカスタマイズしたりできます。

4.1.26.1 メッセージフレームワークモジュールの特長

メッセージフレームワークモジュールは以下の機能をサポートしています。

- スレッド間通信 - 異なるデバイスを制御したりサブシステムを管理したりするアプリケーションスレッドが互いに通信できます。
- パブリッシュ / サブスクライブ方式 - フレームワークの設計は、疎結合型のメッセージング方式に基づいています。複数のスレッドが 1 つのイベントクラスを受信待ちできる設計になっています。メッセー

ジ作成元スレッドは、イベントクラスのメッセージにサブスクライブしているスレッドを知る必要がありません。サブスクライバーは、メッセージの作成元を知る必要がありません。

- メッセージ管理 - フレームワークでは、バッファを制御するフラグやハンドシェイクのためのコールバック関数ポインタなど、各メッセージを管理するためのバッファ制御ブロックがサポートされています。
- メッセージバッファリング - フレームワークは、メッセージング用のバッファの割り当てと解放を管理します。アプリケーションは、割り当てられたバッファを利用してメッセージを書き込んだり、不要になったメッセージを破棄することができます。
- 同期通信 - フレームワークでは、ThreadX メッセージキューを使用した非同期メッセージングがサポートされていますが、メッセージ作成元とサブスクライバスレッドの間のハンドシェイクを確立するためのオプションも提供されています。ハンドシェイクは、サブスクライバスレッドから作成元スレッドのユーザーコールバック関数を呼び出すことで実装されています。
- メッセージ構造の指定 - 定義済みの共通メッセージヘッダが提供されています。また、いくつかの典型的なペイロード構造体テンプレートも例として提供されています。
- メッセージのプライオリティ - 高プライオリティのメッセージを送信して、サブスクライバスレッドが、メッセージキューにある他のメッセージの前にそのメッセージを受け取れるようにすることができます。

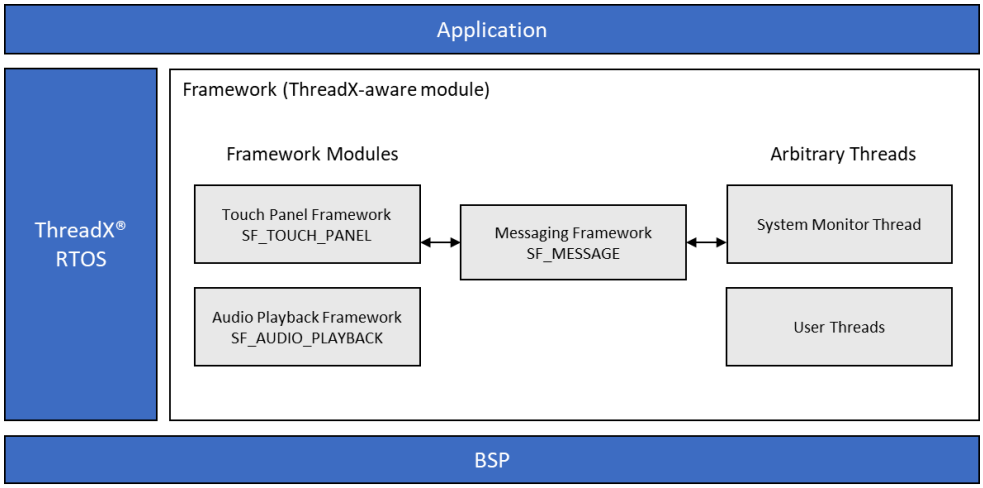


図 97: メッセージフレームワークモジュールのブロック図

4.1.26.2 メッセージフレームワークモジュール API の概要

メッセージフレームワークモジュールは、フレームワークのオープンとクローズ、バッファの取得と解放、サブスクライバへのメッセージのポスト用の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

メッセージフレームワークモジュール API の要約

Function Name	Example API Call and Description
open	g_sf_message.p_api->open (g_sf_message.p_ctrl, g_sf_message.p_cfg); Initialize message framework. Initiate the messaging framework control block, configure the work memory corresponding to the configuration parameters.
close	g_sf_message.p_api->close (g_sf_message.p_ctrl); Finalize message framework.
bufferAcquire	g_sf_message.p_api->bufferAcquire (g_sf_message.p_ctrl, &p_buffer, &acquire_cfg, wait_option); Acquire buffer for message passing from the block.
bufferRelease	g_sf_message.p_api->bufferRelease (g_sf_message.p_ctrl, &p_buffer, option); Release buffer obtained from SF_MESSAGE_BufferAcquire.
post	g_sf_message.p_api->post (g_sf_message.p_ctrl,(*) p_payload, &post_cfg, &err_post, wait_option); Post message to the subscribers.
pend	g_sf_message.p_api->pend (g_sf_message.p_ctrl,&my_queue, &p_buffer, &p_header, wait_option); Pend message.

Function Name	Example API Call and Description
versionGet	g_sf_message.p_api->versionGet (&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API call successful.
SSP_ERR_ASSERTION	Required pointer is NULL.
SSP_ERR_BUFFER_RELEASED	The buffer is released.
SSP_ERR_ILLEGAL_SUBSCRIBER_LISTS	Message subscriber lists is illegal.
SSP_ERR_IN_USE	The messaging framework is in use.
SSP_ERR_INTERNAL	OS service call fails.
SSP_ERR_INVALID_MSG_BUFFER_SIZE	Message buffer size is invalid.
SSP_ERR_INVALID_WORKBUFFER_SIZE	Invalid work buffer size.
SSP_ERR_MESSAGE_QUEUE_EMPTY	Queue is empty. (Timeout occurs before receiving a message if timeout option is specified.)
SSP_ERR_MESSAGE_QUEUE_FULL	Queue is full. (Timeout occurs before sending a message if timeout option is specified.)
SSP_ERR_NO_MORE_BUFFER	No more buffer found in the memory block pool.
SSP_ERR_NO_SUBSCRIBER_FOUND	No subscriber found.
SSP_ERR_NOT_OPEN	Message framework module has yet to be opened.

Name	Description
SSP_ERR_TIMEOUT	OS service call returns timeout.
SSP_ERR_TOO_MANY_BUFFERS	Too many message buffers.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.26.3 メッセージフレームワークモジュールの動作の概要

下図に、メッセージフレームワークモジュールを利用するシステム内の、メッセージ作成元スレッドとサブスクライバ thread(s) の間のメッセージングデータフローの概要を示します。

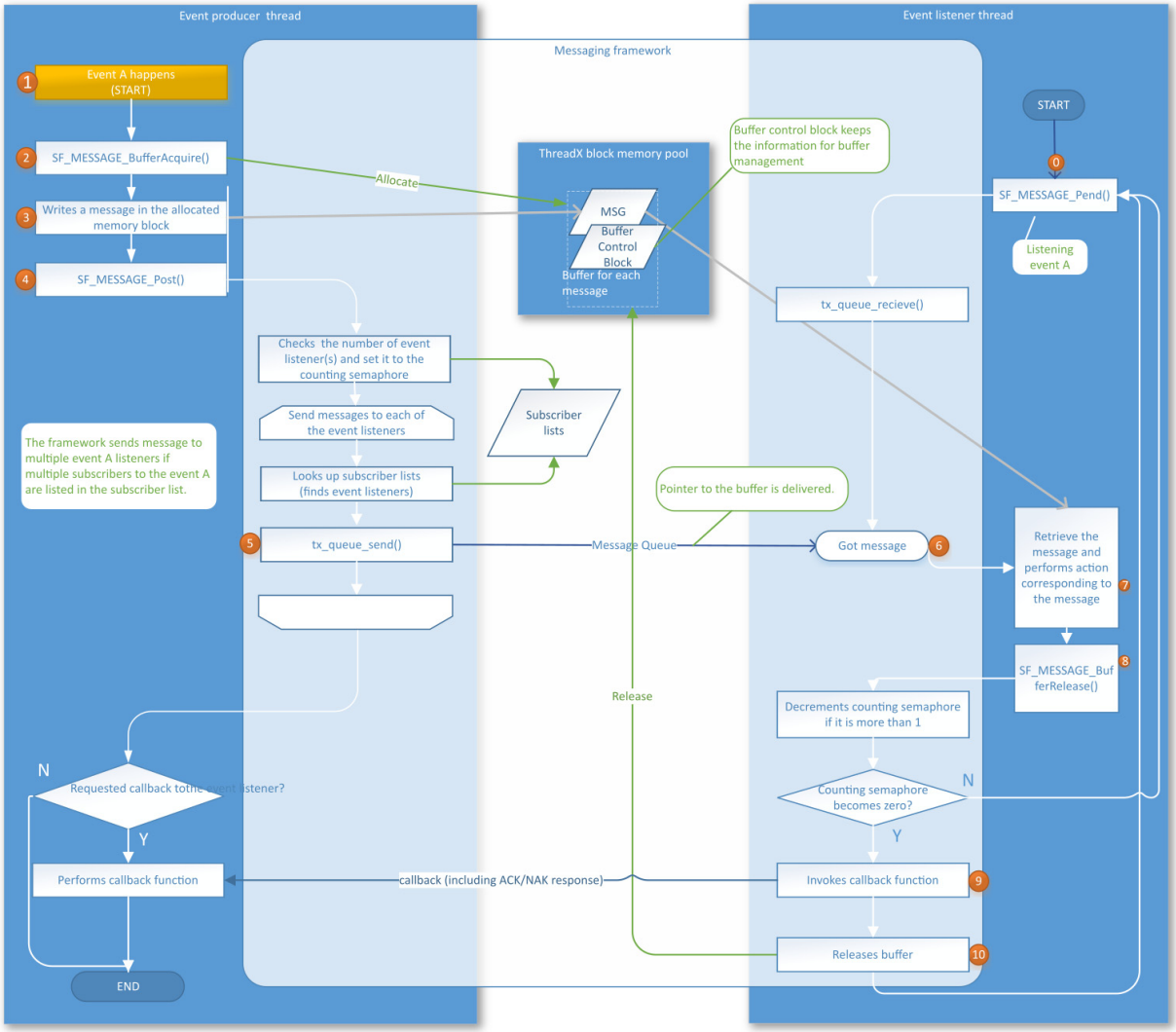


図 98: メッセージフレームワークモジュールのデータフロー

以下で、メッセージ送受信手順の各ステージについて説明します。

注 : システムのスレッドが sf_message_api_t::open API を使用してコールされており、さらにメッセージサブスクライバスレッドが sf_message_api_t::pend API をコールして、イベントクラスのメッセージを待っています。

1) イベント (イベント A) がメッセージ作成元スレッドで発生します。
2) メッセージ作成元スレッドは、sf_message_api_t::bufferAcquire をコールして、メッセージフレームワークモジュールが管理する ThreadX メモリプールからバッファを取得します。
   sf_message_api_t::bufferAcquire は割り当てられたバッファのアドレスを返します。
3) メッセージ作成元が、割り当てられたバッファにメッセージを書き込みます。

4) メッセージ作成元が、sf_message_api_t::post を呼び出してメッセージをポストします。
5) メッセージフレームワークモジュールは、イベントサブスクライバリストを検索し、ThreadX メッセージキュープリミティブを使用して、メッセージサブスクライバスレッドのメッセージキューにメッセージを送信します。フレームワークは、バッファのポインタのみを送信し、メッセージ全体を送信せず、メッセージ送受信を行います。
6) メッセージがメッセージサブスクライバスレッドのメッセージキューに到達し、メッセージサブスクライバスレッドが sf_message_api_t::pend から返されます。API 関数は、メッセージが格納されているバッファアドレスをメッセージサブスクライバスレッドに返します。
7) メッセージ サブスクライバー スレッドは、メッセージを受信し、イベントに応じたアクションを実行します。
8) メッセージ サブスクライバー スレッドは、sf_message_api_t::bufferRelease を呼び出して、メッセージに割り当てられたバッファを解放しようとします。そのメッセージサブスクライバスレッドが、メッセージにサブスクライブしている最後のスレッドでない場合、フレームワークはバッファを解放しません。なぜなら、すべてのサブスクライバがメッセージを受信するまでメッセージをバッファ内に保持する必要があるためです。
9) メッセージフレームワークモジュールは、そのメッセージサブスクライバスレッドが、メッセージサブスクライバグループの最後のスレッドである場合、イベント作成元スレッドによって指定されたユーザーコールバック関数を呼び出します。
10) グループのすべてのサブスクライバがメッセージを使用済みであることがフレームワークに通知されると、メッセージフレームワークモジュールはバッファを解放します。
(sf_message_api_t::bufferRelease API 関数のリリースオプション SF_MESSAGE_RELEASE_OPTION_FORCED_RELEASE は複数サブスクライバのシナリオでは使用しないでください。)

メッセージフレームワークモジュールのメッセージ作成元とサブスクライバ

メッセージ フレームワーク モジュールは、パブリッシュ / サブスクライブ モデルに基づくスレッド間メッセージング システムです。メッセージは、イベント作成元スレッドによって、イベントクラスコードとともにポストされます。メッセージサブスクライバスレッドは、イベントクラスにサブスクライブする保留中のメッセージを確認できます。サブスクライバは、フレームワークによって参照されるサブスクライバリストに登録されます。フレームワークは、サブスクライバリストを使用することで、メッセージを複数のサブスクライバに配信できます。

メッセージング フレームワーク モジュール システム ネットワークに参加するすべてのスレッドと、ネットワーク内のすべてのスレッドがメッセージを受信待ちできます。

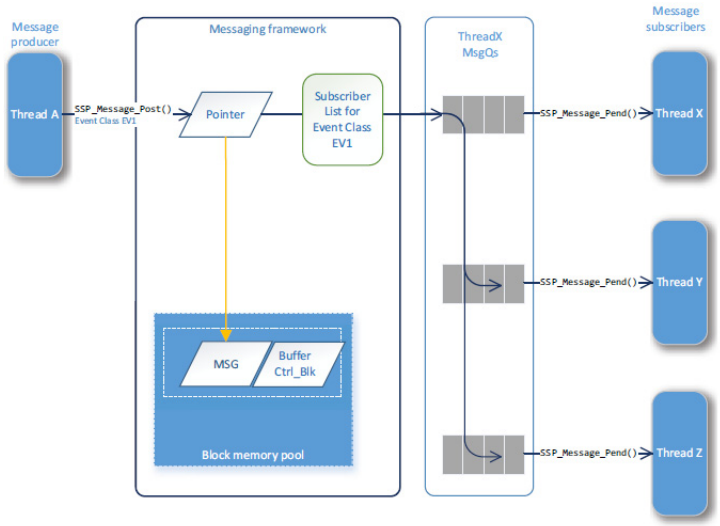


図 99: メッセージフレームワークモジュールのサブスクライブ

メッセージフレームワークモジュールのイベント、サブスクライバ、メッセージ

メッセージフレームワークモジュールのイベントクラスコード

イベントクラスコードは、メッセージモジュールの最も重要な定義です。メッセージモジュールは、メッセージ作成元をサブスクライバに接続するメカニズムとしてイベントクラスコードを使用します。イベントクラスコードは、アプリケーションで発生するイベントのクラス定義です。イベントクラスの分類はユーザー定義に依存しますが、サブシステム内で発生する可能性がある特定のイベントのグループ名であることが想定されています。たとえば、以下のイベント クラスを使用できます。

- タッチ サブシステムの一部である「touch」イベント クラス。Touch イベントクラスは、イベントクラスのウィンドウに自動的にロードされます。このウィンドウは、Synergy プロジェクトにタッチパネルフレームワークを追加するときにプロジェクトコンフィギュレータの [Messaging] タブで使用できます。
- "time" イベントクラスは、時間関連のアプリケーションを管理するサブシステムの一部です。

イベントクラスコードは、sf_message_event_class_t 列挙で定義され、プレフィックス SF_MESSAGE_EVENT_CLASS_XXX. を持ちます。イベントクラスコードの定義はシステムごとに異なりますが、フレームワークでは具体的なイベントクラスコードが用意されておらず、代わりに一連のイベントクラスコードが例として提供されています。(「メッセージフレームワークモジュールの構成」セクションを参照してください。) イベントクラスの最大数は 255 です。

アプリケーションは、イベントクラスコードを以下のように使用できます。

- メッセージ作成元スレッドは、イベントクラスコードを sf_message_header_t 型の共通メッセージヘッダの event_b.class_code ビットフィールドに設定してからメッセージをポストします。
- メッセージサブスクライバスレッドは、メッセージを受信した後、メッセージヘッダに設定されているイベントクラスコードに従ってイベント処理を分岐します。
- イベントクラスコードのサブスクライバは、メッセージフレームワークがメッセージをサブスクライバに配信できるように、グループ化されてサブスクライバリストに登録されている必要があります。

次の図は、ISDE を使用してイベントクラスを構成する方法を示しています。

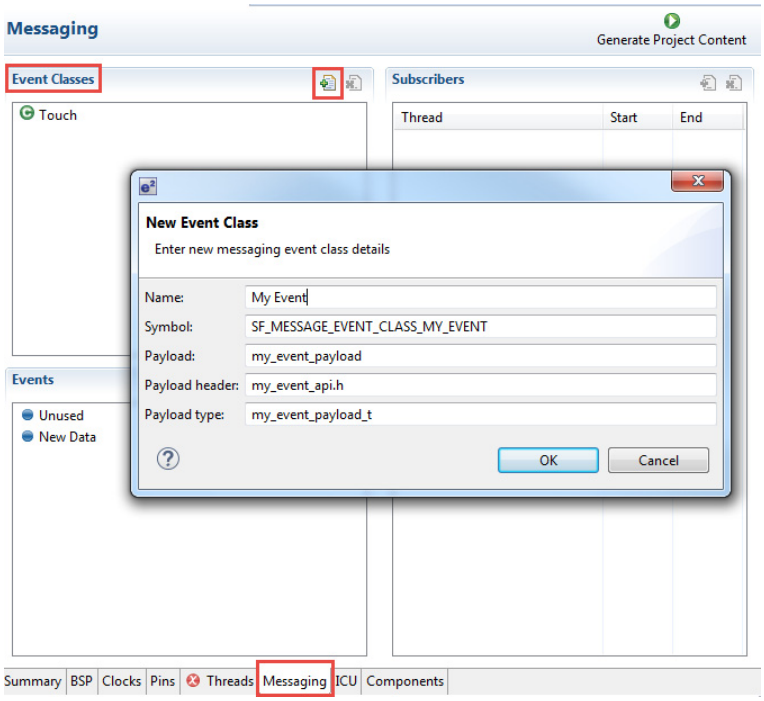


図 100:メッセージフレームワークモジュールの ISDE イベントクラスの構成

メッセージフレームワークモジュールのイベントクラスインスタンス番号

イベントクラスインスタンス番号は、アプリケーションが異なるイベントクラスインスタンスを必要とするときに使用されます。たとえば、オーディオストリーミングイベントクラスでは、ストリーミングチャネル N を示すインスタンス *N* を使用できます。メッセージサブスクライバは、メッセージの共通ヘッダにあるイベントクラスインスタンス番号が、所有する番号と一致する場合にのみメッセージを受信できます。

言い換えれば、イベントクラスインスタンス番号がサブスクライバの範囲外であるメッセージはフィルタ処理で除外され、イベントクラスサブスクライバであっても、そのサブスクライバには配信されません。イベントクラスインスタンス番号の最大値は 255 です。

注 : タッチパネルフレームワークには通常、1 つしかインスタンスがないため、イベントクラスインスタンス番号は 0 となり、サブスクライバリストの開始値と終了値が 0 に設定されます。

アプリケーションは、イベントクラスインスタンス番号を以下のように使用できます。

- メッセージ作成元スレッドは、イベントクラスインスタンス番号を sf_message_header_t 型の共通メッセージヘッダの event_b.class_instance ビットフィールドに設定してからメッセージをポストします。
- サブスクライバリスト内の各サブスクライバインスタンスは、メッセージを受信するにはイベントクラスインスタンス番号の範囲（sf_message_subscriber_t::instance_range.start および sf_message_subscriber_t::instance_range.end）. を指定する必要があります。
- イベントクラスに複数インスタンスが必要でない場合は、サブスクライバリストのサブスクライバインスタンスで sf_message_header_t::event_b.class_instance, sf_message_subscriber_t::instance_range.start, sf_message_subscriber_t::instance_range.end にゼロを指定します。

メッセージフレームワークモジュールのイベントコード

イベントコードには、イベント定義の詳細が含まれています。たとえば、オーディオ再生イベントクラスのイベントコードは、" 再生開始 " と " 再生停止 " です。もう 1 つの例は、"time" イベントクラスの "set" または "get" です。イベントクラスコードは sf_message_event_t で列挙され、プレフィックス SF_MESSAGE_EVENT_XXX. を持ちます。イベントコードの定義は、ユーザーコードとイベントクラスコードに依存します。フレームワークには、いくつかのコードが例として用意されています。イベントコードの構成については、「イベントクラスコードとイベントコードの構成」を参照してください。イベントクラスインスタンス番号の最大値は 65535 です。

タッチパネルフレームワークは、イベントコードとしてのみ新しいデータを使用します。

アプリケーションは、イベントコードを以下のように使用できます。

メッセージ作成元スレッドは、イベントコードを sf_message_header_t 型の共通メッセージヘッダの event_b.code ビットフィールドに設定してからメッセージをポストします。

メッセージサブスクライバスレッドは、メッセージを受信した後、メッセージヘッダに設定されているイベントコードに従ってアクションを実行します。

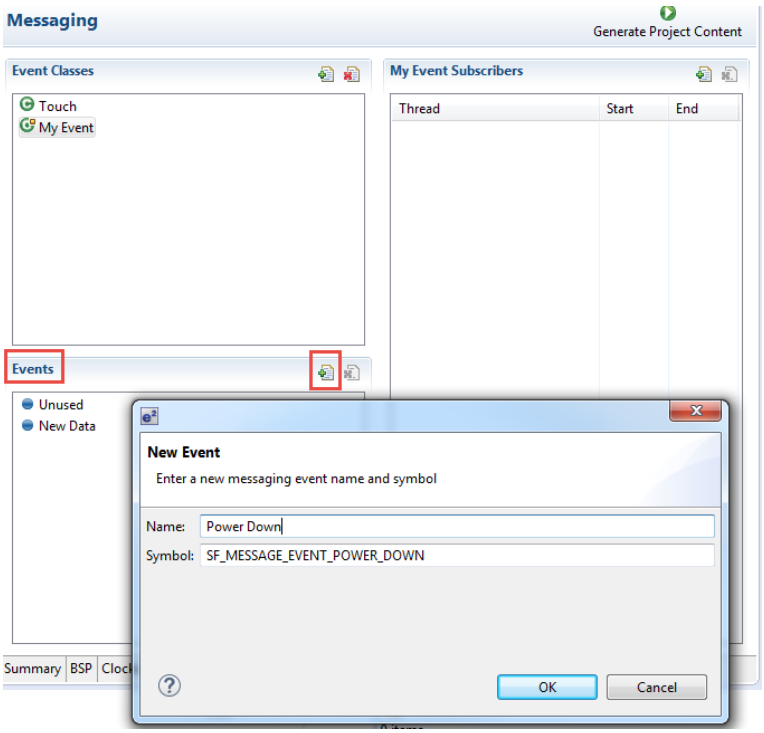


図 101:メッセージフレームワークモジュールの ISDE イベントの構成

メッセージフレームワークモジュールのサブスクライバリスト

サブスクライバリストは、フレームワークによるメッセージ配信と検索に使用されます。

フレームワークは、sf_message_subscriber_list_t インスタンスへのポインタ配列の先頭にリストされたサブスクライバグループから、各サブスクライバスレッドのメッセージキューの検索を開始します。サブスクライバリストがイベントクラスコード (event_class). によりグループ化されるという点は重要です。

フレームワークが実行時に post API 関数のサブスクライバリストを検索する際に、メッセージペイロードデータに含まれているメッセージヘッダ (sf_message_header_t::event_b.class), のイベントクラスコードが、サブスクライバグ

ループインスタンス (event_class). のものと比較されます。一致する場合、フレームワークは、次のレベルに移動し、繰り返し回数が number_of_nodes). に達するまでメッセージキューインスタンス（sf_message_subscriber_t::p_queue）を取得します。一致しない場合、フレームワークは次のサブスクライバグループを検索し、sf_message_subscriber_list_t インスタンスへのポインタ配列に NULL が見つかるまで繰り返します。

検索手順では、サブスクライバリストの先頭にあるサブスクライバグループのメッセージングスループットが最大になりますが、下位のサブスクライバグループは不利となり、メッセージングスループットが低下します。

サブスクライバリストは、すべてのメッセージサブスクライバのルックアップテーブルです。サブスクライバリストはコンパイル時に構成されます。post API 関数が呼び出されると、メモリに静的にマップされ、フレームワークにより検索されます。フレームワークは、サブスクライバリストを使用することで、メッセージの配信先のメッセージキューを決定できます。サブスクライバリストは、下図に示す 2 つの構造体（sf_message_subscriber_list_t と sf_message_subscriber_t）からなります。

- サブスクライバー スレッドのキューは、sf_message_subscriber_t インスタンスに登録されます。
- 同じイベントクラスコードのための上記インスタンスは、グループ化され、ポインタ配列に格納されます。
- サブスクライバグループのポインタ配列は、sf_message_subscriber_list_t インスタンスに登録されます。
- サブスクライバリストは、サブスクライバグループ構造体へのポインタ配列です。サブスクライバは、イベントクラスコードによってグループ化されます。
- ポインタ配列は、NULL で終端している必要があります。

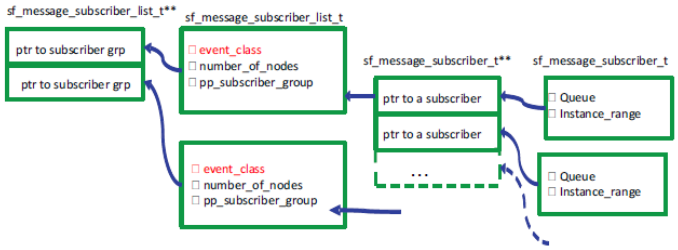


図 102:メッセージフレームワークモジュールのサブスクライバリスト

ISDE では、[Threads] タブで名付けたスレッドのイベントクラス別にグループ化されたサブスクライバを構成できます。以下の例では、スレッドは [Threads] タブで " マイスレッド " と名付けられています。開始値と終了値は、このスレッドが受け入れるイベントクラスのインスタンス数を反映しています。

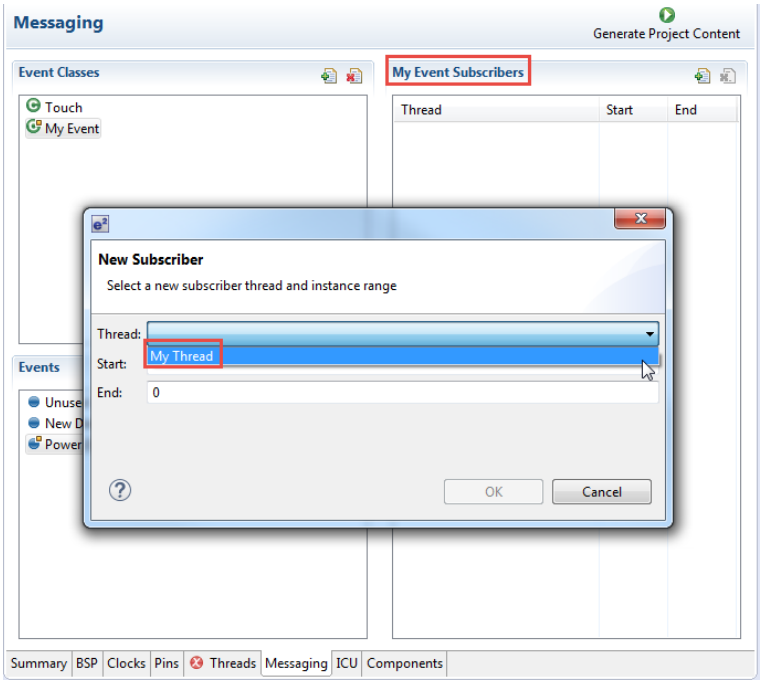


図 103:メッセージフレームワークモジュールの ISDE サブスクライバの構成

メッセージフレームワークモジュールのメッセージペイロード

メッセージペイロードは、メッセージ作成元とメッセージサブスクライバが互いに通信するために使用する構造化データです。メッセージペイロードには、メッセージ作成元がサブスクライバに発生したイベントについて通知するメッセージをポストできるよう、イベントクラスとイベントコードが共通ヘッダに含まれます（sf_message_header_t 型のデータ、イベントクラスとイベントコードを参照）。

タッチパネルフレームワークモジュールやオーディオ再生モジュールなど、事前に定義された構造体が SSP によって提供されるモジュール以外では、システム固有のメッセージペイロード構造体を定義する必要があります。メッセージペイロードには、共通ヘッダに加え、イベント処理に必要な追加データを含むことができます。

メッセージフレームワークモジュールの SSP 事前定義ペイロード

SSP には、以下の事前定義されたメッセージ ペイロード構造体が含まれています。

- sf_touch_panel_payload_t タッチ パネル フレームワーク モジュールの型
- sf_audio_playback_data_t オーディオ再生フレームワーク モジュールの型

これらのフレームワークモジュールは、内部でメッセージフレームワークを使用して、最適なメッセージペイロード構造体を定義します。タッチパネルフレームワークモジュールからのタッチイベントメッセージをサブスクライブするアプリケーションスレッドは、ヘッダファイル filesf_touch_panel_api.h. を含むことにより、事前定義されたメッセージペイロードを使用できます。同様に、オーディオイベントメッセージをオーディオ再生フレームワークモジュールにポストするアプリケーションスレッドは、sf_audio_playback_api.h. を使用できます。

メッセージフレームワークモジュールのユーザー定義ペイロード

各イベントクラスコードに対しメッセージペイロード構造体を定義する必要があります。ただし、前述の SSP 事前定義ペイロード、または共通ヘッダのみを必要とするペイロードは例外です。新しいメッセージペイロード構造体を作成するには、共通のメッセージヘッダ（sf_message_header_t 型の構造体）を、ユーザー固有のメッセージペイロード構造体の先頭に追加します。ヘッダーのサイズは 4 バイトです。

注 : ペイロードサイズは、バッファサイズ以下にしてください。

このバッファサイズによる制限はきわめて重要であり、バッファを超えて大量のデータを書き込むと、ThreadX カーネルに必要なブロックメモリプール内のデータが破壊される可能性があります。サイズの制限に違反すると、ハード故障例外が発生します。バッファサイズは sf_message_ctrl_t::buffer_size. で構成できます。

メッセージフレームワークモジュールの動作に関する重要な注意事項と制限事項

メッセージング フレームワークと OS メッセージ キュー サービス

メッセージフレームワークモジュールは、ThreadX のプリミティブメッセージキューとカーネルサービスを使用し、ThreadX RTOS の機能に対するいくつかの拡張をサポートしています。このため、メッセージフレームワークモジュールは、ThreadX のメッセージキューサービスと全く同じように動作するわけではありません。ただし、メッセージング フレームワーク モジュールを使用したメッセージング システムは、アプリケーション内で ThreadX メッセージ キュー サービスと同時に使用できます (2 つのメッセージング システムが分離されている場合)。

API 呼び出しのコンテキスト

- sf_message_api_t::open API は、1 つのスレッドからのみコールすることができます。メッセージフレームワーク制御ブロックインスタンスあたり 1 回だけコールすることができます。この関数を 2 回呼び出したときの動作は不定です。
- sf_message_api_t::close API は、1 つのスレッドからのみコールすることができます。
- sf_message_api_t::bufferAcquire API は、1 つのスレッドおよび 1 つの ISR からコールすることができます。
- sf_message_api_t::bufferRelease API は、1 つのスレッドからのみコールすることができます。
- sf_message_api_t::post API は、1 つのスレッドおよび 1 つの ISR からコールすることができます。
- sf_message_api_t::pend API は、1 つのスレッドおよび 1 つの ISR からコールすることができます。

バッファ数の見積もり

メッセージング システムを設計する際には、作業メモリ内で割り当てることができるバッファの数を正しく見積もる必要があります。バッファ数は以下のように見積もります。

$$N \approx \frac{W_m}{M_b + B_{cb} + T_x} = \frac{W_m}{M_b + 12\ bytes + 4\ bytes}$$

意味 :

- *N* - バッファ数
- *W**m* - 作業メモリサイズ（バイト単位）
- *M**b* - メッセージバッファサイズ（バイト単位）
- *B*cb = 12 *bytes* - バッファ制御ブロックのサイズ
- *T*x*x* = 4 *bytes* - ThreadX の予約バイト。

同時に割り当てることができるバッファの最大数は、システム内のメッセージ キューの深さの総数に等しくなります。理想的には、堅牢なシステムのバッファ数は、システム内のメッセージキューの深さの合計にする必要があります。

メッセージ キューのサイズと深さの設定

メッセージフレームワークモジュールは、メッセージペイロードが格納されたバッファのポインタを配信するため、メッセージキュー上に 4 バイトのメモリブロックを必要とします。このため、メッセージ キューのサイズは 4 バイトに固定する必要があります。サイズが 4 バイトよりも大きいと、メッセージフレームワーク API 関数によって内部的に実行される ThreadX メッセージキューサービスで余分なメモリがコピーされるため、性能に悪影響を与えます。

メッセージキューの深さは任意ですが、実行時にキューに格納されるメッセージ数を収容できるようにする必要があります。ガイドラインとして、次のように値を見積もります。

$$D \approx \frac{P_{avg}}{S_{avg}}$$

意味 :

- *D* - キューの深さ
- *P*avg - 作成元からの平均メッセージ配信速度
- *S*avg - サブスクライバでの平均イベントループ完了時間。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.26.4 アプリケーションへのメッセージフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにメッセージフレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

メッセージフレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(メッセージフレームワークのデフォルト名は g_sf_message0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

メッセージフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_message0 Messaging Framework on sf_message	Threads	New Stack> Framework> Services> Messaging Framework on sf_message

次の図に示すように、sf_message 上のメッセージフレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

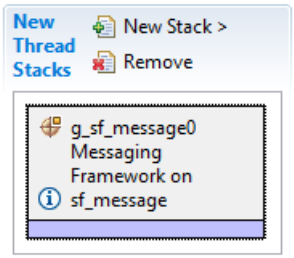


図 104:メッセージフレームワークモジュールのスタック

4.1.26.5 メッセージフレームワークモジュールの構成

ユーザーは必要な動作に合わせてメッセージフレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_message でのメッセージフレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Message Queue Depth (Total number of messages to be enqueued in a Message Queue)	16	Specify the size of Thread X Message Queue in bytes for Message Subscribers. This value is applied to all the Message Queues.
Name	g_sf_message0	The name of Messaging Framework module control block instance.

ISDE Property	Value	Description
Work memory size in bytes	2048	Specify the work memory size in bytes. Choosing a small number results a small number of buffers which can be allocated at the same time (You can confirm the total buffer number on: sf_message_instance_ctrl_t: :number_of_buffers). If the value is smaller than the peak number of messages to be posted at the same time, the Framework occurs a buffer allocation failure affecting system performance.
Pointer to subscriber list array	p_subscriber_lists	Specify the name of pointer to the Subscriber List array.
name of the block pool internally used in the messaging framework	sf_msg_blk_pool	The name of memory block memory the Framework creates in the control block. This parameter might be useful for debugging purpose but NULL can be specified for saving memory.
Name of generated initialization function	sf_message_init0	Name of generated initialization function
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

メッセージキューを作成するメッセージフレームワークモジュール

メッセージングコンフィギュレータは、サブスクライバのメッセージキューを自動的に作成します。

イベントクラスとイベントを構成するメッセージフレームワークモジュール

独自のイベントクラスでメッセージフレームワークを使用するには、ISDE のプロジェクトコンフィギュレータの [Threads] タブと [Messaging] タブを使用します。

[スレッド] タブで、次の操作を実行します。

1) [スレッド] ウィンドウの [スレッド スタック] パネルに メッセージング フレームワーク コンポーネントを追加します。
2) [スレッド] ウィンドウに新しいスレッドを追加し、一意の名前を付けます。
3) [Messaging] タブ（イベントクラスコードを参照）で、次の操作を実行します。
   a) [Event Class] ウィンドウで、イベントを追加します。
   b) サブスクライブするスレッドのイベントクラスの名前を [New Event] ダイアログボックスに入力します。

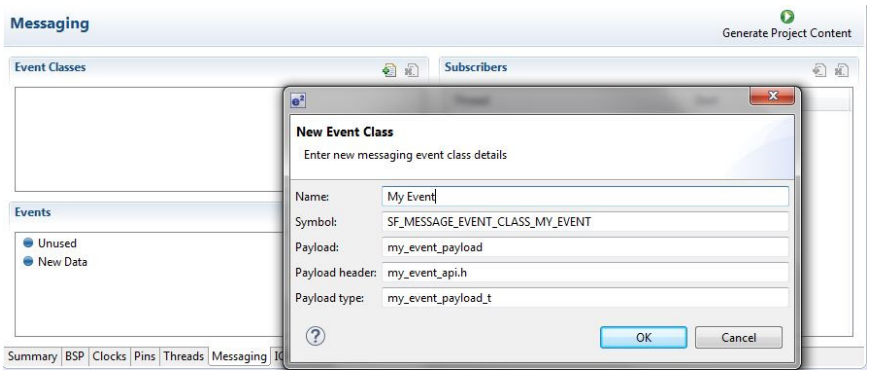


図 105:メッセージフレームワークの新しいイベントクラスの構成

4) **[Events]** ウィンドウで、アプリケーションがサポートする可能性のあるイベントを追加します（イベントコードを参照）。

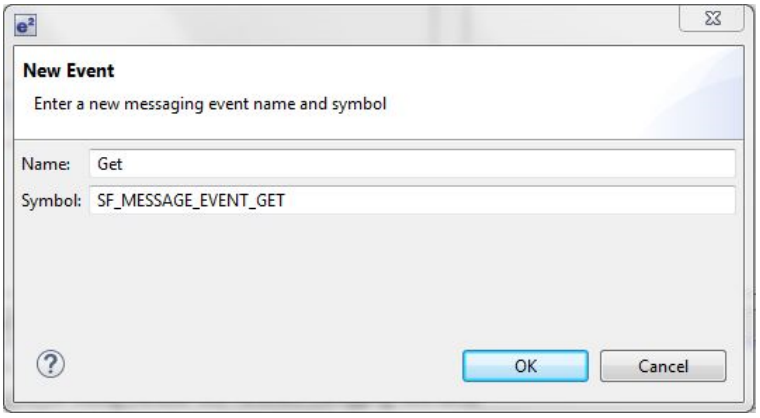


図 106:メッセージフレームワークの新しいイベントの構成

カスタムイベントクラスコードとイベントコードは sf_message_port.h. というファイルに格納されます。オーディオ再生クラスとタッチイベントクラスは、SSP で事前定義されている 2 つのイベントクラスです。タッチイベントクラスは、新しいデータイベント SF_MESSAGE_EVENT_NEW_DATA. のみを使用します。

サブスクライバリストを構成するメッセージフレームワークモジュール

[Messaging] タブ（サブスクライバリストを参照）で、次の操作を実行します

1) [Event Classes] ウィンドウでイベントクラスを選択し、イベントクラスの [Subscribers] ウィンドウでサブスクライバリストのスレッドを構成します。
2) [スレッド] ダイアログ ボックスのドロップダウン リストからスレッドを選択します。
3) [start] の横にイベントクラス instance(s) の開始番号を入力します。システムでこのイベントクラスに対し複数のイベントクラスインスタンスが使用されない場合、または指定する番号が不明な場合は、デフォルトの番号（0）のままにしておきます。使用可能な値範囲は 0 ～ 255 です。
4) [End] の横にイベントクラス instance(s) の終了番号を入力します。システムでこのイベントクラスに対し複数のイベントクラスインスタンスが使用されない場合、または指定する番号が不明な場合は、デフォルトの番号（0）のままにしておきます。使用可能な値範囲は 0 ～ 255 です。
5) [OK] をクリックします。指定したイベントのサブスクライバーがサブスクライバー リストに追加されます。
6) イベント クラス インスタンスが複数ある場合は、すべてでこの手順を繰り返します。

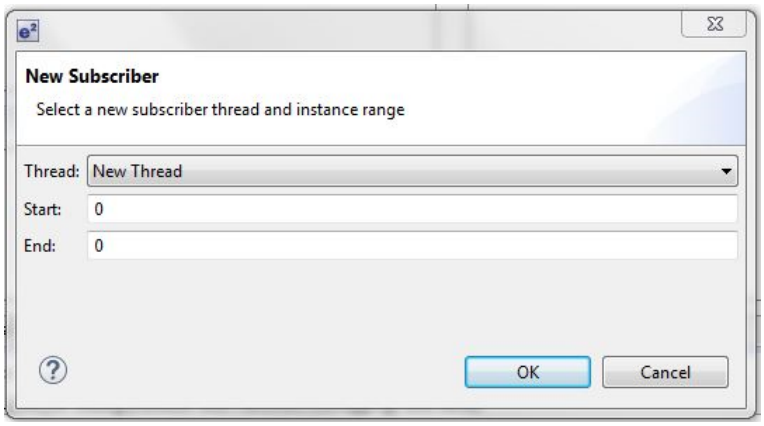


図 107:メッセージフレームワークの新しいサブスクライバの構成

イベントクラスコードとイベントコードを構成するメッセージフレームワークモジュール

独自のメッセージ ペイロード構造体を定義できます。ユーザー定義のメッセージ構造体には必ず、メンバーの 1 つとして sf_message_header_t 型の構造体が含まれている必要がありますが、他のメンバーは完全にユーザー定義が可能です。メッセージ フレームワークでは、メッセージ ペイロード構造体がどこで定義されているかは認識しません。メッセージ作成元とサブスクライバーのスレッドのソース ファイル内に独自のメッセージ ペイロード構造体を定義するファイルを含むことができます。

sf_message_cfg_t 型構成パラメータを、システムに合わせて構成します。構成構造体用のコードは、Synergy 構成ツールを使用して生成できます。[Threads] タブで、スレッドスタックに [Messaging Framework] コンポーネントを追加し、[Properties] ウィンドウでメッセージフレームワークモジュールのプロパティを変更します。[Generate Project Content] ボタンを押すと、メッセージフレームワークモジュールのコードがスレッドコードに生成されます。

メッセージをポストする前に、イベント作成元スレッドは、メッセージング フレームワーク モジュールからメッセージ用のバッファを取得する必要があります。イベント作成元スレッドは、bufferAcquire を呼び出すことでバッファを取得できます。

API 関数が SSP_SUCCESS, を返す場合、Synergy 構成ツールで構成されたメッセージバッファサイズ（バイト単位）のバッファは、メッセージフレームワークによって管理されるメモリプールに割り当てられます。割り当て可能な最大数は、Synergy 構成ツールで指定された [Work memory size in bytes] の構成によって異なります。最大数の見積もりについては、「バッファ数の見積もり」を参照してください。

sf_message_api_t::bufferAcquire API には、メッセージ送受信の動作を変えるためのいくつかのオプションがあります。

- **バッファキープ**：このオプションを使用すると、sf_message_api_t::bufferRelease API により解放されないバッファをアプリケーションスレッドが保持できます（true に設定した場合）。一般に、バッファは、メッセージが渡された後で sf_message_api_t::bufferRelease によって解放されます。ただし、スレッド間の定期的なメッセージや繰り返されるメッセージがある場合は、バッファを何度も割り当てて解放することなく同じバッファをメッセージングに再利用できます。このオプションを有効にすると、バッファの割り当て / 解放操作のオーバーヘッドを低減してシステム スループットを向上できます。
- **wait_options**: これは、待ち時間オプションであり、すべてのバッファが既に取得されている場合に有効です。任意のスレッドティックカウント、TX_WAIT_FOREVER, TX_NO_WAIT をこのオプションに設定できます。詳細については、『*ThreadX ユーザーガイド*』の ThreadX サービスコールに記載されている tx_block_allocate() の説明を参照してください。

メッセージサブスクライバスレッドは、イベント作成元によってポストされたメッセージを受信した後、バッファをフレームワークに解放する必要があります。バッファの解放は sf_message_api_t::bufferRelease を呼び出すことで実行します。システムに複数のイベント サブスクライバーがある場合、API 関数は複数回呼び出される可能性があるため、実際のバッファ解放は、イベント サブスクライバーの中の最後のメッセージ サブスクライバー スレッドによってのみ実行されます。たとえば、イベントクラスのサブスクライバグループ中に 3 つのサブスクライバがある場合、sf_message_api_t::bufferRelease を呼び出す 1 番目と 2 番目のスレッドはバッファを開放しません。3 番目のスレッドのみがバッファを解放します。sf_message_api_t::bufferAcquire により buffer keep オプションが指定されている場合、オプション SF_MESSAGE_RELEASE_OPTION_FORCED_RELEASE が API 関数の引数 option に渡される場合を除いて、バッファが解放されることはありません。(sf_message_api_t::bufferRelease API の使用方法については、「メッセージフレームワークのコールバック」も参照してください。)

この API は、ユーザーコールバック関数を呼び出して、イベント作成元スレッドとメッセージサブスクライバスレッドの間でハンドシェイクを確立するためにも使用されます。

メッセージのポスト

1) sf_message_api_t::bufferAcquire でバッファを取得した後、イベント作成元はメッセージペイロードデータをバッファに書き込むことができます。
2) 注意 : データをバッファに書き込むのはユーザーの責任であり、バッファサイズを超えてデータを書き込むと、メッセージフレームワークモジュールで致命的なエラーとなります。
3) ペイロード構造体の sf_message_header_t::event_b.class_code にイベントクラスコードを書き込みます。
4) ペイロード構造体の sf_message_header_t::event_b.code にイベントコードを書き込みます。この指定は必須ではありませんが、多くの場合に必要となります。
5) sf_message_header_t::event_b.class_instance. にイベントクラスインスタンス番号を書き込みます。システムに特定のイベントクラスのインスタンスが複数ある場合は、0 ～ 255 の番号を指定します。イベントクラスのインスタンスがシステムに 1 つしかない場合は、0 を指定します。
6) post API によってメッセージをポストします。バッファへのポインタは API に渡す際に sf_message_header_t * 型でキャストする必要があります。メッセージは、メッセージサブスクライバリストに登録されているメッセージサブスクライバに配信されます。post API には、メッセージ送受信の動作を変えるためのいくつかのオプションがあります。

    - **メッセ ― ジプライオリティ**：メッセ ― ジには、SF_MESSAGE_PRIORITY_NORMAL と SF_MESSAGE_PRIORITY_HIGH. の 2 つのレベルのメッセージプライオリティがあります。SF_MESSAGE_PRIORITY_HIGH が指定された場合、メッセージは、メッセージサブスクライバのメッセージキューの先頭に格納されます。一般にこれは、緊急メッセージに使用され、メッセージ サブスクライバーは、メッセージ キューに格納済みのイベントよりも前にそのイベント処理するようになります。

- **ユーザーコールバック関数**: この関数は、メッセージフレームワークモジュールのバッファ制御ブロックに登録されます。コールバック関数は、bufferRelease によって呼び出されます。この関数は、イベント作成元スレッドとメッセージ サブスクライバー スレッドの間でのハンドシェイクに使用できます。
- **Wait_options**: これは、待ち時間オプションであり、メッセージサブスクライバスレッドのメッセージキューが一杯の場合に有効です。任意の ThreadX ティックカウント、TX_WAIT_FOREVER TX_NO_WAIT をこのオプションに設定できます。詳細については、『ThreadX ユーザーガイド』の ThreadX サービスコール tx_queue_send() の説明を参照してください。

保留メッセージの確認

1) メッセージフレームワークモジュールがオープンされた後、メッセージサブスクライバスレッドは、pend を呼び出すことで、メッセージを待つことができます。通常、2 つ目の API 引数では、[Messaging] タブの [Thread Subscribers] ペインでメッセージサブスクライバスレッドに対して構成したメッセージキューへのポインタを指定しますが、必要に応じて他のメッセージキューを指定することもできます。
2) イベント作成元からメッセージが配信されると、スレッドは pend から戻ります。
3) API は、API の第 3 引数を介してスレッドへのメッセージを含むバッファへのポインターを返します。
4) メッセージサブスクライバは、ユーザーカスタムメッセージペイロード構造体に上記のポインタ型のポインタをキャストし、イベントクラスコード sf_message_header_t::event_b.class_code, イベントコード sf_message_header_t::event_b.code、メッセージ内にあるユーザー定義の任意のデータに応じてイベントを処理します。

pend には、API 関数の動作を変更するための wait_optionto オプションがあることに注意してください。

pend の 4 つ目の引数は待ち時間オプションであり、メッセージサブスクライバスレッドのメッセージキューが空の場合にのみ有効です。任意の ThreadX ティックカウント、TX_WAIT_FOREVER, TX_NO_WAIT をこのオプションに設定できます。詳細については、『ThreadX ユーザーガイド』の tx_queue_send()ThreadX サービスコールの説明を参照してください。

メッセージフレームワークモジュールの割り込み

メッセージフレームワークモジュールは割り込みを使用しません。

4.1.26.6 アプリケーションでのメッセージフレームワークモジュールの使用

一般的なアプリケーションで sf_message モジュール上のメッセージフレームワークを使用する際の手順は次のとおりです。

- メッセージキューの作成
- イベントクラスとイベントの構成
- サブスクライバリストの構成
- イベントクラスコードとイベントコードの構成

構成が完了したら、次のようにモジュールの API をターゲットアプリケーションで使用できます。

1) sf_message_api_t::open API でメッセージフレームワークを初期化する
2) sf_message_api_t::bufferAcquire API を使用してバッファを取得する
3) sf_message_api_t::post API でメッセージをポストする
4) sf_message_api_t::pend API で保留メッセージを確認する
5) sf_message_api_t::bufferRelease API を使用してバッファを解放する

6) sf_message_api_t::close API でメッセージフレームワークを閉じる

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

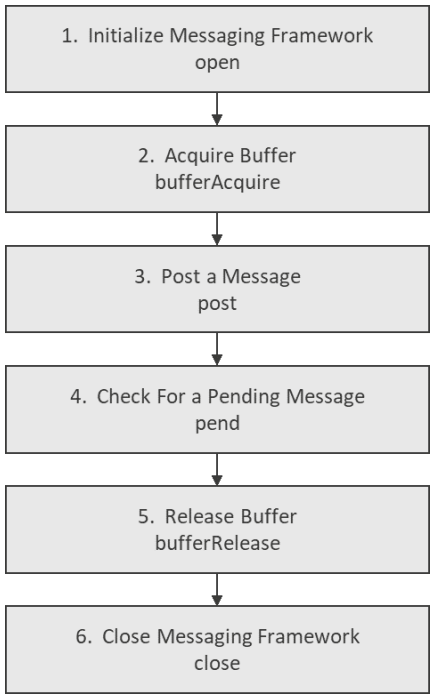


図 108:通常のメッセージフレームワークモジュールアプリケーションのフロー図

4.1.27 パワー プロファイル フレームワーク

パワープロファイルフレームワークでは、より低電力のソフトウェアスタンバイモードで MCU を配置できるように構成が可能な汎用 API が提供されます。

4.1.27.1 パワープロファイルフレームワークモジュールの特長

- ソフトウェアスタンバイ操作をサポート
- 複数の動作モードをサポート
  - 実行
  - BSP_MCU_REG_PROTECT_S3A7
  - 外部割り込み
- ソフトウェアスタンバイ時に無効にするクロックと周辺機能を選択
- 既存のソフトウェアスタンバイの前後に出力ピン状態を選択

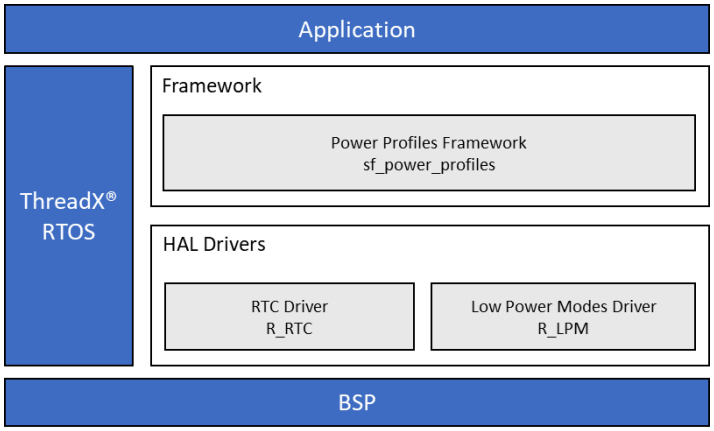


図 109:パワープロファイルフレームワークモジュールのブロック図

4.1.27.2 パワープロファイルフレームワークモジュールの API の概要

パワープロファイルでは、オープン、スリープ、クローズなどの一般的な機能の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

パワープロファイルフレームワークモジュール API の要約

Function Name	Example API Call and Description
open	g_sf_power_profiles0.p_api->open(g_sf_power_profiles0.p_ctrl, g_sf_power_profiles0.p_cfg); Initialize the Power Profiles Framework.
sleep	g_sf_power_profiles0.p_api->sleep(g_sf_power_profiles0.p_ctrl) Places the MCU in software standby mode.
close	g_sf_power_profiles0.p_api->close(g_sf_power_profiles0.p_ctrl) Close the Power Profiles Framework.

Function Name	Example API Call and Description
versionGet	g_sf_power_profiles0.p_api->versionGet(&p_version); Get version and store it in provided pointer p_version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successful.
SSP_ERR_ASSERTION	Assertion error.
SSP_ERR_IN_USE	The framework has already been initialized.
SSP_ERR_INVALID_HW_CONDITION	Incompatible system clock configuration.
SSP_ERR_NOT_OPEN	Device not open.
SSP_ERR_UNSUPPORTED	The function is not supported by the module.
SSP_ERR_INTERNAL	Internal error.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.27.3 パワープロファイルフレームワークモジュールの動作の概要

パワー プロファイル フレームワークは、Synergy マイクロコントローラ ハードウェアの RTC、LPM、IOPORT および CGC 周辺機器を使用するもので、ローパワー動作モードにアクセスする際に使いやすいソフトウェア インタフェースとなっています。パワー プロファイル フレームワークにより、事前に定義された状態でソフトウェア スタンバイ モードの開始と終了を実行できるように、システムが設定されます。パワー プロファイル フレームワークは、スレッド環境と非スレッド環境の両方で使用できます。

パワー プロファイル モジュールは、ユーザーが行う初期設定により動作モード (RTC または外部割り込み) が決定されるという形で動作します。パワープロファイルを使用して、ユーザーは、2 つのピン構成表（ソフトウェアスタンバイおよび復帰）を定義できます。構成表により、ソフトウェアスタンバイモード中とその後の復帰時の MCU ポートピンの状態が定義されます。両方の表は、ともに bsp_pin_cfg.h ファイルをベースとして使用し、これに従って表の名前などを含むエントリを修正することにより生成できます。

それらの表へのポインタは、復帰およびソフトウェア スタンバイのピン設定に関するパワー プロファイルのプロパティに記載されています。これはピン設定表の定義にあたって必須ではありません。これらのポインタの一方または両方に NULL を指定することにより、それぞれのスリープ / 復帰状態に関するピンの再設定が防止されます。

パワープロファイルフレームワークモジュールの動作に関する重要な注意事項と制限事項

パワープロファイルフレームワークを使用すると、MCU をソフトウェアスタンバイモードに設定できます。ソフトウェアスタンバイモードでは、消費電力を非常に低く抑えながら、すべての RAM および操作パラメータを保持できます。外部割り込みモードは、主に RTC に加えて LOCO およびサブクロックのクロックソースがオフになっているため、最小限の電力しか使用しないモードです。

RTC モードでは、RTC は動作を許可されているため、結果的により多くの電力を使用します。通常 MCU がソフトウェア スタンバイ モード中に動作の継続を許可している周辺機器またはクロックは、ソフトウェア スタンバイ モードに入る前にパワー プロファイル モジュールによりオフにされ、その後復帰時にオンになります。

この動作対象から除外したい動作が他にある場合は（LOCO をソフトウェアスタンバイ中も動作させる場合など）、SF_POWER_PROFILES_EVENT_PRE_SLEEP または SF_POWER_PROFILES_EVENT_POST_SLEEP コールバックイベントを使用し、特定の動作を含めることで可能となります。

注 : パワープロファイルフレームワークは番号付きの IRQ（IRQ0-IRQ15...）による復帰はできないため、追加の作業が必要です。番号付きの IRQ を使用して MCU から復帰するには、アプリケーションで r_lpm API 関数 lpm_api_t::wupenGet と lpm_api_t::wupenSet を使用して、番号付きの IRQ によるスタンバイモードからの復帰を有効にする必要があります。

パワープロファイルフレームワークでサポートされるオペレーティングシステム

ThreadX と併用した場合、このフレームワークはミューテックスのような ThreadX に固有のオブジェクトを使用します。ThreadX を利用した動作はオプションです。

スリープモードの開始と終了

次のブロック図は、スリープ開始（ソフトウェアスタンバイ）モードの処理フローを示したものです。

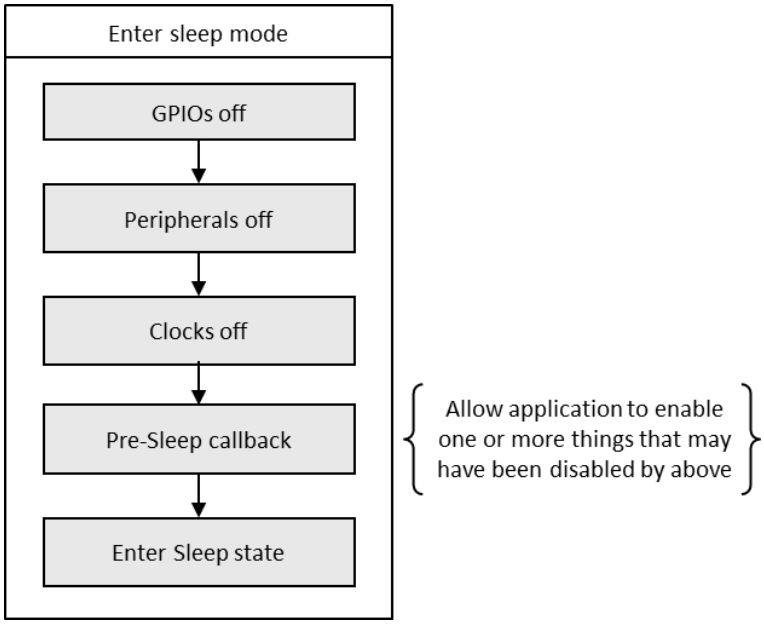


図 110:パワープロファイルフレームワークモジュールのスリープモードの開始

次のブロック図は、スリープ（ソフトウェアスタンバイ）モードからの復帰に関する処理フローを示したものです。

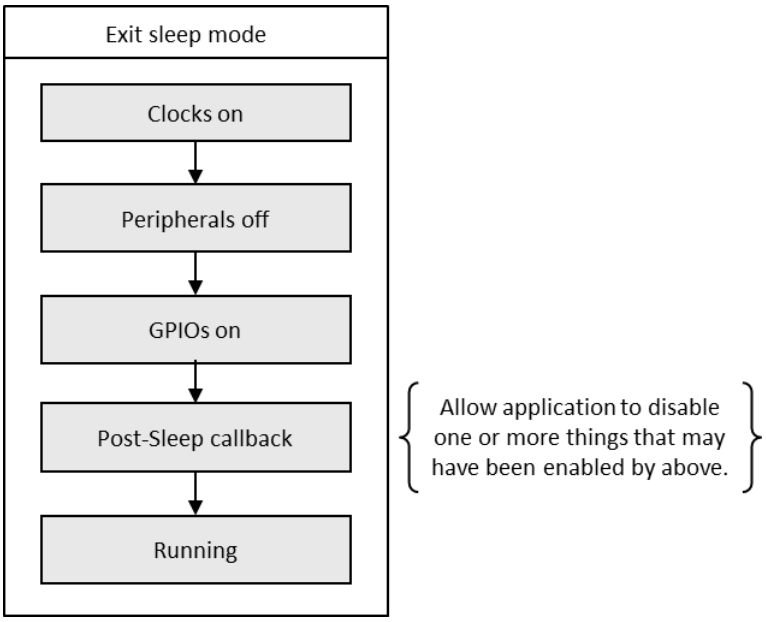


図 111:パワープロファイルフレームワークモジュールのスリープモードの終了

パワープロファイルフレームワークは、現在ソフトウェアスタンバイ以外のどのスリープモードもサポートしていません。

SF_POWER_PROFILES_V2 フレームワークは SF_POWER_PROFILES フレームワークに優先します。プロジェクトでは SF_POWER_PROFILES の使用を継続できますが、SF_POWER_PROFILES_V2. に更新することをお勧めします。新しいプロジェクトでは、SF_POWER_PROFILES. ではなく SF_POWER_PROFILES_V2 を使用する必要があります。SF_POWER_PROFILES は R_LPM, で使用する必要があり、SF_POWER_PROFILES_V2 は R_LPM_V2. で使用する必要があります。相違点と特長については、SF_POWER_PROFILES V2 の使用上の注意を参照してください。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.27.4 アプリケーションへのパワープロファイルフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにパワープロファイルフレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

パワープロファイルフレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（パワープロファイルフレームワークのデフォルト名は g_sf_power_profiles0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

パワープロファイルフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_power_profiles0 Power Profiles Framework on sf_power_profiles	Threads	New Stack > Framework > Services > Power Profiles Framework on sf_power_profiles

次の図に示すように、sf_power_profiles 上のパワープロファイルフレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

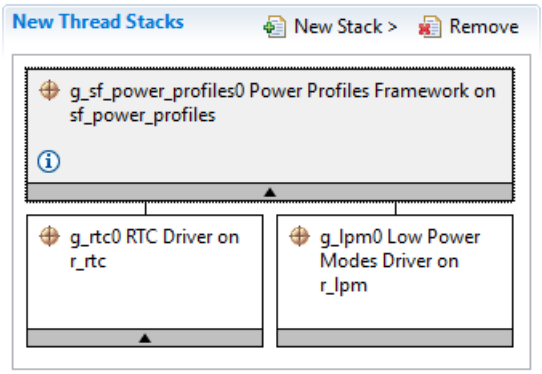


図 112:パワープロファイルフレームワークモジュールのスタック

4.1.27.5 パワープロファイルフレームワークモジュールの構成

ユーザーは必要な動作に合わせてパワープロファイルフレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_power_profiles でのパワープロファイルフレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Controls whether to include code for API parameter checking.
RTC Support	Enabled, Disabled Default: Disabled	Required to be Enabled when using RTC mode. Setting to Disabled when in External mode will slightly reduce the module code size.
Name	g_sf_power_profiles0	Module name.
Callback	NULL	Specify the name (if any) of the Pre-Software Standby/Post-Software Standby callback function.
Wakeup pin config table	NULL	This name corresponds to a pointer to a user defined table of port pin settings to be used subsequent to waking up.
Sleep pin config table	NULL	This name corresponds to a pointer to a user defined table of port pin settings to be used just prior to entering Software Standby.
Operating mode	Run, RTC, External Default: Run	RUN, RTC, or External Interrupt mode.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

パワープロファイルフレームワークモジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_rtc での RTC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.
Name	g_rtc0	The name to be used for the RTC module control block instance. This name is also used as the prefix of the other variable instances. See the example code below
Clock Source	LOCO, Sub-clock Default: LOCO	Clock source for the RTC block.
Error Adjustment Value	0	Warning: Deprecated configuration field. Must be 0.
Error Adjustment Type	None, Add prescaler, Subtract prescaler Default: None	Warning: Deprecated configuration field. Must be 0.
Callback	NULL	The name of the ISR that is called when one of the three interrupts fire. The argument passed into this ISR has an indication of which interrupt caused it to be called. See the example code below.
Alarm Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Disabled	Alarm interrupt priority selection

ISDE Property	Value	Description
Period Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Disabled	Period interrupt priority selection
Carry Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Carry interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_lpm 上の低消費電力モード HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_lpm0	Module name.
Operating power mode	High speed operating mode, middle speed operating mode, low speed operating mode, low voltage operating mode Default: high speed operating mode	Select operating power.
Sub-oscillator mode	Sub oscillator mode not enabled, Sub oscillator mode enabled. Default: Sub oscillator mode not enabled	Select sub oscillator.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

パワープロファイルフレームワークモジュールのクロック構成

パワープロファイルフレームワークは、固有のクロック設定を必要としません。

パワープロファイルフレームワークモジュールのピン構成

このアプリケーションでは、ソフトウェアスタンバイ開始前とソフトウェアスタンバイからの復帰後のポートピン状態の設定に使用する、ソフトウェアスタンバイと復帰時のピン構成表を任意に利用できます。

4.1.27.6 アプリケーションでのパワープロファイルフレームワークモジュールの使用

モジュールの設定が完了し、ISDE ファイルが生成されたら、以下の手順に従ってパワー プロファイル フレームワークをスレッドで使用します。パワー プロファイルは、任意にスタンドアロン (非スレッド環境) アプリケーションで使用できます。

パワープロファイルフレームワークフィールド p_callback. に構成したコールバック関数の本体を定義します。MCU がソフトウェア スタンバイ モードに移行する直前と、ソフトウェア スタンバイ モードから復帰した直後に、コールバック関数によってアプリケーションに通知が行われます。コールバックの使用は任意ですが、パワープロファイルのプロパティにコールバックを 1 つ定義している場合は、それに対する定義が必要です。

注：パワープロファイルは番号付きの IRQ（IRQ0-IRQ15...）による復帰はできません。番号付きの IRQ を使用して MCU からウェイクアップするには、アプリケーションで r_lpm API 関数 lpm_api_t::wupenGet() と lpm_api_t::wupenSet() を使用して、番号付きの IRQ によるスタンバイモードからのウェイクアップを有効にする必要があります。

一般的なアプリケーションでパワープロファイルフレームワークモジュールを使用する際の手順は次のとおりです。

1) アプリケーションスレッドで、sf_power_profiles_api_t::open API を使用してフレームワークを初期化します。
2) sf_power_profiles_api_t::sleep API によりソフトウェアスタンバイモードに入ります。
3) sf_power_profiles_api_t::close API をコールしてフレームワークを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

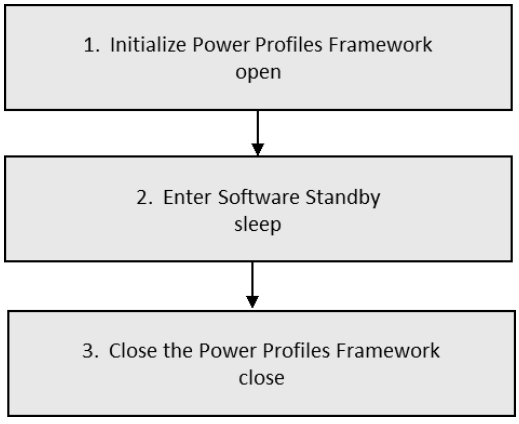


図 113:通常のパワープロファイルフレームワークモジュールアプリケーションのフロー図

4.1.28 パワープロファイル V2 フレームワーク

パワープロファイル V2 フレームワークでは、MCU のシステムクロック、I/O ポート、動作モード（クロック制御による間接的な制御）および低消費電力モードを制御するのに使用するハイレベル API が提供されます。パワープロファイル V2 フレームワークを LPM V2 ドライバー、CGC ドライバー、I/O ポートドライバーとともに使用すると、MCU の電力消費に対する高度な制御が可能となります。

4.1.28.1 パワープロファイル V2 フレームワークモジュールの特長

- 低消費電力モード V2 の使用
- 低消費電力モードの開始および終了時における CGC クロック構成および I/O ポートピン構成の設定
- スレッド動作と非スレッド動作の両方をサポート

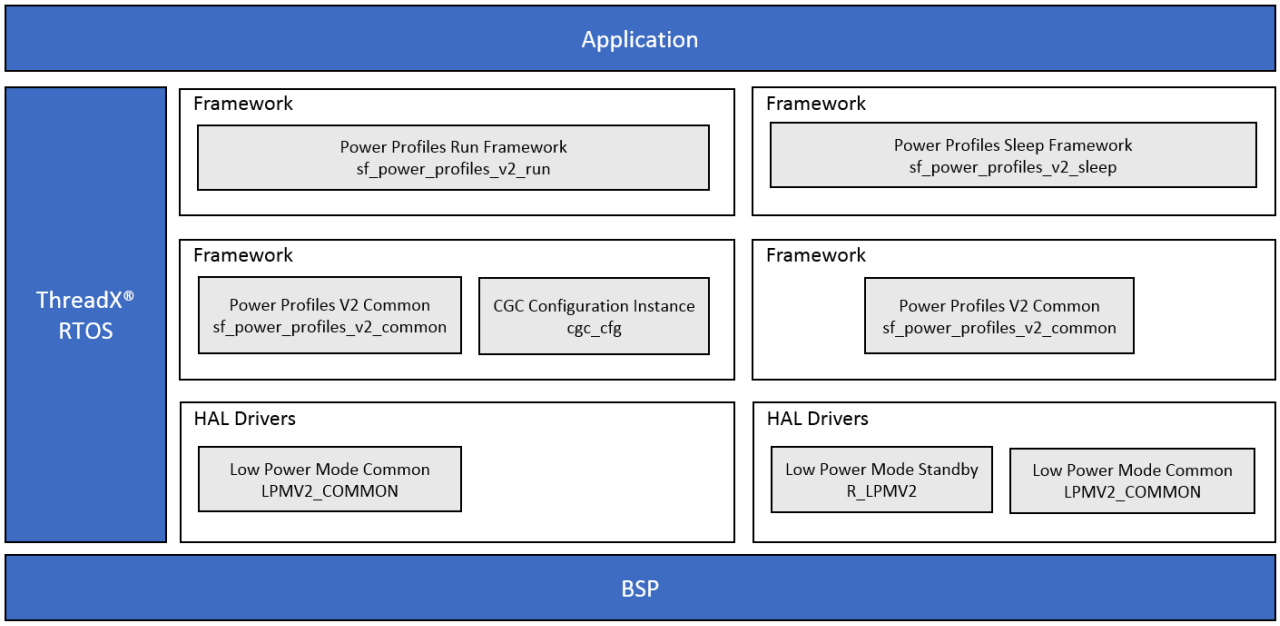


図 114:パワープロファイル V2 フレームワークモジュールのブロック図

4.1.28.2 パワープロファイル V2 フレームワークモジュール API の概要

ターゲット MCU に応じてフレームワークで使用されるさまざまなローレベル LPM V2 HAL モジュールがあります（次の表を参照）。

MCU	Driver
S124	S124 Low Power Mode Sleep on r_lpmv2
S124	S124 Low Power Mode Standby on r_lpmv2

MCU	Driver
S128	S128 Low Power Mode Sleep on r_lpmv2
S128	S128 Low Power Mode Standby on r_lpmv2
S3A3	S3A3 Low Power Mode Sleep on r_lpmv2
S3A3	S3A3 Low Power Mode Standby on r_lpmv2
S3A7	S3A7 Low Power Mode Sleep on r_lpmv2
S3A7	S3A7 Low Power Mode Standby on r_lpmv2
S5D9	S5D9 Low Power Mode Sleep on r_lpmv2
S5D9	S5D9 Low Power Mode Standby on r_lpmv2
S5D9	S5D9 Low Power Mode Deep Standby on r_lpmv2
S7G2	S7G2 Low Power Mode Sleep on r_lpmv2
S7G2	S7G2 Low Power Mode Standby on r_lpmv2
S7G2	S7G2 Low Power Mode Deep Standby on r_lpmv2

使用可能なローレベル LPM V2 HAL モジュールは数多くあるため、それぞれの API と構成設定を識別するのは困難です。このモジュールガイドでは、S7G2 MCU の詳細のみを示します。すべての API と構成設定は同じです（ディープスタンバイをサポートしない MCU を除く）。ターゲット MCU に簡単に推定されます。

パワープロファイルでは、オープン、スリープ、クローズなどの一般的な機能の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

パワープロファイル V2 フレームワークモジュール API の要約

Function Name	Example API Call and Description
open	g_sf_power_profiles_v2_common.p_api->open(g_sf_power_profiles_v2_low_power_0.p_ctrl, g_sf_powevr_profiles0.p_cfg); Initialize the Power Profiles V2 Framework.

Function Name	Example API Call and Description
runApply	g_sf_power_profiles_v2_common.p_api->runApply(g_sf_power_profiles_v2_common.p_ctrl, &g_sf_power_profiles_v2_run_0); Apply the selected run power profile.
lowPowerApply	g_sf_power_profiles_v2_common.p_api->lowPowerApply(g_sf_power_profiles_v2_common.p_ctrl, &g_sf_power_profiles_v2_low_power_0); Apply the selected low power profile.
close	g_sf_power_profiles_v2_common.p_api->close(g_sf_power_profiles_v2_common.p_ctrl); Close the module.
versionGet	g_sf_power_profiles_v2_common.p_api->versionGet(&p_version); Get version and store it in provided pointer p_version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successful.
SSP_ERR_ASSERTION	Assertion error.
SSP_ERR_IN_USE	The framework has already been initialized.
SSP_ERR_INVALID_HW_CONDITION	Incompatible system clock configuration.
SSP_ERR_NOT_OPEN	Device not open.

Name	Description
SSP_ERR_UNSUPPORTED	The function is not supported by the module.
SSP_ERR_INTERNAL	Internal error.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.28.3 パワープロファイル V2 フレームワークモジュールの動作の概要

パワープロファイル V2 フレームワークでは、MCU のシステムクロック、I/O ポート、動作モード（クロック制御による間接的な制御）および低消費電力モードを制御するのに使用するハイレベル API が提供されます。パワープロファイル V2 フレームワークを LPM V2 ドライバー、CGC ドライバー、I/O ポートドライバーとともに使用すると、MCU の電力消費に対する高度な制御が可能となります。

パワープロファイル V2 フレームワークでは、MCU の消費電力を制御するための 2 つのメイン関数 sf_power_profiles_v2_api_t::runApply および sf_power_profiles_v2_api_t::lowPowerApply が提供されます。runApply() 関数は、CGC クロック構成と I/O ポートピン構成を使用して、MCU のシステムクロックと I/O ポートピンを設定します。lowPowerApply() 関数は、構成された低消費電力モードの開始前と低消費電力モードからの復帰後に LPM V2 構成と 2 つの I/O ポート構成を使用して低消費電力モードと I/O ポートピンを設定します。使用可能な低消費電力モードの詳細については LPM V2 モジュールの概要と MCU ハードウェアマニュアルを参照してください。

パワープロファイル V2 フレームワークは、Synergy Software Package の LPM V2、IOPORT および CGC ドライバーを使用し、MCU の電力消費を制御するための使いやすいソフトウェアインタフェースを提供します。

パワープロファイル（V1）フレームワークとの比較

パワープロファイル（V1）フレームワークは LPM V1 ドライバーを使用します。パワープロファイル（V1）と LPM（V1）の両方にある制約のため、ローパワー機能のサブセットのみがサポートされます。LPM V2 の開発では、さらに多くのローパワー機能をユーザーに公開できるよう LPM V2 を設計しました。また、これらの機能を新しいフレームワークモジュールでも使用できるように設計しました。パワープロファイル（V1）は LPM（V1）を使用しているため、LPM V2 をサポートするためにパワープロファイル V2 が開発されました。パワープロファイル V2 は、パワープロファイル（V1）のすべての機能と、このドキュメントで説明されている他の機能をサポートしています。インタフェースは似ていますが、パワープロファイル（V1）と V2 には互換性はありません。

注 : パワープロファイル（V1）モジュールとパワープロファイル V2 モジュールは、同じプロジェクト内では使用できません。すべての新規プロジェクトでは、アプリケーションでパワープロファイル V2 を使用することをお勧めします。

動作の説明

パワープロファイル V2 フレームワークは、実行状態とローパワー状態の両方でシステムを構成します。MCU のシステムクロック、I/O ピンおよび低消費電力モードは、すべてパワープロファイル V2 フレームワークを使用することで、処理と制御が行われます。

パワープロファイル V2 フレームワーク API 関数 sf_power_profiles_v2_api_t::open は、パワープロファイル V2 フレームワークの内部変数、インスタンス変数、および LPM V2 ドライバーを初期化します。プロジェクトが ThreadX を使用している場合は、フレームワークによってマルチスレッド環境で安全に使用できます。

runApply() および lowPowerApply() 関数は、オプションで I/O ポートのピン構成を使用して、MCU I/O ポートを制御します。lowPowerApply() 関数では、2 つのピン構成を使用できます。1 つ目は、MCU が命令を実行しないときに低消費電力モードに適した状態にピンを設定する構成です。2 つ目は、低消費電力モードからの復帰後、命令の実行を再開するための構成です。

パワープロファイル V2 フレームワークの runApply() 関数は、ユーザーが定義したオプションのピン構成を適用し、さらにユーザーが定義した CGC クロック構成を適用します。ユーザーは、CGC クロック構成構造体を使用して、クロックのオンとオフの切り替え、クロック分周器の変更、システムクロックの切り替えを行うことができます。

パワープロファイル V2 フレームワークの API 関数 runApply() は、次のタスクを順番に実行します。

1) プロジェクトが ThreadX を使用している場合、この関数はローレベルドライバーをコールする前に ThreadX ミューテックスを取得します。
2) ユーザーが指定したオプションのピン構成を適用します。
3) ユーザーが指定した CGC クロック構成を適用します。
4) プロジェクトが ThreadX を使用している場合、この関数は ThreadX ミューテックスを返します。

lowPowerApply() 関数は、LPM V2 構成を使用して低消費電力モード、低消費電力モードからの復帰のトリガ、バスピンの状態およびその他の MCU 固有の設定を設定します。lowPowerApply() 関数は、オプションでアプリケーションコールバック関数を使用できます。プロトタイプは、/src/synergy_gen/hal_data.c または /src/synergy_gen/< スレッド名 >.c.

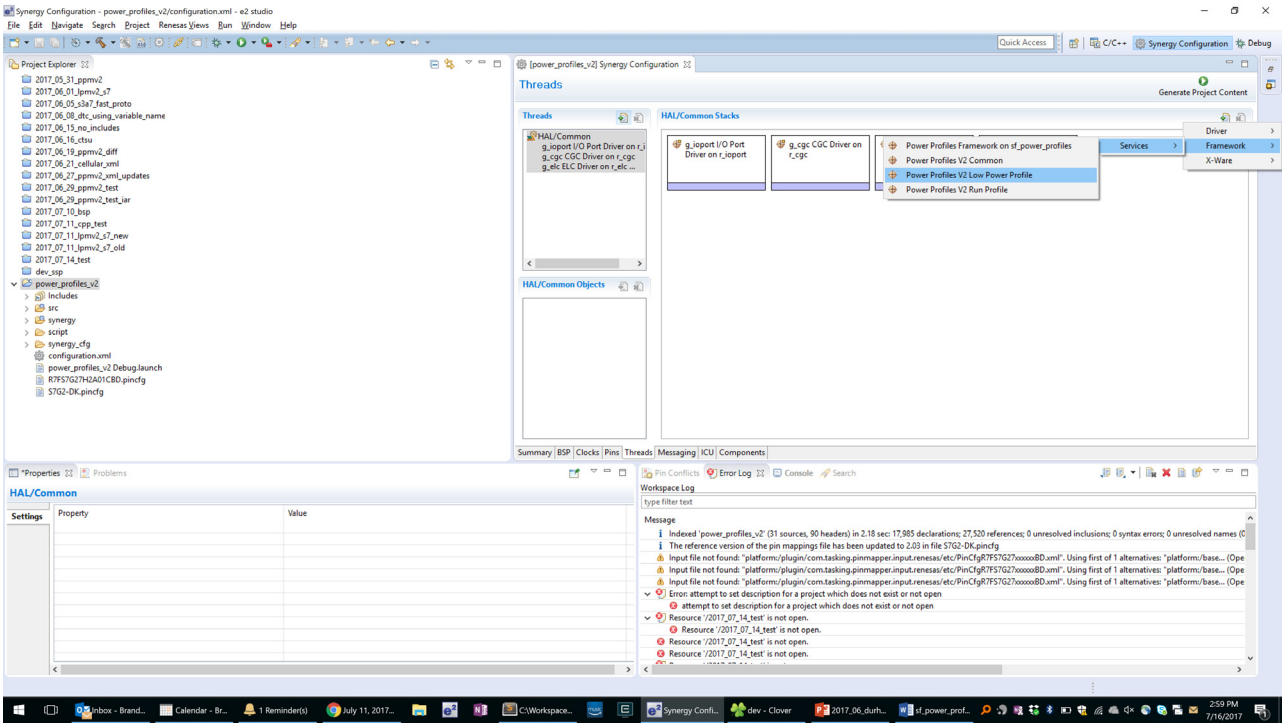
にあります。

パワープロファイル V2 フレームワークの lowPowerApply() 関数は、次のタスクを順番に実行します。

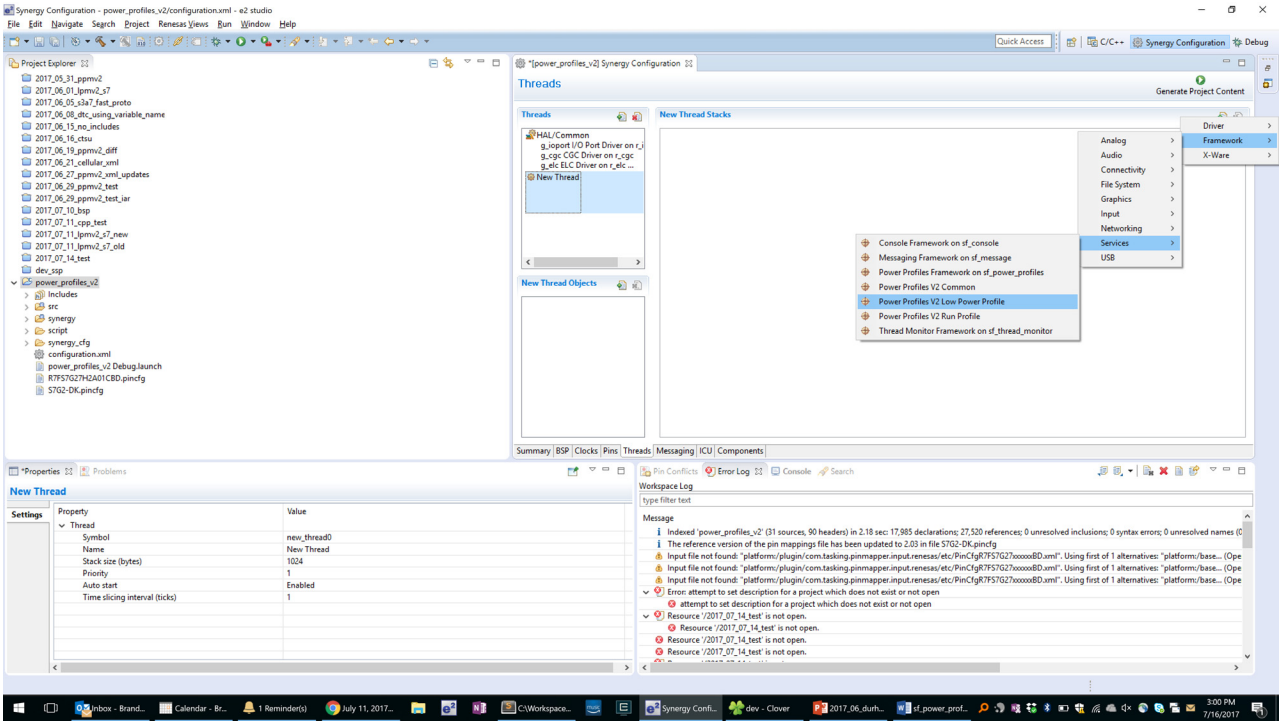
1) プロジェクトが ThreadX を使用している場合、この関数はローレベルドライバーをコールする前に ThreadX ミューテックスを取得します。
2) ユーザーが指定したオプションのローパワー開始ピン構成を適用します。
3) 列挙 SF_POWER_PROFILES_V2_EVENT_PRE_LOW_POWER. を使用して、ユーザーが指定したコールバック関数をコールします。
4) ユーザーが指定した低消費電力モード構成を適用します。任意の有効な LPM V2 構成を使用できます。
5) 低消費電力モードを開始します。
6) 選択された低消費電力モードがディープスタンバイ以外だった場合は、復帰トリガが検出されると MCU は同じポイントからコードの実行を再開します。(LPM V2 低消費電力モード構成がディープスタンバイだった場合は、復帰トリガが検出されると MCU はコードの実行を再開しないでソフトリセットを行います。)
7) ユーザーが指定したオプションのローパワー終了ピン構成を適用します。
8) 列挙 SF_POWER_PROFILES_V2_EVENT_POST_LOW_POWER. を使用して、ユーザーが指定したコールバック関数をコールします。
9) プロジェクトが ThreadX を使用している場合、この関数は ThreadX ミューテックスを返します。

ユーザーは、パワープロファイル V2 実行プロファイルまたはローパワープロファイルをプロジェクトに追加する必要があります。パワープロファイル V2 共通モジュールが自動的に追加されます。

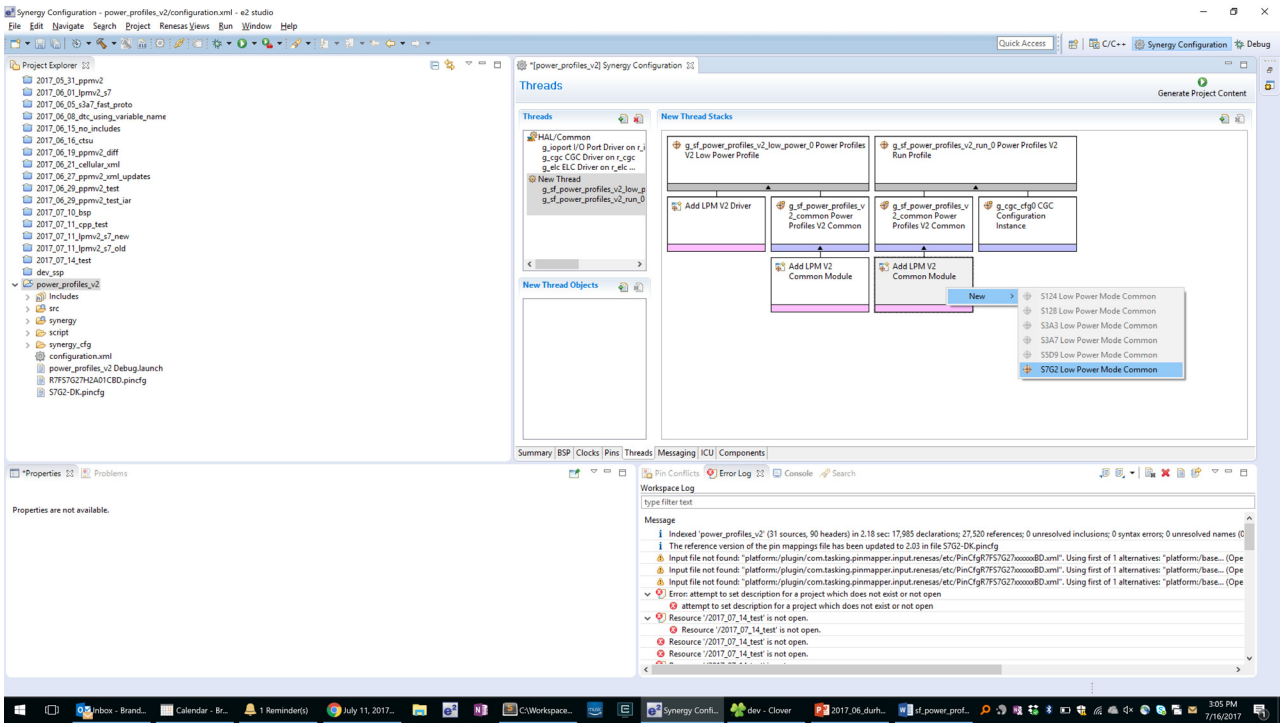
スレッド外：


Synergy Configuration - power_profiles_v2/configuration.xml - e2 studio
File Edit Navigate Search Project Renesas Views Run Window Help
Project Explorer
[power_profiles_v2] Synergy Configuration
Threads
Generate Project Content
HAL/Common Stacks
HAL/Common Objects
Power Profiles Framework on sf_power_profiles
Power Profiles V2 Common
Power Profiles V2 Low Power Profile
Power Profiles V2 Run Profile
Services
Driver
Framework
X-Ware
Summary BSP Clocks Pins Threads Messaging ICU Components
*Properties
Problems
HAL/Common
Settings
Property
Value
Pin Conflicts
Error Log
Console
Search
Workspace Log
Message

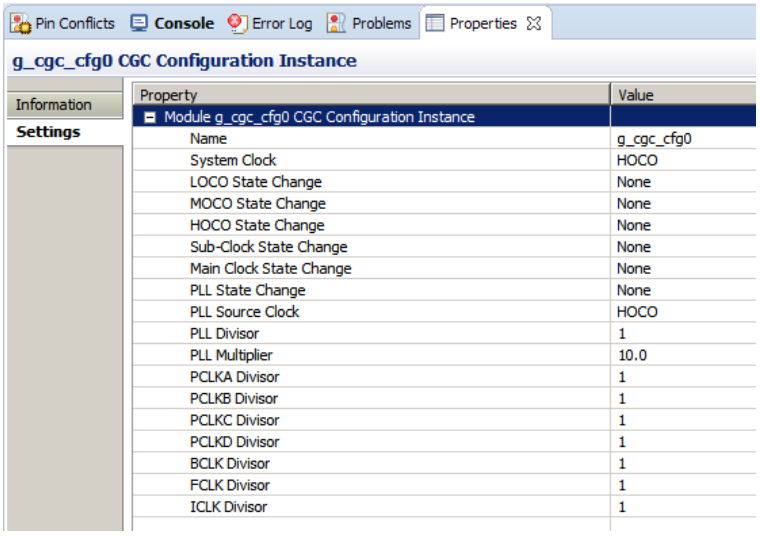
スレッド内：



パワープロファイル V2 実行プロファイルまたはローパワープロファイルの追加後、LPM V2 共通モジュールを追加する必要があります。LPM V2 共通インスタンスが既にプロジェクトに存在する場合は、そのインスタンスが使用されます。パワープロファイル V2 実行プロファイルは直接 LPM V2 を使用しませんが、引き続きビルドの成功との依存関係があります。

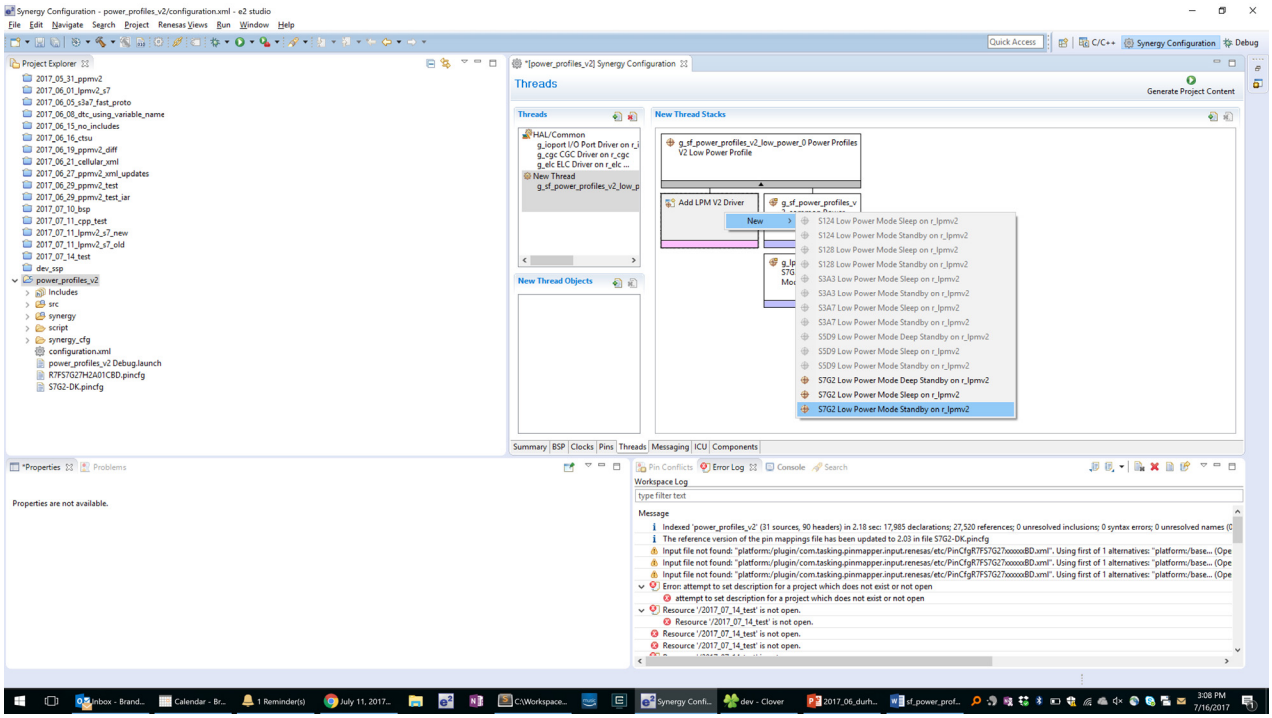


パワープロファイル V2 実行プロファイルは、CGC クロック構成に依存します。以下に、CGC クロック構成インスタンスの構成オプションを示します。CGC クロック構成インスタンスがプロジェクトに存在しない場合は、自動的に追加されます。存在する場合は、そのインスタンスを使用できます。システムクロックの制御は、MCU の電力消費を制御するうえで非常に重要な操作です。詳細については、CGC の使用上の注意を参照してください。



Property	Value
Module g_cgc_cfg0 CGC Configuration Instance	
Name	g_cgc_cfg0
System Clock	HOCO
LOCO State Change	None
MOCO State Change	None
HOCO State Change	None
Sub-Clock State Change	None
Main Clock State Change	None
PLL State Change	None
PLL Source Clock	HOCO
PLL Divisor	1
PLL Multiplier	10.0
PCLKA Divisor	1
PCLKB Divisor	1
PCLKC Divisor	1
PCLKD Divisor	1
BCLK Divisor	1
FCLK Divisor	1
ICLK Divisor	1

パワープロファイル V2 ローパワープロファイルは、LPM V2 スタンバイインスタンスを使用しますが、現在使用中の MCU に応じて、LPM V2 ディープスタンバイインスタンスまたは LPM V2 スリープインスタンスを使用できます。



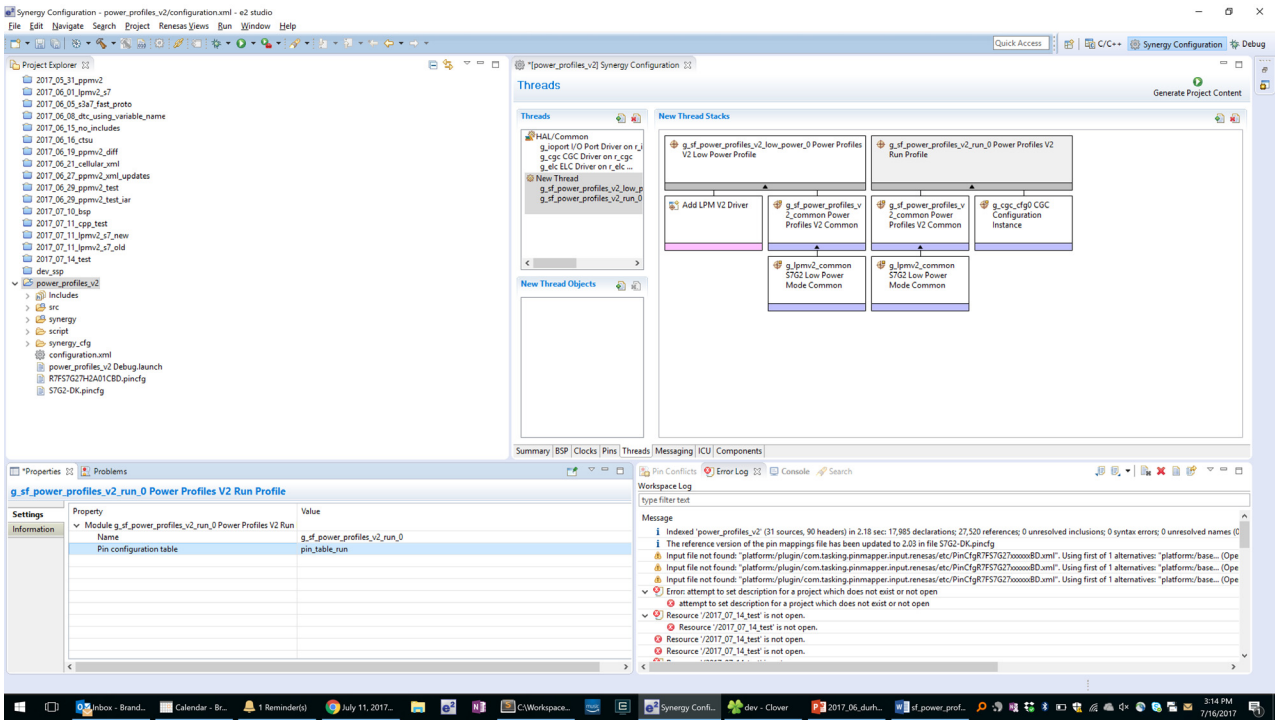
LPM V2 インスタンスのプロパティは、インスタンスを選択し、[Properties] ペインを確認することで構成できます。

パワープロファイル V2 フレームワークのピンの構成

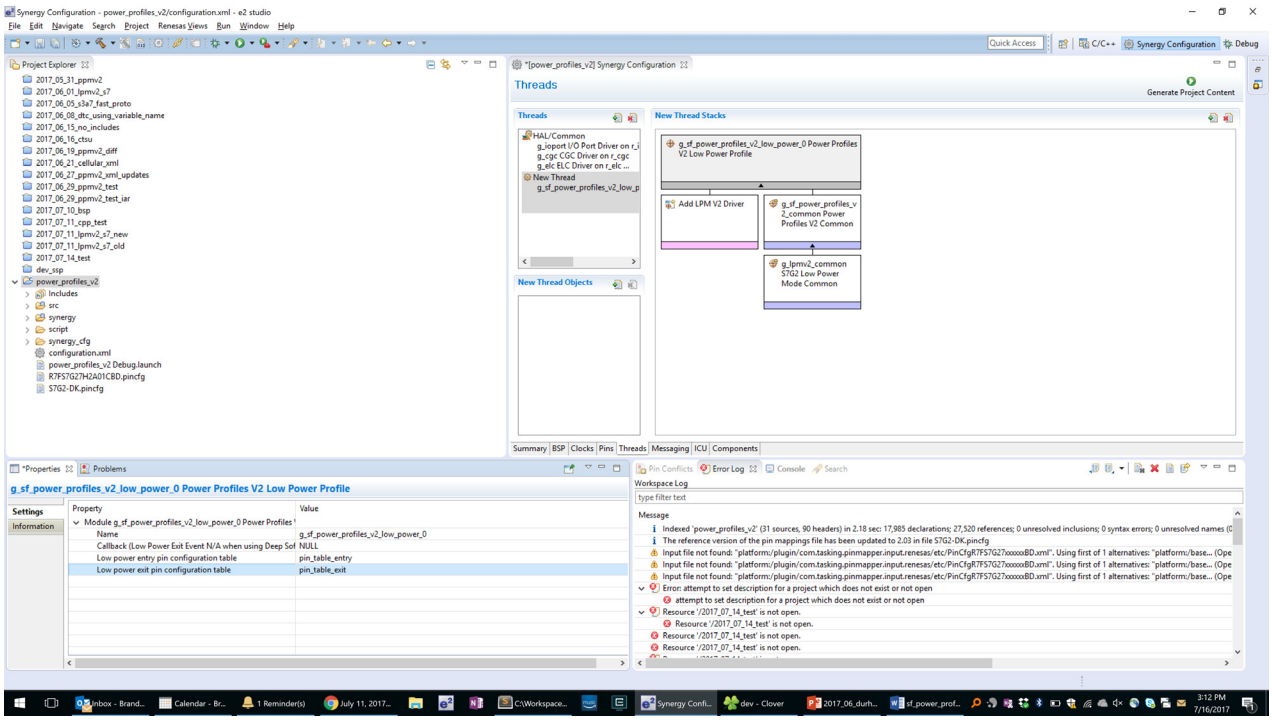
オプション：次のように、[Pins] タブを使用して追加の I/O ポートピン構成を作成します。

1) プロジェクトの最上位ディレクトリで、ファイル拡張子が *.pincfg のファイルを探します。
2) そのファイルのコピーを作成して、名前を変更し、プロジェクトの同じ最上位ディレクトリに新しいファイルを保存します。
3) これで、新しいファイルは Synergy 構成の [Pins] タブでオプションとして使用できます。[Pins] タブの「Select pin configuration」の下にあるドロップダウンリストで、そのファイル名を探します。
4) [Generate data] チェックボックスを選択し、ピン構成名を入力します。ピン構成のドロップダウンの右側にチェックボックスとテキストエントリが見つかります。
5) 実行プロファイルまたはローパワープロファイルのいずれかに対して、目的に合わせてピンを構成します。
6) プロジェクト構成を保存して、プロジェクトコンテンツを生成します。
7) 生成されたピン構成を表示するには、ファイル {project_directory}/src/synergy_gen/pin_data.c で、テキストボックスに入力した名前と同じ名前の ioport_cfg_t 構造体を探します。
8) ioport_cfg_t 構造体の名前を実行プロファイルまたはローパワープロファイルのピン構成テーブルエントリの 1 つに追加します。

パワープロファイル V2 実行プロファイルの場合：



パワープロファイル V2 ローパワープロファイルの場合：



パワープロファイル V2 フレームワークの割り込みの構成

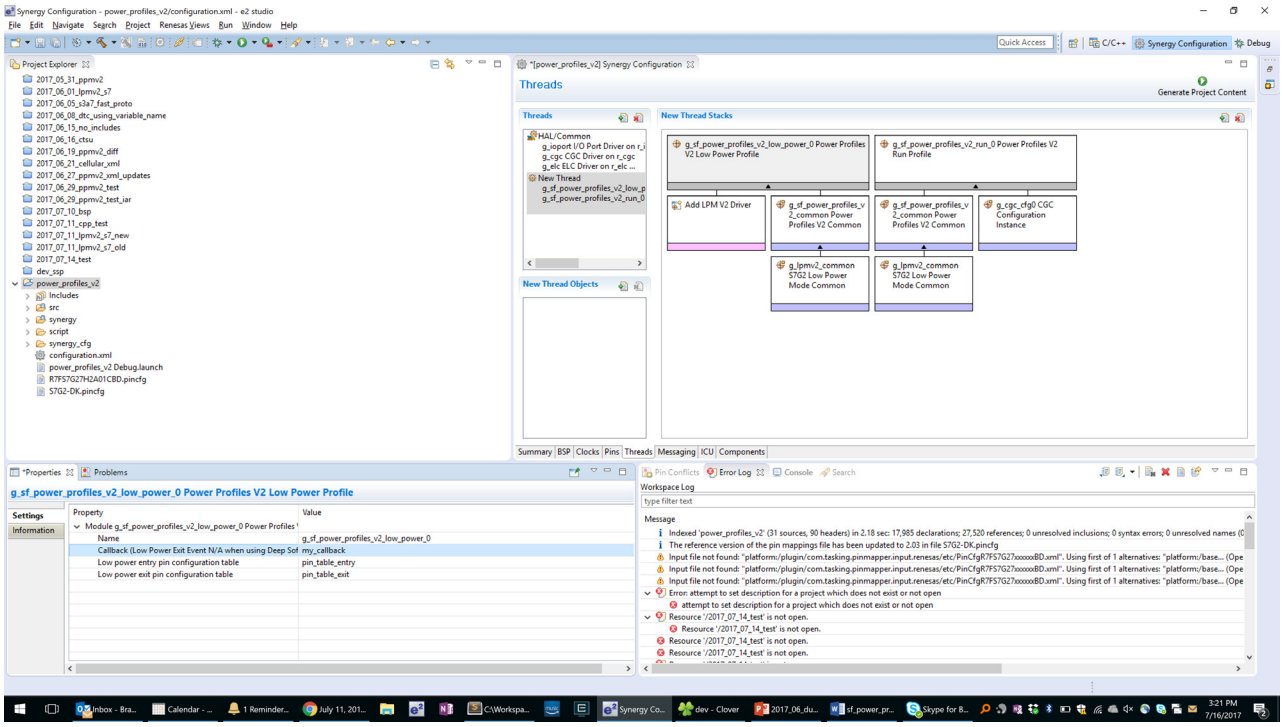
どの割り込みでもスリープモード中に MCU を復帰できるものの、パワープロファイル V2 フレームワークは割り込みを直接には使用しません。割り込みは、LPM V2 ドライバー構成を通じて処理されます。

パワープロファイル V2 のコールバックの構成

パワープロファイル V2 ローパワープロファイルは、低消費電力モードの開始前および低消費電力モードからの復帰後にアプリケーションに通知できます。プロトタイプは、/src/synergy_gen/hal_data.c または /src/synergy_gen/<スレッド名 >.c.

にあります。

次のように、パワープロファイル V2 ローパワープロファイルによって使用されるコールバックを指定します。



ローパワー モジュール パラメーターの設定

LPM V2 の使用方法の詳細な説明については、LPM V2 の使用上の注意を参照してください。

パワープロファイル V2 フレームワークモジュールの動作に関する重要な注意事項と制限事項

- LPM V2 ドライバーインスタンスはパワープロファイル V2 アプリケーションを作成するために必要です。CGC ドライバーは、デフォルトで Synergy プロジェクトに含まれています。パワープロファイル V2 runApply() 関数を使用するには、CGC クロック構成の作成用に CGC ドライバーの別のインスタンスが必要です。CGC ドライバーはすべての Synergy プロジェクトにデフォルトで含まれているため、このことによってプロジェクトのコードサイズが増えることはありません。
- I/O ポートのピン構成は、I/O ポートドライバーのインスタンスを追加せずに作成できます。
- ThreadX と併用した場合、このフレームワークはミューテックスのような ThreadX に固有のオブジェクトを使用します。ThreadX を利用した動作はオプションです。
- パワープロファイル V1 とパワープロファイル V2 は、同じプロジェクト内では使用できません。すべての新規プロジェクトでは、アプリケーションでパワープロファイル V2 を使用することをお勧めします。

パワープロファイル V2 フレームワークでは、MCU ペリフェラルの開始や停止を処理しません。

パワープロファイル V2 フレームワークの open 関数は、プロジェクトが ThreadX を使用していない場合は main の前に自動的にはコールされません。g_common_init()or をコールするか sf_power_profiles_v2_api_t::open API を明示的にコールして初期化を明示的に行う必要があります。これはパワープロファイル V2 の制限事項ではなく、RTOS なしで任意のフレームワークモジュールを使用できるようにしていることによるものです。

```
#include "hal_data.h"
void hal_entry(void)
{
```

```
    g_common_init();
    g_sf_power_profiles_v2_common.p_api->runApply(
        g_sf_power_profiles_v2_common.p_ctrl,
        &g_sf_power_profiles_v2_run_0);
    g_sf_power_profiles_v2_common.p_api->lowPowerApply(
        g_sf_power_profiles_v2_common.p_ctrl,
        &g_sf_power_profiles_v2_low_power_0);
    g_sf_power_profiles_v2_common.p_api->close(g_sf_power_profiles_v2_common.p_c
trl);
}
```

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.28.4 アプリケーションへのパワープロファイル V2 フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにパワープロファイル V2 フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初の方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

パワープロファイル V2 フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。(パワープロファイル V2 フレームワークモジュールのデフォルト名は sf_power_profiles_v2_0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

パワープロファイル V2 フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_power_profiles_low_power_0 Power Profiles V2 Low Power Profile	Threads	New Stack > Framework > Services > Power Profiles V2 Low Power Profile
g_sf_power_profiles_run_0 Power Profiles V2 Run Profile	Threads	New Stack > Framework > Services > Power Profiles V2 Run Profile

次の図に示すように、sf_power_profiles_v2 のパワープロファイル V2 フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

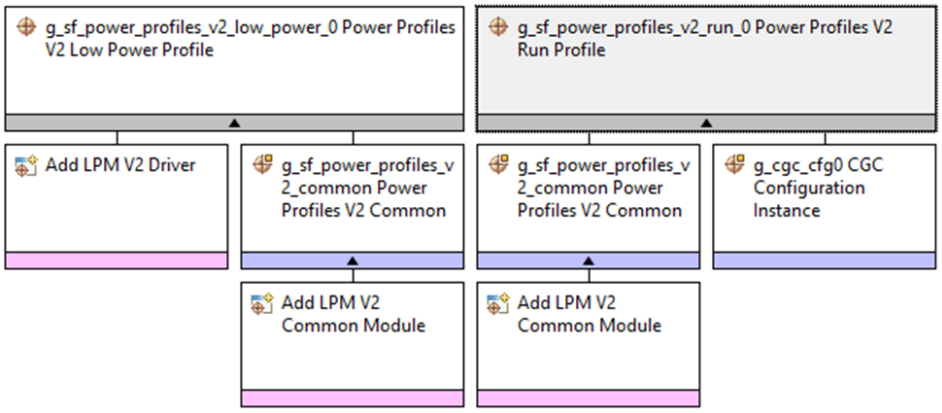


図 115:パワープロファイル V2 フレームワークモジュールのスタック

4.1.28.5 パワープロファイル V2 フレームワークモジュールの構成

ユーザーは必要な動作に合わせてパワープロファイル V2 フレームワークモジュールを構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成テーブルの設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

パワープロファイル V2 フレームワーク、実行プロファイル、ローパワープロファイルには 2 つの異なるスタック選択項目があります。それぞれの構成設定について、以降のセクションで別々に説明します。

パワープロファイル V2 実行プロファイルの構成

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [Properties] セクションのすべての設定を示します。

パワープロファイル V2 実行プロファイルの構成設定

ISDE Property	Value	Description
Name	g_sf_power_profiles_v2_run_0	Module name

ISDE Property	Value	Description
Pin configuration table	NULL	Pin configuration table selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

パワープロファイル V2 共通の構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_sf_power_profiles_v2_common	Module name

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

CGC 構成インスタンスの構成設定

ISDE Property	Value	Description
Name	g_cgc_cfg0	Module name
System Clock	HOCO, MOCO, LOCO, Main Oscillator, Sub Clock, PLL Default: HOCO	System clock selection
LOCO State Change		LOCO state change selection
MOCO State Change	None, Start, Stop Default: None	MOCO state change selection

ISDE Property	Value	Description
HOCO State Change	None, Start, Stop Default: None	HOCO state change selection
Sub-Clock State Change	None, Start, Stop Default: None	Sub-clock state change selection
Main Clock State Change	None, Start, Stop Default: None	Main clock state change selection
PLL State Change	None, Start, Stop Default: None	PLL state change selection
PLL Source Clock	HOCO, MOCO, LOCO, Main Oscillator, Sub Clock, PLL Default: HOCO	PLL source clock selection
PLL Divisor	1, 2, 3, 4 Default: 1	PLL divisor seiection
PLL Multiplier	10.0, 10.5, 11.0, 11.5, 12.0, 12.5, 13.0, 13.5, 14.0, 14.5, 15.0, 15.5, 16.0, 16.5, 17.0, 17.5, 18.0, 18.5, 19.0, 19.5, 20.0, 20.5, 21.0, 21.5, 22.0, 22.5, 23.0, 23.5, 24.0, 24.5, 25.0, 25.5, 26.0, 26.5, 27.0, 27.5, 28.0, 28.5, 29.0, 29.5, 30.0, 31.0 Default: 10.0	PLL multiplier selection
PCLKA Divisor	1, 2, 4, 8, 16, 64 Default: 1	PCLKA divisor selection

ISDE Property	Value	Description
PCKLB Divisor	1, 2, 4, 8, 16, 64 Default: 1	PCKLB divisor selection
PCLKC Divisor	1, 2, 4, 8, 16, 64 Default: 1	PCLKC divisor selection
PCLKD Divisor	1, 2, 4, 8, 16, 64 Default: 1	PCLKD divisor selection
BCLK Divisor	1, 2, 4, 8, 16, 64 Default: 1	BCLK divisor selection
FCLK Divisor	1, 2, 4, 8, 16, 64 Default: 1	FCLK divisor selection
ICLK Divisor	1, 2, 4, 8, 16, 64 Default: 1	ICLK divisor selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

低消費電力モード共通の構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_lpmv2_common	Module name

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

パワープロファイル V2 ローパワープロファイルの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

パワープロファイル V2 ローパワープロファイルの構成設定

ISDE Property	Value	Description
Name	g_sf_power_profiles_v2_low_power_0	Module name
Callback (Low Power Exit Event N/A when using Deep Software Standby)	NULL	Callback selection
Low power entry pin configuration table	NULL	Low power entry pin configuration table selection
Low power exit pin configuration table	NULL	Low power exit pin configuration table selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

低消費電力モードのディープスタンバイの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking
Name	g_lpmv2_deep_standby	Module name
Output port state in standby and deep standby, applies to address output, data output, and other bus control output pins	High impedance state, No change Default: No change	Output port state selection

ISDE Property	Value	Description
Maintain or reset the IO port states on exit from deep standby mode	Maintain the IO port states, Reset the IO port states Default: Maintain the IO port states	Maintain or reset the IO port states selection
Internal power supply control in deep standby mode	Maintain the internal power supply, Cut the power supply to standby RAM, low-speed on-chip oscillator, AGTn, and USPFS/HS resume detecting unit, Cut the power supply to LVDn, standby RAM, low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit Default: Maintain the internal power supply	Internal power supply control selection
IRQ0-15	Enabled, Disabled Default: Disabled	IRQ0-15 selection
IRQ0-15 Edge	Disabled, Rising Edge, Falling Edge Default: Disabled	IRQ0-15 Edge selection
LVD1	Enabled, Disabled Default: Disabled	LVD1 selection
LVD1 Edge	Disabled, Rising Edge, Falling Edge Default: Disabled	LVD1 Edge selection
LVD2	Enabled, Disabled Default: Disabled	LVD2 selection

ISDE Property	Value	Description
LVD2 Edge	Disabled, Rising Edge, Falling Edge Default: Disabled	LVD2 Edge selection
RTC Interval	Enabled, Disabled Default: Disabled	RTC Interval selection
RTC Alarm	Enabled, Disabled Default: Disabled	RTC Alarm selection
NMI	Enabled, Disabled Default: Disabled	NMI selection
NMI Edge	Disabled, Rising Edge, Falling Edge Default: Disabled	NMI Edge selection
USBFS	Enabled, Disabled Default: Disabled	USBFS selection
UBSHS	Enabled, Disabled Default: Disabled	UBSHS selection
AGT1	Enabled, Disabled Default: Disabled	AGT1 selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

低消費電力モードのスリープの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking
Name	g_lpmv2_sleep0	Module name

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

低消費電力モードのスタンバイの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking
Name	g_lpmv2_standby0	Module name
Choose the low power mode	Standby, Standby with snooze Enabled Default: Standby	Low power mode selection
Output port state in standby and deep standby, applies to address output, data output, and other bus control output pins	High impedance state, No change Default: No change	Output port state selection
IRQ1-15	Enabled, Disabled Default: Disabled	IRQ1-15 selection
IWDT	Enabled, Disabled Default: Disabled	IWDT selection

ISDE Property	Value	Description
Key Interrupt	Enabled, Disabled Default: Disabled	Key Interrupt selection
LVD1 Interrupt	Enabled, Disabled Default: Disabled	LVD1 Interrupt selection
LVD2 Interrupt	Enabled, Disabled Default: Disabled	LVD2 Interrupt selection
Analog Comparator High-speed 0 Interrupt	Enabled, Disabled Default: Disabled	Analog Comparator High-speed 0 Interrupt selection
RTC Alarm	Enabled, Disabled Default: Disabled	RTC Alarm selection
RTC Period	Enabled, Disabled Default: Disabled	RTC Period selection
USB High-speed	Enabled, Disabled Default: Disabled	USB High-speed selection
USB Full-speed	Enabled, Disabled Default: Disabled	USB Full-speed selection
AGT1 underflow	Enabled, Disabled Default: Disabled	AGT1 underflow selection

ISDE Property	Value	Description
AGT1 Compare Match A	Enabled, Disabled Default: Disabled	AGT1 Compare Match A selection
AGT1 Compare Match B	Enabled, Disabled Default: Disabled	AGT1 Compare Match B selection
12C 0	Enabled, Disabled Default: Disabled	12C 0 selection
Snooze Entry Source	RXD0 falling edge, IRQ0-IRQ15, KINT, ACMPHS0, RTC Alarm, RTC Period, AGT1 Underflow, AGT1 Compare Match A, AGT1 Compare Match B Default: RXD0 falling edge	Snooze Entry Source selection
AGT1 Underflow	Enabled, Disabled Default: Disabled	AGT1 Underflow selection
DTC Transfer Completion	Enabled, Disabled Default: Disabled	DTC Transfer Completion selection
DTC Transfer Completion Negated Signal	Enabled, Disabled Default: Disabled	DTC Transfer Completion Negated Signal selection
ADC0 Compare Match	Enabled, Disabled Default: Disabled	ADC0 Compare Match selection

ISDE Property	Value	Description
ADC0 Compare Mismatch	Enabled, Disabled Default: Disabled	ADC0 Compare Mismatch selection
ADC1 Compare Match	Enabled, Disabled Default: Disabled	ADC1 Compare Match selection
ADC1 Compare Mismatch	Enabled, Disabled Default: Disabled	ADC1 Compare Mismatch selection
SCI0 Address Match	Enabled, Disabled Default: Disabled	SCI0 Address Match selection
DTC state in Snoooze Mode	Enabled, Disabled Default: Disabled	DTC state in Snoooze Mode selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

パワープロファイル V2 共通の構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_sf_power_profiles_v2_common	Module name

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

低消費電力モード共通の構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_lpmv2_common	Module name

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

パワープロファイル V2 フレームワークモジュールのクロック構成

パワープロファイル V2 フレームワークは、固有のクロック設定を必要としません。

パワープロファイル V2 フレームワークモジュールのピン構成

アプリケーションは、低消費電力モード中に I/O ポート状態を維持することもできます。

4.1.28.6 アプリケーションでのパワープロファイル V2 フレームワークモジュールの使用

アプリケーションでパワープロファイル V2 フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) ローパワープロファイルに構成されたコールバック関数の本体を定義します。MCU が低消費電力モードに移行する直前と、低消費電力モードから復帰した直後に、コールバック関数によってアプリケーションに通知が行われます。コールバックの使用は任意ですが、パワープロファイル V2 のプロパティにコールバックを 1 つ定義している場合は、それに対する定義が必要です。
2) ThreadX が使用されている場合は、ユーザーアプリケーションコードに到達する前に、Synergy で生成されるコードによってパワープロファイル V2 フレームワークの sf_power_profiles_v2_api_t::open 関数がコールされます。ThreadX が使用されていない場合は、アプリケーションが open 関数を呼び出す必要があります。
3) runApply() 関数を使用して任意の時点で実行プロファイルを適用します。runApply() 関数は実行プロファイルを第 2 パラメータとして受け入れます。パラメータは任意の有効な実行プロファイルにすることができ、アプリケーションで実行プロファイル間を簡単に切り替えることができます。
Profiles.g_sf_power_profiles_v2_common.p_api->runApply(g_sf_power_profiles_v2_common.p_ctrl,&g_sf_power_profiles_v2_run_0);
4) lowPowerApply() 関数を使用してローパワープロファイルを適用します。lowPowerApply() 関数はローパワープロファイルを第 2 パラメータとして受け入れます。パラメータは任意の有効なローパワープロファイルにすることができ、アプリケーションでローパワープロファイル間を簡単に切り替えることができます。
Profiles.g_sf_power_profiles_v2_common.p_api->lowPowerApply(g_sf_power_profiles_v2_common.p_ctrl,&g_sf_power_profiles_v2_low_power_0);
5) sf_power_profiles_v2_api_t::close 関数をコールしてフレームワークを閉じます。
[Optional]g_sf_power_profiles_v2_common.p_api->close(g_sf_power_profiles_v2_common.p_ctrl);

これらの一般的な手順を、次の図の通常の動作フローに示します。

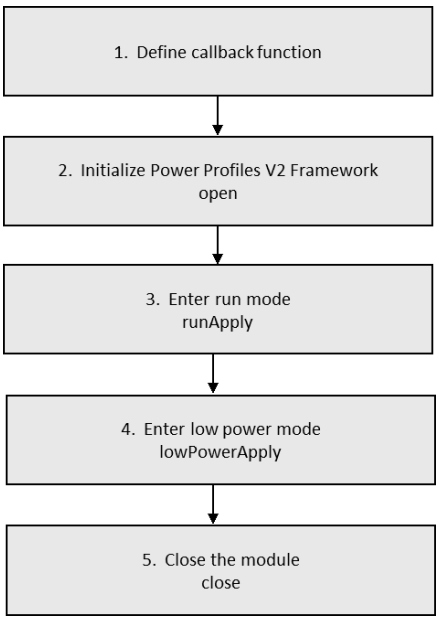


図 116:通常のパワープロファイル V2 フレームワークモジュールアプリケーションのフロー図

4.1.29 SPI フレームワーク

SPI フレームワークモジュールは ThreadX 対応のフレームワーク API を提供し、（チップ選択処理とそのレベルのアクティブ化を含め）SPI バス上の複数の SPI ペリフェラルの統合と同期を処理します。SPI フレームワークを使用すると、1 つ以上の SPI バスを作成し、複数の SPI 周辺機能をそれぞれの SPI バスに接続できます。SPI フレームワークモジュールは、単一のインタフェースを使用して、SCI SPI ドライバーと RSPI ドライバーの両方にアクセスします。SPI フレームワークモジュールは、Synergy MCU 上の SCI および RSPI 周辺機能を使用します。

4.1.29.1 SPI フレームワークモジュールの特長

SPI フレームワークモジュールは、SPI モードの SCI を（SCI 共通ローレベルモジュールとともに）使用するか RSPI ローレベルドライバーモジュールを使用して、Synergy マイクロコントローラ上の SPI 周辺機能と通信します。

- バス上の複数のデバイスをサポート
- モジュールの初期化、転送、クローズ用の高レベルの API を提供
- 同期転送をサポート
- チップ選択操作をサポート
- バスロックをサポート

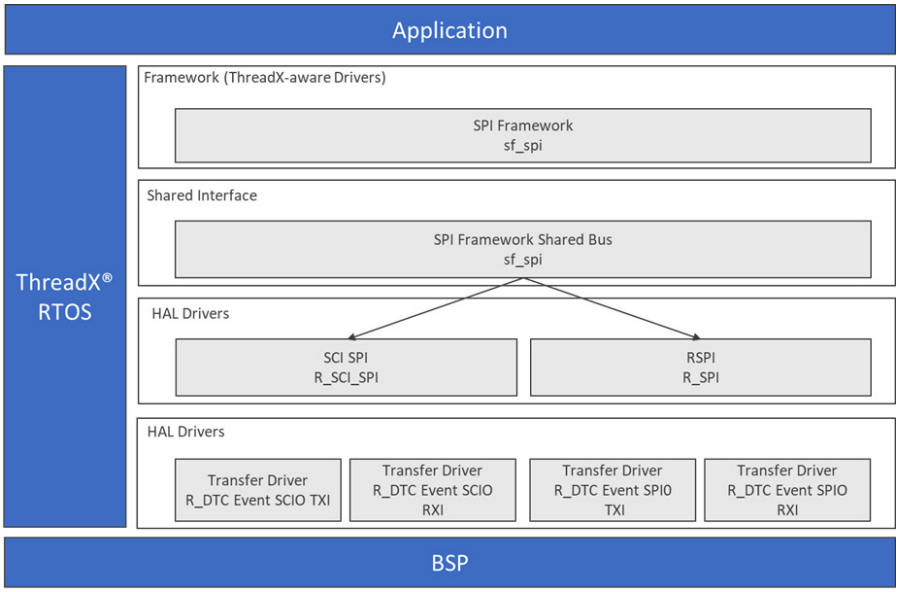


図 117:SPI フレームワークモジュールのブロック図

4.1.29.2 SPI フレームワークモジュール API の概要

SPI フレームワークモジュールは、オープン、クローズ、リード、ライト、その他の有用な機能の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

SPI フレームワークモジュール API の要約

Function Name	Example API Call and Description
open	g_sf_spi_device.p_api->open(g_sf_spi_device.p_cntl, g_sf_spi_device.p_cfg); Open a designated SPI device on a bus.
read	g_sf_spi_device.p_api->read(g_sf_spi_device.p_cntl, dst8, length, SPI_BIT_WIDTH_8_BITS, TX_WAIT_FOREVER); Receive data from SPI device.

Function Name	Example API Call and Description
write	g_sf_spi_device.p_api->write(g_sf_spi_device.p_cntl, src8, length, SPI_BIT_WIDTH_8_BITS, TX_WAIT_FOREVER); Transmit data to SPI device.
writeRead	g_sf_spi_device.p_api->writeRead (g_sf_spi_device.p_cntl, &source, &destination, length, SPI_BIT_WIDTH_8_BITS, TX_WAIT_FOREVER); Simultaneously transmits data to an SPI device while receiving data from an SPI device (full duplex). The writeread API gets a mutex object, handles the SPI data transmission at SPI HAL layer, and receives data from the SPI HAL layer. The API uses the event flag wait to synchronize to completion of data transfer.
close	g_sf_spi_device.p_api->close(g_sf_spi_device.p_cntl); Disable the SPI device designated by the control handle and close the RTOS services used by the bus, if no devices are connected to the bus. This function removes power to the SPI channel designated by the handle and disables the associated interrupts.
lock	g_sf_spi_device.p_api->lock(g_sf_spi_device.p_cntl); Lock the bus for a device. The locking allows devices to reserve a bus to themselves for a given period of time (such as between lock and unlock). This allows devices to complete several reads and writes on the bus without an interrupt.

Function Name	Example API Call and Description
unlock	g_sf_spi_device.p_api->unlock(g_sf_spi_device.p_cntl); Unlock the bus for a particular device and make the bus usable for other devices.
version	g_sf_spi_device.p_api->version(&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function completed successfully
SSP_ERR_INVALID_MODE	Invalid mode
SSP_ERR_INVALID_CHANNEL	Invalid channel
SSP_ERR_IN_USE	In-use error
SSP_ERR_INVALID_ARGUMENT	Invalid argument
SSP_ERR_QUEUE_UNAVAILABLE	Queue unavailable
SSP_ERR_INVALID_POINTER	Invalid pointer
SSP_ERR_INTERNAL	Internal error
SSP_ERR_TRANSFER_ABORTED	Transfer aborted
SSP_ERR_MODE_FAULT	Mode fault
SSP_ERR_READ_OVF	Read overflow
SSP_ERR_PARITY	Parity error
SSP_ERR_OVERRUN	Overrun error

Name	Description
SSP_ERR_UNDEF	Unknown error
SSP_ERR_TIMEOUT	Timeout error
SSP_ERR_NOT_OPEN	Device not opened

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.29.3 SPI フレームワークモジュールの動作の概要

SPI フレームワークモジュールは、SSP の階層化されたドライバーアーキテクチャに準拠します。また、SPI モジュール上の SCI または RSPI モジュールを使用して、Synergy マイクロコントローラ上の SPI 周辺機器と通信します。

同じバス上での複数のスレーブデバイスの接続

SPI フレームワークモジュールは、バスとバス上のデバイスによるアーキテクチャを使用します。ローレベルドライバーには一度に 1 つのデバイスのみを構成でき、その他のデバイスは必要に応じてリード動作またはライト動作時に再構成されます。ローレベルドライバーはバスがロックされていないときにのみ再構成できます。どのスレーブデバイスも、接続先のバスに関連付けられており、他のすべてのスレーブデバイスとバスを共有します。

ユーザーは、SPI フレームワーク共有バスと、バスに接続されている各 SPI フレームワークモジュールのローレベル SPI HAL レイヤを構成する必要があります。ユーザーは、複数のデバイスを同じバス上で構成するときに既存のフレームワーク共有バスモジュールを追加することができます。各 SPI フレームワークモジュールは、ISDE コンフィギュレータで一意の名前を使用して構成する必要があります。

共通の開始および停止手順が、すべての SPI データ転送操作に使用されます（spi_api_t::read、spi_api_t::write、および spi_api_t::writeRead）。開始プロセスの際に、SPI フレームワークモジュールは再構成が必要かどうかを確認します。バスがロックされていない場合、チップ選択は、転送開始処理の中でアサートされ、転送終了処理の中でアサート解除されます。ユーザーは、チップ選択 IO ピンとチップ選択アクティブレベルを構成する必要があります。

バスロック

SPI フレームワークモジュールではバスロック機能がサポートされるため、特定のスレーブペリフェラル用にバスをロックできます。ロックすると、ロックコマンドからロック解除コマンドの間までの期間、スレーブデバイスがバスを予約できるようになります。これにより、（一部の状況に対応して）デバイスが複数のリードおよびライトを中断することなく完了できるようになります。これにより、一部のニーズに対応して、デバイスが複数の読み書き操作を中断することなく完了できるようになります。ロックとロック解除の間にライトとリードを行っても、チップ選択ラインは変更されません。

SPI フレームワークモジュールの動作に関する重要な注意事項と制限事項

- 複数の SPI デバイスが共通バスを共有するように構成できます。SPI フレームワークバスモジュールを構成したら、各種の SPI 周辺機器（デバイス）をそのバスに接続できます。
- バスに接続している SPI デバイスごとに、1 つの SPI HAL モジュール（新規または共有）と 1 つの SPI フレームワークデバイスモジュールを追加する必要があります。
- フレームワークによって内部的に処理されるため、ユーザー定義コールバックは不要です。
- 割り込みを異なる優先度に設定すると、適切に動作しなくなる可能性があります。
- SPI バスの互換性については、MCU の仕様に関するマニュアルを参照してください。SPI バスとのデバイスの互換性はフレームワークでチェックされないため、互換性のない SPI デバイスによって不適切な動作が発生することがあります。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.29.4 アプリケーションへの SPI フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して SPI フレームワークモジュールをアプリケーションに組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

SPI フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。(SPI フレームワークモジュールのデフォルト名は g_sf_spi_device0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

SPI フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_spi_device0 on sf_spi	Threads	New Stack> Framework> Connectivity> SPI Framework Device on sf_spi

次の図に示すように、sf_spi の SPI フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

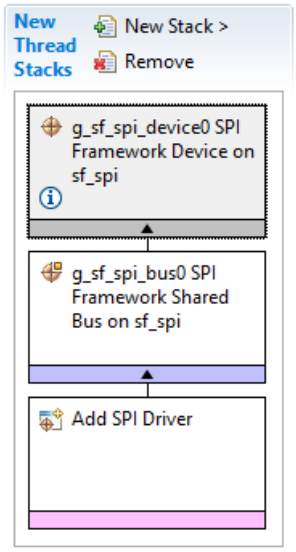


図 118:SPI フレームワークモジュールのスタック

4.1.29.5 SPI フレームワーク モジュールの設定

ユーザーは必要な動作に合わせて SPI フレームワークモジュールを構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成テーブルの設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_ spi の SPI フレームワークデバイスモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_sf_spi_device0	Module name.

ISDE Property	Value	Description
Clock Phase	Data sampling on odd edge, data variation on even edge/Data sampling on even edge, data variation on odd edge Default: Data sampling on odd edge, data variation on even edge	Select the clock phase.
Clock Polarity	Low when idle, High when idle Default: Low when idle	Select the clock polarity.
Chip Select Port	00 thru 11 Default: 00	Select GPIO port used for the chip select.
Chip Select Pin	00 thru 15 Default: 00	Select GPIO pin used for the chip select.
Chip Select Active Level	Low, High Default: Low	Polarity of the Chip Select signal, active High or Low

*注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。*デフォルト以外の設定が望ましい場合もあります。たとえば、異なるチップ選択 GPIO やレベルの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

SPI フレームワークのローレベルモジュールの構成

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、ローレベルモジュールの [properties] セクションのすべての設定を示します。

sf_spi 上の SPI フレーム共有バスの構成設定

ISDE Property	Value	Description
Name	g_sf_spi_bus0	Module name.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_rspi 上の RSPI HAL ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Name	g_spi0	Module name.
Channel	0	SCI or SPI Channel number to which the device has been connected.
Operating Mode	Master, Slave Default: Master	Configure as a Master or Slave device. Note: Current version of SSP supports only SPI Master mode.
Clock Phase	Data sampling on odd edge, data variation on even edge	Data sampling on odd or even clock edge.
Clock Polarity	Low when idle	Clock level when idle.
Mode Fault Error	Enable, Disable Default: Disable	Indicates Mode fault error (master/slave conflict) flag.

ISDE Property	Value	Description
Bit Order	MSB First, LSB First Default: MSB First	Select transmit order MSB/LSB first.
Bitrate	500000	Transmission or reception rate. Bits per second.
Callback	NULL	Optional Callback function pointer.
SPI Mode	SPI Operation, Clock synchronous operation Default: SPI Operation	Select spi or clock syn mode operation.
Slave Select Polarity(SSL0)	Active Low, Active High Default: Active Low	Select SSL0 signal polarity.
Slave Select Polarity(SSL1)	Active Low, Active High Default: Active Low	Select SSL1 signal polarity.
Slave Select Polarity(SSL2)	Active Low, Active High Default: Active Low	Select SSL2 signal polarity.
Slave Select Polarity(SSL3)	Active Low, Active High Default: Active Low	Select SSL3 signal polarity.
Select Loopback1	Normal, Inverted Default: Normal	Select the data mode for loopback 1.
Select Loopback2	Normal, Inverted Default: Normal	Select the data mode for loopback 2.

ISDE Property	Value	Description
Enable MOSI Idle State	Enable, Disable Default: Disable	Select MOSI idle fixed value and selection
MOSI Ildle State	MOSI Low, MOSI High Default: MOSI Low	Select mosi idle fixed value and selection
Enable Parity	Enable, Disable Default: Disable	Enable/disable parity
Parity Mode	Parity Odd, Parity Even Default: Parity Odd	Select parity
Select SSL(Slave Select)	SSL0, SSL1, SSL2, SSL3 Default: SSL0	Select which slave to use; 0-SSL0; 1-SSL1; 2-SSL2; 3-SSL3
Select SSL Level After Transfer	SSL Level Keep, SSL Level Do Not Keep Default: SSL Level Do Not Keep	Select SSL level after transfer completion; 0-negate; 1-keep
Clock Delay Enable	Clock Delay Enable, Clock Delay Disable Default: Clock Delay Disable	Clock delay enable selection
Clock Delay Count	Clock Delay 1 thru 8 RSPCK Default: Clock Delay 1 RSPCK	Clock delay count selection

ISDE Property	Value	Description
SSL Negation Delay Enable	Negation Delay Enable, Negation Delay Disable Default: Negation Delay Disable	SSL negation delay enable selection
Negation Delay Count	Negation Delay 1 thru 8 RSPCK Default: Negation Delay 1 RSPCK	Negation delay count selection
Next Access Delay Enable	Next Access Delay Enable, Next Access Delay Disable Default: Next Access Delay Disable	Next access delay enable selection
Next Access Delay Count	Next Access Delay 1 thru 8 RSPCK Default: Next Access Delay 1 RSPCK	Next access delay count selection
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Receive interrupt priority selection
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit interrupt priority selection
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit interrupt priority selection

ISDE Property	Value	Description
Error Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Error interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SPI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	
Destination Pointer	NULL	Destination pointer selection

ISDE Property	Value	Description
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SPI0 TXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SPI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection

ISDE Property	Value	Description
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SPI0 RXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sci_spi 上の SPI ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.

ISDE Property	Value	Description
Name	g_spi0	Module name
Channel	0	SCI or SPI Channel number to which the device has been connected.
Operating Mode	Master, Slave Default: Master	Configure as a Master or Slave device. Note: Current version of SSP supports only SPI Master mode.
Clock Phase	Data sampling on odd edge, data variation on even edge	Data sampling on odd or even clock edge.
Clock Polarity	Low when idle	Clock level when idle.
Mode Fault Error	Enable, Disable Default: Disable	Indicates Mode fault error (master/slave conflict) flag.
Bit Order	MSB First, LSB First Default: MSB First	Select transmit order MSB/LSB first
Bitrate	100000	Transmission or reception rate. Bits per second.
Bit Rate Modulation Enable	Enable, Disable Default: Enable	Bitrate Modulation Function enable or disable
Callback	NULL	Optional Call back function pointer.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Bitrate Modulation Function enable or disable. Note: This is applicable only for SCI SPI.

ISDE Property	Value	Description
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit interrupt priority selection.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit end interrupt priority selection
Error Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Error interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection

ISDE Property	Value	Description
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency seleciton
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SCI0 TXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SCI0 RXI	Activation source selection
Auto Enable	False	Auto enable selection

ISDE Property	Value	Description
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

SPI フレームワークモジュールのクロック構成

SCI ペリフェラルモジュールは PCLKB をそのクロックソースとして使用します。PCLKB 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

SPI フレームワークモジュールのピン構成

SPI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

注 : 一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号が決まり、それに従って必要な MCU ピンが決定されます。

SPI フレームワークモジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SCI	Pins	Select Peripherals > Connectivity: SCI> SCI1
RSPI	Pins	Select Peripherals > Connectivity: SPI > SPI0

注 : 選択シーケンスでは、SCI1 および SPI0 がドライバーに必要なハードウェアターゲットであることを想定しています。

SPI フレームワークモジュールのピン構成設定

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Asynchronous UART, Synchronous UART, Simple I2C, Simple SPI, SmartCard Default: Disabled	Select Simple SPI as the Operation Mode for SPI on SCI
CTS0_RTS0_SS0	None, P103, P413 Default: None	SS0 Pin selection
RXD0_SCL0_MISO0	None, P100, P410 Default: None	MISO0 Pin selection
SCK0	None, P102, P412 Default: None	SCK0 Pin selection
TXD1_SDA1_MOSI0	None, P213, P709 Default: None	MOSI0 Pin selection

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

SPI フレームワークモジュールの追加設定

外部チップ選択が使用されている場合は、チップ選択ピンを GPIO 出力として構成します。

4.1.29.6 アプリケーションでの SPI フレームワークモジュールの使用

SPI フレームワークモジュールの共通アプリケーションには、単一バス上に複数のスレーブデバイスが必要です。この共通アプリケーションの実装については、以下で説明しています。2 番目の実装では、それぞれに 2 つのスレーブデバイスが接続された 2 つのバスを示しています。

単一共有バス上の 2 つのスレーブデバイスに対する実装手順

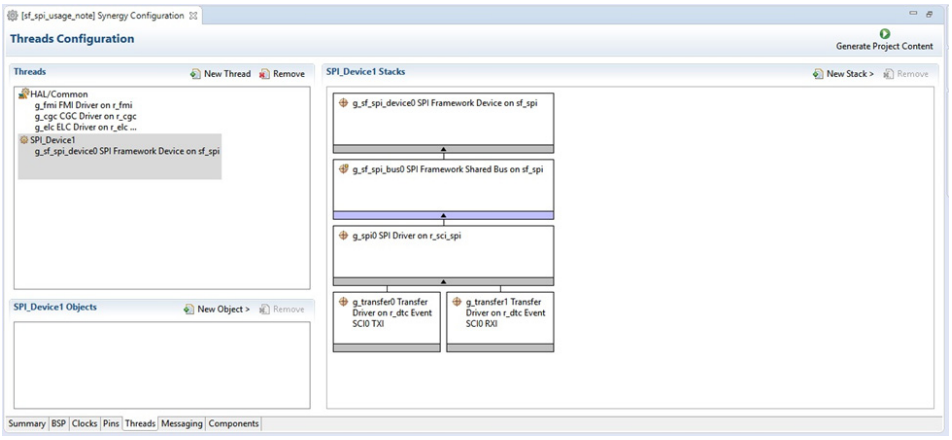
SPI フレームワークモジュールを使用して複数のスレーブデバイスを接続した単一のバスを作成するには、2 つのスレッドスタックを作成し、それぞれが I2C フレームワークインスタンスを使用するようにします。これらのインスタ

ンスは、同じ共有バスインスタンスを使用します。以下の手順に従って、SSP コンフィギュレータでこの操作を行う方法を確認してください。

注 : 以下の手順は、SSP 開発環境の使用について、ある程度習熟していることを前提としています。以下の手順でわかりにくい点がある場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を読んで、SSP 開発環境に習熟するようにしてください。

1）最初の SPI フレームワークデバイスモジュールを新しいスレッドまたは既存のスレッドに追加します。これにより、SPI マスタスタックが作成されます。sf_spi 上の共有バスが、I2C ドライバーとともに追加されます。SPI ドライバーは、r_rspi または r_sci_spi. 上での実装のために選択できます。デフォルトで DTC 転送ドライバーも追加されます。このドライバーは、代わりに CPU 転送モードが必要な場合には削除できます。

結果のモジュールスタックを次の図に示します。図の下の表に、構成設定の例を示します。



スレーブデバイス #1 で使用する重要な最初のスレッドスタックモジュールの構成設定の例は、以下のとおりです。

sf_spi 上の SPI フレームワークモジュールの構成設定

Property	Value	Description
Parameter Checking	Disabled	Enable or Disable Parameter Checking.
Name	g_sf_spi_device1	Give a name to identify the SPI Framework device. API, Config and Control instances will be created based on this name.
Clock Phase	Data sampling on odd edge	Specify the clock phase for data variation and data sampling
Clock Polarity	Low when idle	Select the clock polarity when clock is idle.

Property	Value	Description
Clock Select Port	01	Select GPIO port used for the chip select.
Chip Select Pin	04	Select GPIO pin used for the chip select.
Chip Select Active Level	Low	Select Polarity of the chip select signal.

sf_spi 上の SPI フレーム共有バスの構成設定

Property	Value	Description
Name	g_sf_spi_bus0	Give a name to identify the SPI Framework shared bus. This shared bus will be shared by multiple SPI Framework Devices

r_rspi 上の SPI ドライバーの構成設定

Property	Value	Description
Parameter Checking	BSP	Enable or Disable Parameter Checking.
Name	g_spi0	Give a name to identify the SPI Driver device. This will be used by Framework internally.
Channel	0	Channel number
Operating Mode	Master	Operating mode selection
Clock Phase	Data sampling on odd edge/ data variation on edge	Clock phase selection. This field will be locked as these is already set in the SPI Framework Device on sf_spi module.

Property	Value	Description
Clock Polarity	Low when idle	Clock polarity selection. This field will be locked as these is already set in the SPI Framework Device on sf_spi module.
Mode Fault Error	Enable	Mode fault error selection
Bit Order	MSB First	Bit order selection
Bitrate	500000	Bit rate selection
Callback	NULL	Callback function name
SPI Mode	SPI Operation	SPI mode selection
SPI Communication Mode	Full Duplex	SPI communication mode selection.
Slave Select Polarity (SSL0)	Active Low	Slave select polarity selection 0
Slave Select Polarity (SSL1)	Active Low	Slave select polarity selection 1
Slave Select Polarity (SSL2)	Active Low	Slave select polarity selection 2
Slave Select Polarity (SSL3)	Active Low	Slave select polarity selection 3
Select Loopback 1	Normal	Loopback 1 selection
Select Loopback 2	Normal	Loopback 2 selection
Enable MOSI Idle	Disable	Enable MOSI idle selection.
MOSI Idle State	MOSI Low	Enable MOSI idle state selection.
Enable Parity	Disable	Enable parity selection
Parity Mode	Parity Odd	Enable parity mode selection
Select SSL (Slave Select)	SSL0	Select SSL selection

Property	Value	Description
Select SSL Level After Transfer	SSL Level Keep	Select SSL level after transfer selection
Clock Delay Enable	Disable	Clock delay enable selection
Clock Delay Count	Clock Delay 1 RSPCK	Clock delay count selection
SSL Negation Delay Enable	Disable	SSL Negation Delay Enable selection.
Negation Delay Count	Clock Delay 1 RSPCK	Negation Delay Count selection
Next Access Delay Enable	Disable	Next Access Delay Enable selection
Next Access Delay Count	Clock Delay 1 RSPCK (	Next Access Delay Count selection
Receive Interrupt Priority	Priority 2	Receive interrupt priority selection.
Transmit Interrupt Priority	Priority 2	Transmit interrupt priority selection.
Transmit End Interrupt Priority	Priority 2	Transmit end interrupt priority selection.

r_sci_spi 上の SPI ドライバーの構成設定

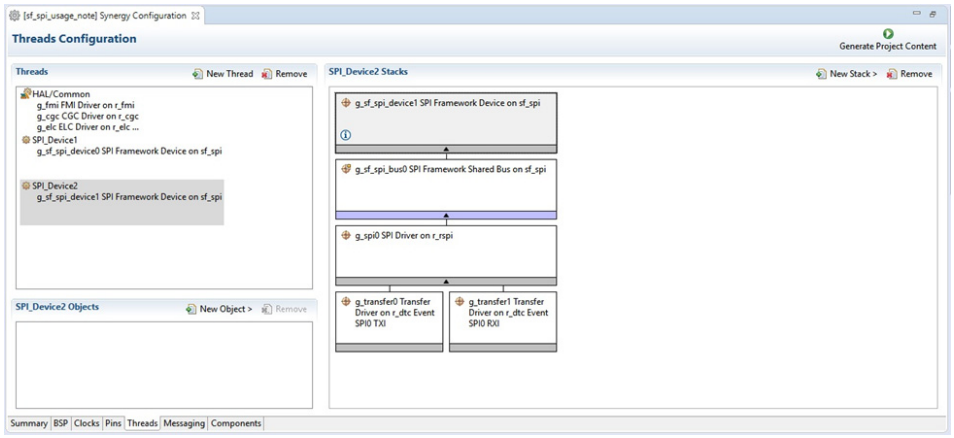
Property	Value	Description
Parameter Checking	BSP	Enable or Disable Parameter Checking.
Name	g_spi0	Give a name to identify the SPI Driver device. This will be used by Framework internally.
Channel	0	Channel number
Operating Mode	Master	Operating mode selection.

Property	Value	Description
Clock Phase	Data sampling on odd edge/ data variation on even edge	Clock phase selection. This field will be locked as these is already set in the SPI Framework Device on sf_spi module.
Clock Polarity	Standard	Clock polarity selection. This field will be locked as these is already set in the SPI Framework Device on sf_spi module.
Mode Fault Error	Disable	Mode fault error selection.
Bit Order	MSB First	Bit order selection
Bitrate	500000	Bit rate selection
Bit Rate Modulation Enable	Enable	Enables/Disable the bit rate modulation.
Callback	NULL	Callback function name. This field will be locked as callback is handled internally in the framework.
Receive Interrupt Priority	Priority 2	Receive interrupt priority selection.
Transmit Interrupt Priority	Priority 2	Transmit interrupt priority selection.
Transmit End Interrupt Priority	Priority 2	Transmit end interrupt priority selection.
Error Interrupt Priority	Priority 2	Error interrupt priority selection.

注 :DTC 構成設定は簡略化された形では示されません。

2) 2 番目の SPI フレームワークデバイスを別のスレッドに追加します。sf_spi 上の SPI フレームワーク共有バスは自動的に追加されません。追加するには、既存の共有バスを使用するオプションを選択します。このようにすると、コンフィギュレータによって、sf_spi 上の SPI フレームワーク共有バスと残りのモジュールが自動的に追加されます。最初の SPI フレームワークインスタンスの以前定義した設定と整合するよう、ローレベルモジュールが自動的に構成されます。このことにより、Clock Phase、Clock Polarity、Chip Select Pin、Chip Select Port、Chip Select Active Level の各プロパティを除き、SPI ドライバー構成は両方のデバイスで同じになります。これは、これらのプロパティは、SPI フレームワークデバイスモジュールで定義され、スレーブデバイスごとに異なる場合があるためです。

結果のモジュールスタックを次の図に示します。



2 番目のスタックの構成パラメータにおける唯一の違いは、2 番目の SPI フレームワークデバイスモジュールの名前と、非共有のスレーブ設定（Clock Phase、Clock Polarity、Clock Select Port、Chip Select Pin、Chip Select Active Level）の違いです。設定例を次の表に示します。

sf_spi 上の SPI フレームワークデバイスの構成設定（スレーブ #2）

Property	Value	Description
Parameter Checking	BSP	Enable or Disable Parameter Checking.
Name	g_sf_spi_device2	Give a name to identify the SPI Framework device. API, Config and Control instances will be created based on this name.
Clock Phase	Data sampling on odd edge/ data variation on even edge	Specify the clock phase for data variation and data sampling
Clock Polarity	High when idle	Select the clock polarity when clock is idle.
Clock Select Port	05	Select GPIO port used for the chip select.
Chip Select Pin	01	Select GPIO pin used for the chip select.
Chip Select Active Level	Low	Select Polarity of the chip select signal.

2 つの共有バス上にある 2 つのスレーブデバイスに対する実装手順

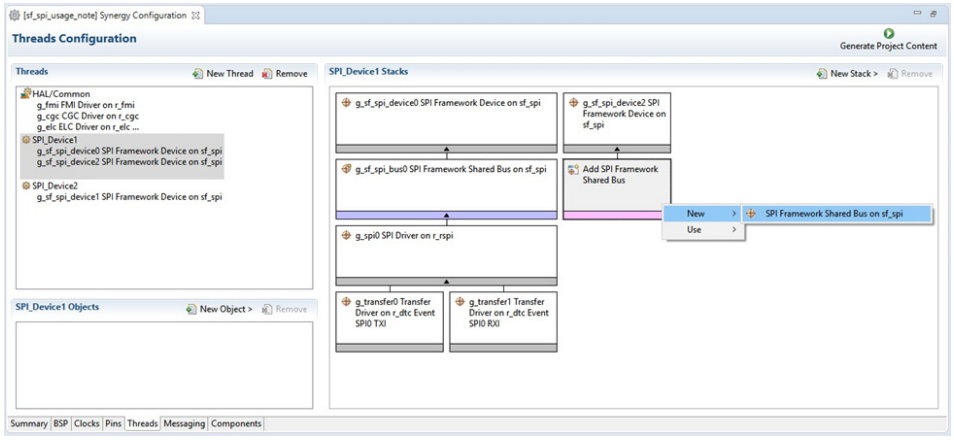
SPI フレームワークモジュールを使用して複数のスレーブデバイスを接続した単一のバスを作成するには、2 つのスレッドスタックを作成し、それぞれが I2C フレームワークインスタンスを使用するようにします。これらのインスタンスは、同じ共有バスインスタンスを使用します。以下の手順に従って、SSP コンフィギュレータでこの操作を行う方法を確認してください。

注 : 以下の手順は、SSP 開発環境の使用について、ある程度習熟していることを前提としています。以下の手順でわかりにくい点がある場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を読んで、SSP 開発環境に習熟するようにしてください。

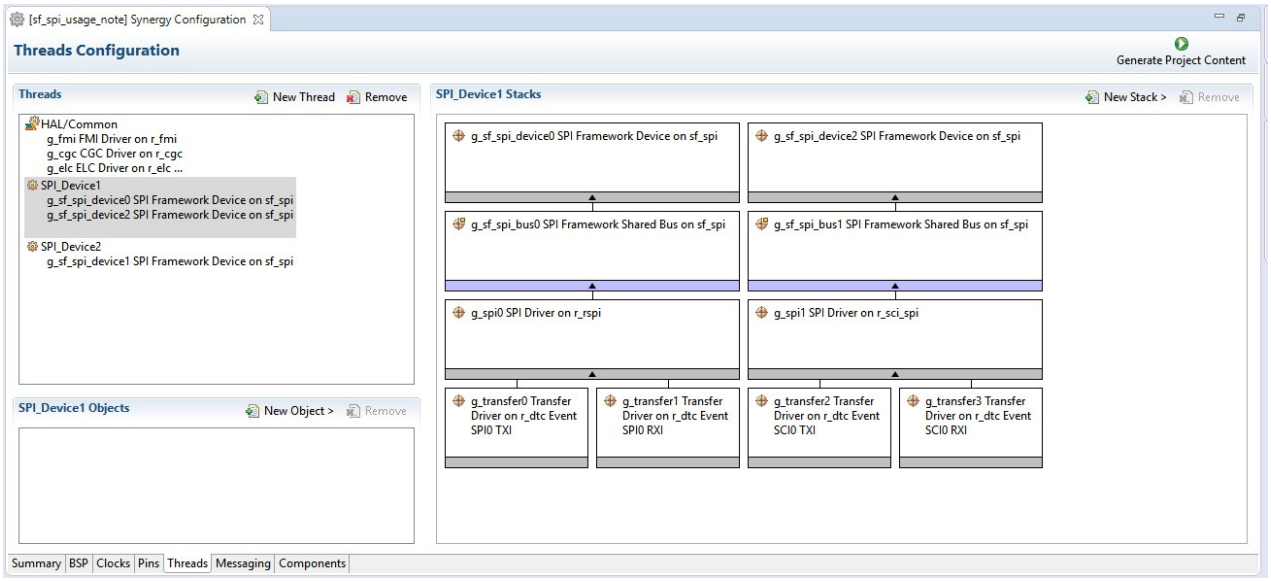
別の共有バスの追加

別の共有バスを追加するには、以下の手順に従います。前の例を使用して開始します。

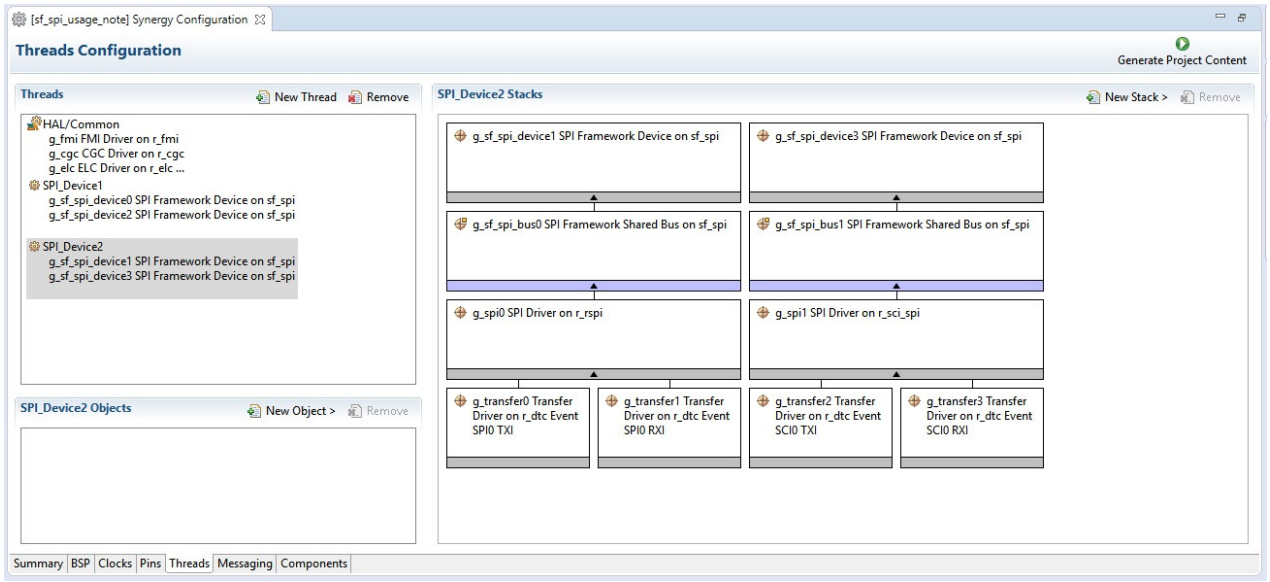
1）2 番目の共有バスを使用する SPI フレームワークモジュールは、任意のスレッドに追加できます。前の例のモジュールから順に SPI_Device1 スレッドに追加すると、モジュールスタックは以下のように表示されます。共有バスで使用できるオプションは [New] または [Use] です。



2）[New] を選択し、sf_spi モジュール上の別の SPI フレームワーク共有バスを追加します。必要に応じて、アプリケーションに対して共有バスプロパティを設定します。必要なローレベル SPI ドライバーを選択します。g_spi1 SPI ドライバーモジュールのチャネル番号は、g_spi0 SPI ドライバーモジュールのチャネル番号とは異なる必要があります。以下に結果のスレッドスタックを示します。



3）前述の手順と同じ手順を使用して、2 番目のデバイスを SPI_Device2 スレッドに追加できます。以下に結果のスレッドスタックを示します。



アプリケーションで SPI フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_spi_api_t::open API 関数を使用して SPI フレームワークデバイスモジュールを初期化します。各 SPI フレームワークデバイスモジュールは、バスに対して操作を実行する前に spi_api_t::open API 関数を 1 回以上コールする必要があります。
2) 特定の SPI フレームワークデバイスモジュール用に、sf_spi_api_t::lock API 関数を使用して、連続転送のためにバスをロックします。特定の SPI フレームワークデバイスモジュールによってバスがロックされると、バス上の他の SPI フレームワークデバイスモジュールがそのバスを使用することはできません。このことによって、バスの所有権は、ロックされたモジュールがバスを明示的にロック解除

するまで、このモジュールに残ります。その期間は、このバス上の他の SPI フレームワークデバイスモジュールが行うすべての種類の操作で、失敗のステータスが返されます。バスに対するリード / ライト動作の前に、バスをロックする必要はありません。このことはオプションです。

3) sf_spi_api_t::read API 関数を使用してデータを読み取ります。他の SPI フレームワークデバイスモジュールによってバスがすでにロックされている場合は、リード動作は失敗します。
4) sf_spi_api_t::write API 関数を使用してデータを書き込みます。他の SPI フレームワークデバイスモジュールによってバスがすでにロックされている場合は、ライト動作は失敗します。
5) sf_spi_api_t::writeRead API 関数を使用してデータのライトとリードを同時に行います。他の SPI フレームワークデバイスモジュールによってバスがすでにロックされている場合は、同時リードおよびライト動作は失敗します。
6) 同じデバイスによってバスがすでにロックされている場合は、sf_spi_api_t::unlock API 関数を使用して連続転送からバスをロック解除します。バスがロック解除されると、他の SPI フレームワークデバイスモジュールがバスを使用できます。目的のリード / ライト動作の完了後は、ロックされたバスをロック解除する必要はありません。
7) sf_spi_api_t::close API 関数を使用して SPI フレームワークデバイスモジュールを閉じます。各 SPI フレームワークデバイスモジュールは、バスに対するすべてのリード / ライト動作が終了すると、sf_spi_api_t::close API 関数をコールできます。

これらの一般的な手順を、次の図の通常の動作フローに示します。

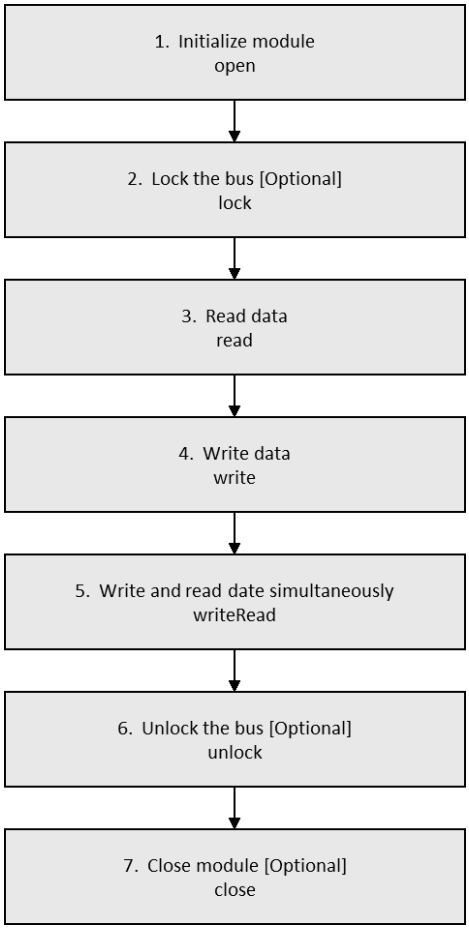


図 119:通常の SPI フレームワークモジュールアプリケーションのフロー図

4.1.30 スレッド監視フレームワーク

スレッド監視フレームワークは、ウォッチドッグタイマ（WDT）または独立ウォッチドッグタイマ（IWDT）を使用してプログラムの実行を監視するために、システム監視アプリケーションにハイレベル API を提供します。スレッド監視フレームワークは、Synergy MCU デバイス上の WDT ペリフェラルまたは IWDT ペリフェラルを使用します。

4.1.30.1 スレッド監視フレームワークモジュールの特長

- スレッド監視フレームワークインタフェースは、ウォッチドッグタイマを使用して RTOS スレッドを監視します。スレッド間メッセージングは、監視対象のスレッドが想定外の動作を行った場合、ウォッチドッグによるマイクロコントローラのリセットを強制的に実行します。
- スレッド監視は、API への変更なしに、WDT ペリフェラルまたは IWDT ペリフェラルおよび HAL モジュールを使用して、すべての Synergy デバイスをサポートするように設計されています。

- プロファイリング モードでは、登録されたスレッドのカウンターの最小値および最大値を決定できます。プロファイリングモード中は、ウォッチドッグタイマが常にリフレッシュされるため、デバイスはリセットされません。
- このフレームワークモジュールは、WDT と IWDT HAL モジュールの両方をサポートしています。

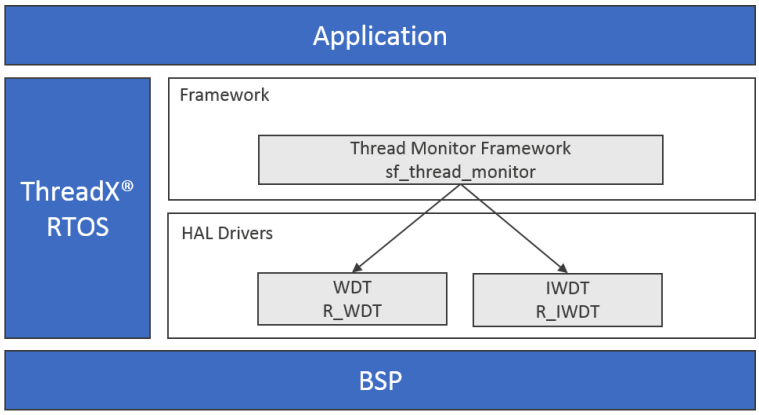


図 120:スレッド監視フレームワークモジュールのブロック図

4.1.30.2 スレッド監視フレームワークモジュール API の概要

スレッド監視フレームワークは、フレームワークのオープンとクローズ、および監視用スレッドの登録と登録解除用の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。戻りステータス値の表は API 要約表の後にあります。

スレッド監視フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_thread_monitor.p_api->open (g_sf_thread_monitor.p_ctrl,g_sf_thread_monitor.p_cfg); Configures the WDT or IWDT module. From the configuration data, the timeout period of the WDT/IWDT is determined. A thread created to monitor registered threads.
close	g_sf_thread_monitor.p_api->close (g_sf_thread_monitor.p_ctrl); Suspends the thread monitoring thread. The watchdog peripheral no longer refreshes.

Function Name	Example API Call and Description
threadRegister	g_sf_thread_monitor.p_api-> threadRegister (g_sf_thread_monitor.p_ctrl, &p_min_max_struct); Registers a thread for monitoring.
threadUnregister	g_sf_thread_monitor.p_api-> threadUnregister (g_sf_thread_monitor.p_ctrl); Removes a thread from monitoring.
countIncrement	g_sf_thread_monitor.p_api-> countIncrement (g_sf_thread_monitor.p_ctrl); Safely increments a monitored thread's count value.
versionGet	g_sf_thread_monitor.p_api-> versionGet(&version); Retrieves the API version and stores it in the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_ASSERTION	Pointer is null.
SSP_ERR_IN_USE	Thread monitor has already been opened.
SSP_ERR_INVALID_MODE	Low-level watchdog peripheral returns an error when opened.
SSP_ERR_UNSUPPORTED	Data structure could not be allocated.

Name	Description
SSP_ERR_INVALID_ARGUMENT	One or more configuration options is invalid for the low-level driver.
SSP_ERR_NOT_OPEN	sf_thread_monitor_api_t::open has either not been called or was not called successfully.
SSP_ERR_INSUFFICIENT_SPACE	Not enough entries in the threads-to-be-monitored array to add this thread. Increases the value of THREAD_MONITOR_CFG_MAX_NUMBER_OF_THREADS in sf_thread_moinitor_cfg.h

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.30.3 スレッド監視フレームワークモジュールの動作の概要

スレッド監視は、スレッドがスレッド監視にカウンタ変数と、このカウンタ変数の予測される最小値と最大値を登録する、という形で実行されます。監視されているスレッドは、動作中にカウンター変数をインクリメントします。ウォッチドッグタイムアウト期間の半分の期間で、スレッド監視は登録されたスレッドのカウンタ変数をチェックします。最小値から最大値の範囲外となった場合、ウォッチドッグ タイマーはマイクロコントローラーをリセットできるようになります。すべてが予測範囲内の場合は、ウォッチドッグ タイマーがリフレッシュされ、カウンター変数はクリアされてゼロになります。

スレッド監視フレームワークモジュールの動作に関する重要な注意事項と制限事項

次の図は、スレッド監視フレームワークモジュールの動作に関するフローチャートを示しています。

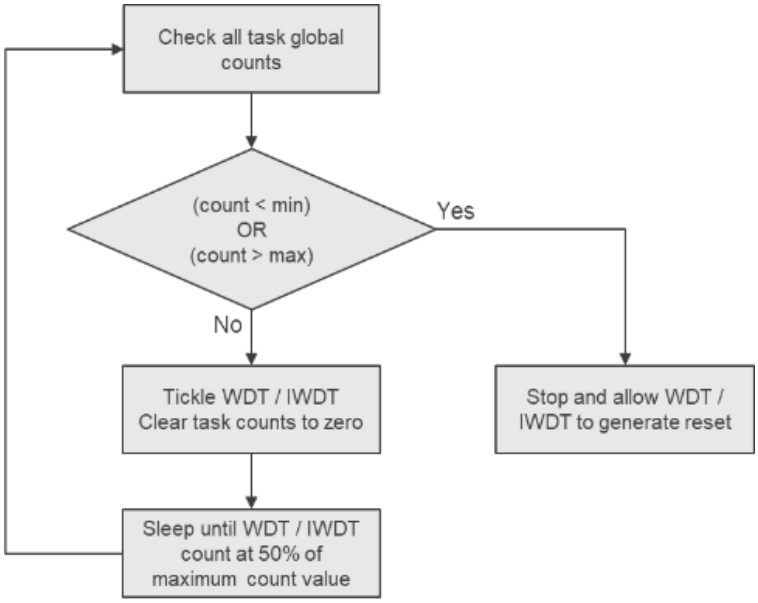


図 121:スレッド監視フレームワークの動作図

次の図は、WDT または IWDT がリフレッシュされた場合を示しています。有効なリフレッシュ期間は、カウント期間の中心にあたる 50%、すなわち 50% のカウント値の両側 25% にわたるものであることに注意してください。

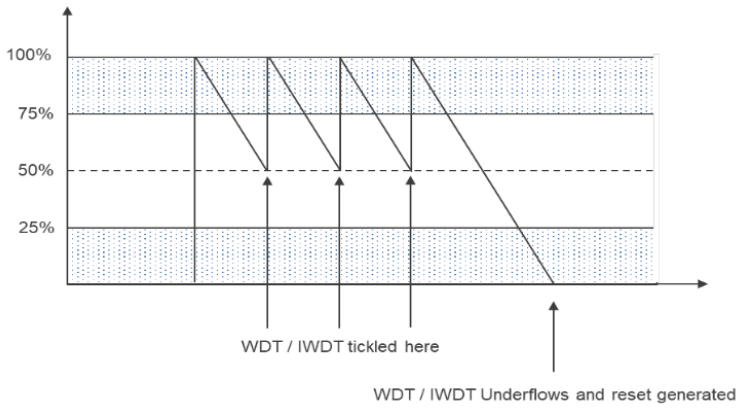


図 122:WDT/IWDT リフレッシュ動作

- IWDT には、安全性を向上させるための独自のクロックソースがあります。
- WDT はアプリケーションから開始することができます。
- スレッドがスリープモードを実行していない場合は、WDT の stop-control プロパティを [WDT Count Enabled in Low Power Mode] に設定します。スレッドが実行されていない場合（スリープ状態）、ThreadX® は WFI 命令を実行してデバイスを事実上ソフトスリープにしますが、これは WDT がカウントを停止する原因となります。

注:監視対象スレッドファイルで WDT を開いたりリフレッシュしたりしないでください。これはスレッド監視フレームワークによって自動的に行われます。

内部的に、スレッド監視フレームワークはウォッチドッグタイマのリセット期間の半分の速度で動作します。この速度により、スレッド監視フレームワークはウォッチドッグカウント値の 50% で動作するため、有効なリフレッシュウィンドウの範囲内に十分収まります。スレッド間メッセージングは、ローレベルウォッチドッグドライバーにクエリを実行することにより、内部的にリセット期間を計算します。

- スレッド監視フレームワークには、sf_thread_monitor_api_t::close API コールがあります。WDT と IWDT が使用されている場合は、これらを停止することはできません。スレッド監視フレームワークが閉じられた場合は、ウォッチドッグをリフレッシュするために他のプロビジョニングを実行する必要があり、このようなプロビジョニングを実行しなければデバイスはリセットされます。
- 特定のデバイスで JLink を使用して動作させている場合は、デバッグモードサポートが必要です。JLink デバッグハードウェアを使用していると、WDT または IWDT のカウントは実行されません。スレッド監視フレームワークスレッドは通常、WDT または IWDT カウンタと同期されますが、JLink を使用して動作している場合はこのような同期化が省略されます。
- スレッド監視フレームワークスレッドには高プライオリティ（ThreadX で低い数値）を割り当てます。スレッド監視フレームワークの実行中に遅延が発生すると、有効なリフレッシュウィンドウの範囲外でウォッチドッグがリフレッシュされることがあり、マイクロコントローラがリセットされる原因となります。スレッド監視フレームワークスレッドは長時間にわたる動作はしないため、システムの性能に影響を及ぼすことはありません。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.1.30.4 アプリケーションへのスレッド監視フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにスレッド監視フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

スレッド監視フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（スレッド監視フレームワークのデフォルト名は g_sf_thread_monitor. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

スレッド監視フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_thread_monitor0 Thread Monitor Framework	Threads	New Stack> Framework> Services> Thread Monitor Framework on sf_thread_monitor

次の図に示すように、sf_thread_monitor 上のスレッド監視フレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは

推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

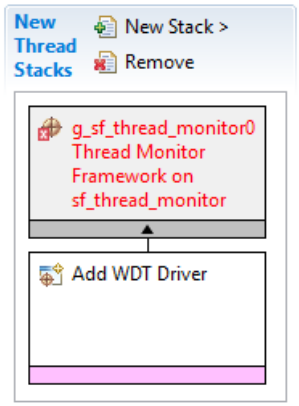


図 123:スレッド監視フレームワークモジュールのスタック

4.1.30.5 スレッド監視フレームワークモジュールの構成

ユーザーは必要な動作に合わせてスレッド監視フレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_thread_monitor でのスレッド監視フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	Enabled, Disabled, BSP Default: BSP	Controls whether to include code for API parameter checking.
Maximum Number of Monitored Threads	5	Maximum number of threads that can be monitored.

ISDE Property	Value	Description
Name	g_sf_thread_monitor0	The name of the Thread Monitor instance.
Profiling Mode	Enabled, Disabled, Default: Disabled	Whether profiling mode should be enabled.
Thread Monitor Thread Priority	1	Priority of thread monitor internal thread.
Name of generated initialization function	sf_thread_monitor_init0	Name of generated initialization function
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

スレッド監視フレームワークモジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_iwdt 上の独立ウォッチドッグタイマの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_wdt0	Module name.

ISDE Property	Value	Description
NMI Callback	NULL	Callback. A user callback function can be registered in external_irq_api_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the IRQn triggers. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_wdt 上のウォッチドッグタイマの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_wdt0	Module Name
Start Mode	Register, Auto Default: Register	Configures the start mode as register start or auto-start.
Start Watchdog After Configuration	True, False Default: True	Controls whether WDT is started during initialization

ISDE Property	Value	Description
Timeout	1024 cycles, 4096 cycles, 8192 cycles, 16384 cycles Default: 16384 cycles	WDT timeout period.
Clock Division Ratio	PCLK/4, PCLK/64, PCLK/128, PCLK/512, PCLK/2048, PCLK/8192 Default: PCLK/8192	WDT clock divider.
Window Start Position	100% (Window Position Not Specified), 75%, 50%, 25% Default: 100% (Window Position Not Specified)	Permitted refresh period start postion.
Window End Position	0% (Window Position Not Specified), 25%, 50%, 75% Default: 0% (Window Position Not Specified)	Permitted refresh period end postion.
Reset Control	Reset Outpout, NMI Generated Default: Reset Output	Select whether WDT should reset the MCU or generate an NMI.
Stop Control	WDT Count Enabled in Low Power Mode, WDT Count Disabled in Low Power Mode Default: WDT Count Disabled in Low Power Mode	Select whether the WDT should stop counting in low power modes.

ISDE Property	Value	Description
NMI Callback	NULL	Callback. A user callback function can be registered in open. If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the IRQn triggers. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

スレッド監視フレームワークモジュールのクロック構成

ISDE を使用して、[Clocks] タブを使用して WDT クロックを構成します。

初期設定では、WDT は PCLKB 周波数に基づいて動作します。クロックコンフィギュレータを使用するか、実行時に CGC インタフェースを使用して、ISDE で PCLKB の周波数を設定できます。

PCLKB が 60MHz で動作している状態での WDT の最大タイムアウト期間は約 2.24 秒です。

IWDT クロックが 15kHz で動作している場合の最大タイムアウト期間は、35 秒をわずかに下回ります。

スレッド監視フレームワークモジュールのピン構成

スレッド監視フレームワークは、動作にあたってピンを必要としません。

スレッド監視フレームワークモジュールアプリケーションのスレッド

スレッド監視フレームワークにより監視されるスレッドが監視モジュールと連携するには、実装されている必要があります。監視するスレッドがスレッド間メッセージングに登録されていると、予測された最小および最大カウント値が、これらの値を含む sf_thread_monitor_counter_min_max_t 型の構造体へのポインタを通じて渡されます。

スレッドのカウンタと最小値および最大値は、g_sf_thread_monitor.p_api->threadRegister() を呼び出してスレッド監視フレームワークに登録する必要があります。

監視対象のスレッドのループが完了するたびに、g_sf_thread_monitor.p_api->countIncrement() を呼び出してカウンタの値を更新する必要があります。

その他のスレッド監視フレームワークモジュール設定

スレッド監視フレームワークは割り込みを使用しません。WDT または IWDT はリセットを生成するように構成する必要があります。

4.1.30.6 アプリケーションでのスレッド監視フレームワークモジュールの使用

一般的なアプリケーションで、sf_thread_monitor 上のスレッド監視フレームワークモジュールを使用する際の手順は次のとおりです。

1) sf_thread_monitor_api_t::open API を使用してスレッド監視を初期化します。
2) sf_thread_monitor_api_t::threadRegister API で監視する必要のあるスレッドを登録します。
3) sf_thread_monitor_api_t::countIncrement API を使用して、登録されたスレッドが実行されるたびにカウントをインクリメントします。
4) 登録されたスレッドを監視する必要がなくなったら、sf_thread_monitor_api_t::threadUnregister API を使用してスレッドを登録解除します。
5) sf_thread_monitor_api_t::close API でスレッド監視を閉じます（スレッド監視がアプリケーションで不要になった場合のみ）。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

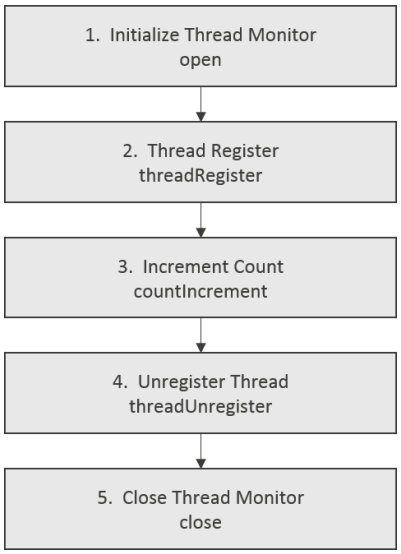


図 124:通常のスレッド監視フレームワークモジュールアプリケーションのフロー図

4.1.31 タッチ パネル フレームワーク

タッチパネルフレームワークモジュールは、タッチコントローラからメッセージを読み取るためのハイレベル API を提供し、I2C ポートを使用して実装されています。タッチパネルフレームワークモジュールは、Synergy MCU 上の IIC ペリフェラルまたは SCI ペリフェラルを使用します。

4.1.31.1 タッチパネルフレームワークモジュールの特長

- タッチコントローラからデータを読み取り、メッセージフレームワークのタッチイベントにサブスクライブされたキューに対してタッチメッセージをパブリッシュする

- 位置データ（X および Y 座標）を提供する
- タッチイベントタイプ（下、上、移動、保持、無効）を提供する
- 外部割り込み要求による I2C ベースのタッチパネルの実装をサポートする

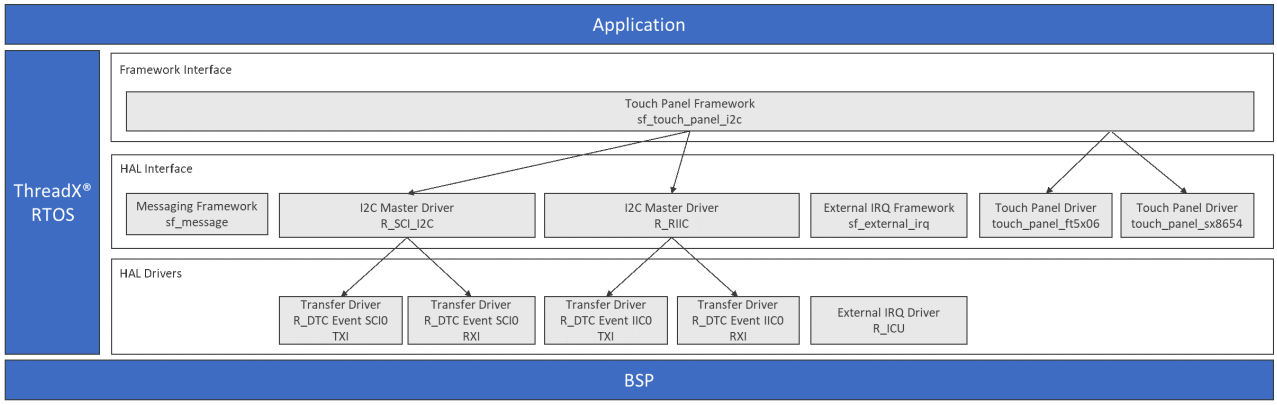


図 125:タッチパネルフレームワークモジュールのブロック図

4.1.31.2 タッチパネルフレームワークモジュール API の概要

タッチパネルフレームワークモジュールは、オープン、較正、開始、停止、クローズなどの API 機能を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

タッチパネルフレームワークモジュール API の要約

Function Name	Example API Call and Description
open	g_sf_touch_panel_i2c0.p_api->open(g_sf_touch_panel_i2c0.p_ctrl, g_sf_touch_panel_i2c0.p_cfg); Create required RTOS objects, call lower level module for hardware specific initialization, and create a thread to post touch data to a message queue.

Function Name	Example API Call and Description
calibrate	g_sf_touch_panel_i2c0.p_api->calibrate(g_sf_touch_panel_i2c0.p_ctrl, &expected, &actual, timeout); Begin calibration routine based on provided expected coordinates. Returns SSP_SUCCESS only if the tolerance is longer than the distance from the expected touch point to the actual touch point (using the formula below): p_calibrate->tolerance_pixels ^ 2 > (p_calibrate->x - x_measured) ^ 2 + (p_calibrate->y - y_measured) ^ 2
start	g_sf_touch_panel_i2c0.p_api->start(g_sf_touch_panel_i2c0.p_ctrl); Start scanning for touch events.
stop	g_sf_touch_panel_i2c0.p_api->stop (g_sf_touch_panel_i2c0.p_ctrl); Stop scanning for touch events.
reset	g_sf_touch_panel_i2c0.p_api->reset(g_sf_touch_panel_i2c0.p_ctrl); Reset touch controller if reset pin is provided, and resets the I2C bus.
close	g_sf_touch_panel_i2c0.p_api->close(g_sf_touch_panel_i2c0.p_ctrl); Terminate touch thread and close channel at HAL layer.
versionGet	g_sf_touch_panel_i2c0.p_api->versionGet(&version); Retrieves API version and stores it in the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API call successful.
SSP_ERR_ASSERTION	A pointer parameter was NULL, or a lower level driver reported this error.
SSP_ERR_INTERNAL	The touch panel thread or event flags could not be created, or a lower level driver reported this error.
SSP_ERR_CALIBRATE_FAILED	Actual touch value was not in expected range.
SSP_ERR_NOT_OPEN	Touch panel is not configured. Call SF_TOUCH_PANEL_I2C_Open.
SSP_ERR_IN_USE	Mutex was not available, or a lower level driver reported this error.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.31.3 タッチパネルフレームワークモジュールの動作の概要

タッチパネルフレームワークモジュールは、タッチコントローラからデータを読み取り、メッセージフレームワークのタッチイベントにサブスクライブされたキューに対してタッチメッセージをパブリッシュします。タッチ イベントには、位置データ (X および Y 座標) や、タッチ イベントの種類 (上、下、移動、保持、または無効) が含まれます。タッチ パネル フレームワークは、外部割り込みを使用する I2C ベースのタッチパネル実装をサポートしています。タッチ パネル フレームワークは、内部的にスレッドを作成し、タッチ コントローラーをリードします。

タッチフレームワークを使用する場合の通常のフローは次のとおりです。

1）タッチメッセージを保留にします。

2）タッチメッセージを解析します。

次の例は、アプリケーションで　g_sf_message　という名前のメッセージフレームワークインスタンスを使用し、g_sf_touch_panel_queue.という名前のキューにポストされたタッチメッセージを受信するための主な手順を示しています。

```
while(1)
{
/** Wait for an update message on the system queue. Process events as         fo
llows. */
    sf_message_header_t * p_message = NULL;

g_sf_message.p_api->pend(g_sf_message.p_ctrl, &g_sf_touch_panel_queue,      (sf_
message_header_t **) &p_message, TX_WAIT_FOREVER);
    switch (p_message->event_b.class)
```

```
    {
        case SF_MESSAGE_EVENT_CLASS_TOUCH:
        {
            /** Touch   event class events are handled as follows: */
            switch (p_message->event_b.code)
            {
                case SF_MESSAGE_EVENT_NEW_DATA:
                {
sf_touch_panel_payload_t * p_touch_payload =          (sf_touch_panel_payload_t
*) p_message;

                    switch (p_touch_payload->event_type)
                    {
                        case SF_TOUCH_PANEL_EVENT_DOWN:
                            /** Do something. */
                            break;

                        case SF_TOUCH_PANEL_EVENT_UP:
                            /** Do something. */
                            break;
                        case SF_TOUCH_PANEL_EVENT_HOLD:
                        case SF_TOUCH_PANEL_EVENT_MOVE:
                            /** Do something. */
                            break;
                        case SF_TOUCH_PANEL_EVENT_INVALID:
                            /** Do something. */
                            break;
                        default:
                            break;
                    }
                }
            }
                case SF_MESSAGE_EVENT_NEW_DATA:
                    break;
        }
    }
    /** After message is processed, release buffer. */
    err = g_sf_message.p_api->bufferRelease(g_sf_message.p_ctrl, (sf_message_hea
der_t *) p_message, SF_MESSAGE_RELEASE_OPTION_ACK);
}
```

タッチパネルフレームワークモジュールの動作に関する重要な注意事項と制限事項

初期構成

タッチパネルフレームワークをプロジェクトに追加すると、e^2 studio ISDE はタッチイベントクラスと新しいデータイベントをプロジェクトコンフィギュレータの [Messaging] タブに自動的に生成します。プロジェクトでタッチイベントメッセージ subscriber(s) を構成するには、次の手順に従います。

- e^2 studio Synergy コンフィギュレーターの [メッセージ] タブの [イベント クラス] ペインで、タッチ イベント クラスを選択します。
- [メッセージ] タブの [タッチ サブスクライバー] ペインでサブスクライバー スレッドのチェックボックスを選択します（スレッドがペインに表示されていない場合は、[Touch Subscriber] ペインの右上にある [New] アイコンをクリックしてスレッドを追加します）。

カスタムタッチパネルチップドライバーの作成

カスタムタッチチップドライバーを作成するには、synergy/ssp_supplemental/ touch_drivers にある SSP の既存のドライバーコードを参照し（このディレクトリは既存のタッチチップドライバーが Synergy コンフィギュレータで選択されている場合にのみ表示され、この操作を行うには Synergy コントローラに移動して [New] > [Framework] > [Input] > [Touch Panel Framework on sf_touch_panel_i2c] > [Add Touch Driver] を選択し、参照用の既存のタッチドライバーを選択してプロジェクトコンテンツを生成します）、次の手順と擬似コード例を使用してタッチパネルフレームワークに接続します。

- タッチ チップ ドライバー インスタンスを実装し、ドライバーをアタッチするタッチ パネル フレームワークを有効化します。
- チップ固有の設定が、使用するタッチスクリーンとアプリケーションのユースケースに基づいて構成されていることを確認します。たとえば、SX8654 ドライバーの RPNDT 設定（SX8654_REG_TOUCH1_RPDNT_RESISTOR_VALUE_CFG）は、使用するタッチスクリーンに基づいて設定されている必要があります。
- sf_touch_panel_i2c_chip_t::payloadGet 関数を実行し、タッチパネルコントローラデバイスから I2C インタフェースを介してタッチイベントとタッチ座標を取得します。
- sf_touch_panel_api_t::reset 関数を実行し、関連付けられた GPIO ピンと I2C インタフェースを使用してタッチパネルコントローラデバイスをリセットします。

タッチチップドライバーのテンプレート

注：以下の例では、xxxxx をカスタムタッチコントローラチップのパーツ番号に置換します。

```
/** Template for Touch Chip Driver Instance */
const sf_touch_panel_i2c_chip_t g_sf_touch_panel_i2c_chip_xxxxx =
{
    .payloadGet = xxxxx_payload_get,
    .reset      = xxxxx_reset
};

/** Pseudo code for payloadGet function */
ssp_err_t xxxxx_payload_get (
        sf_touch_panel_ctrl_t * const p_ctrl,
        sf_touch_panel_payload_t * const p_payload)
{
    /* The following is an overview of the I2C touch function.
     * Refer to the existing touch chip driver code for details.
     */

    /* Get I2C interface and the control block from p_ctrl.
     * p_i2c_api  = p_ctrl->p_lower_lvl_ctrl->p_lower_lvl_i2c->p_api;
     * p_i2c_ctrl = p_ctrl->p_lower_lvl_ctrl->p_lower_lvl_i2c->p_ctrl;
     */

    /* Get External IRQ interface and the control block from p_ctrl.
     * p_irq_api  = p_ctrl->p_lower_lvl_ctrl->p_lower_lvl_irq->p_api;
     * p_irq_ctrl = p_ctrl->p_lower_lvl_ctrl->p_lower_lvl_irq->p_ctrl;  */
```

```
    /* Call the wait API of the external IRQ interface to get interrupt
     * from touch chip.
     * p_irq_api->wait(pirq_ctrl, wait_option);
     */

    /* Call the read API of the I2C Interface to read touch event and
     * coordinate from touch chip.
     * p_i2c_api->read(pi2c_ctrl, …);
     */

    /* Set the obtained touch event and coordinate to p_payload.
     * p_payload->event_type = SF_TOUCH_PANEL_EVENT_XXXXX;
     * p_payload->x = <coordinate x>;
     * p_payload->y = <coordinate y>;
     */

    /* Store the touch event and coordinate in p_ctrl->last_payload,
     * which can be referred in the next touch event processing.
     * p_ctrl->last_payload.event_type = p_payload->event_type;
     * p_ctrl->last_payload.x = p_payload->x;
     * p_ctrl->last_payload.y = p_payload->y;
     */

    /* Return the error code. */
}

/** Pseudo code for reset function */
ssp_err_t xxxxx_chip_reset (sf_touch_panel_ctrl_t * const p_ctrl)
{
    /* The following is an overview of the I2C touch reset function.
     * Refer the existing touch chip driver code for details.
     */

    /* Get I2C interface and the control block from p_ctrl.
     * p_i2c_api  = p_ctrl->p_lower_lvl_ctrl->p_lower_lvl_i2c->p_api;
     * p_i2c_ctrl = p_ctrl->p_lower_lvl_ctrl->p_lower_lvl_i2c->p_ctrl;
     */

    /* Reset touch chip by setting GPIO reset pin low. */
    g_ioport_on_ioport.pinWrite(p_ctrl->p_lower_lvl_ctrl->pin,
                                            IOPORT_LEVEL_LOW);

    /* Delay a certain time. */

    /* Call reset API of I2C peripheral to issue reset to the touch chip.
     * p_i2c_api->reset(pi2c_ctrl);
     */

    /** Release touch chip from reset */
```

```
    g_ioport_on_ioport.pinWrite(p_ctrl->p_lower_lvl_ctrl->pin,
                                            IOPORT_LEVEL_HIGH);

    /* Return the error code */
}
```

カスタムタッチパネルチップドライバーの構成

プロジェクトに対してカスタムタッチチップドライバーを構成するには、以下の手順に従います。

手順 1. Synergy C プロジェクトを作成します。

手順 2. 以下の手順に従って、既存のタッチドライバー XML をカスタムタッチドライバー用に更新します。以下の説明に従って、既存のタッチ XML を変更します。

- プロジェクトルートフォルダの下にある .module_descriptions フォルダに移動します。
- タッチドライバー XML を編集します。
- Renesas##HAL Drivers##touch panel##touch_panel_i2c_ft5x06####< バージョン >OrRenesas##HAL Drivers##touch panel##touch_panel_i2c_sx8654####< バージョン >
- [value] フィールドで、インスタンス構造体の名前をカスタムドライバーインスタンス構造体の名前に変更します。たとえば、上のコード例の「"g_sf_touch_panel_i2c_chip_xxxxx"」です。
- 新しいインスタンス宣言を </property> タブの後に追加します。例は次のとおりです。

```
<declarations>
  extern const sf_touch_panel_i2c_chip_t g_sf_touch_panel_i2c_chip_xxxxx;
<declarations>
```

- XML 内のすべてのタッチチップ番号をカスタムタッチチップ番号に変更します。
- XML ファイルを保存して閉じます。変更した XML ファイルの内容の例を以下に示します。

```
<?xml version="1.0" ?>
<synergyModuleDescription>
    <module config="config.external.ex_touch_panel_i2c_xxxxxx" display="Driver|I
nput|Touch Panel Driver on touch_panel_xxxxxx" id="module.external.ex_touch_pane
l_i2c_xxxxxx_on_ex_touch_panel_i2c_xxxxxx" version="1">
                    <requires interface="_rtos" />
                    <provides interface="interface.external.ex_touch_panel" />
          <!-- Give name of the instance structure used in the driver in the va
lue field -->
                    <property id="module.external.ex_touch_panel.name" display=
"Name" default="module.external.ex_touch_panel.name.xxxxxx">
                                <option display="g_sf_touch_panel_i2c_chip_xxxxx
x" id="module.external.ex_touch_panel.name.xxxxxx" value="g_sf_touch_panel_i2c_c
hip_xxxxx " />
                    </property>

             <declarations>
             extern const sf_touch_panel_i2c_chip_t g_sf_touch_panel_i2c_chip_xx
xxx;
```

```
    </declarations>
      </module>
</synergyModuleDescription>
```

手順 3. プロジェクトコンフィギュレータを開き、[New] > [Framework] > [Input] > [Touch Panel Framework on sf_touch_panel_i2c] の順に選択してタッチパネルフレームワークを追加します（コンフィギュレータが既に開いていた場合は、いったん閉じてから再び開く必要があります）。

手順 4. すべてのコンポーネントを構成します。

手順 5. カスタムタッチパネルチップドライバーを [Add Touch Driver] ボックスに追加します。（上記の説明に従って XML を正しく変更した場合は、開いているカスタムタッチドライバーが表示されます）。

手順 6. プロジェクトコンテンツを生成し、新規ディレクトリ（たとえば、touch_panel_i2c_xxxxxx）構造体を synergy/ ssp_supplemental/ touch_drivers, の下に追加します。ディレクトリ構造体は次のようになります：synergy/ ssp_supplemental/ touch_drivers/ touch_panel_i2c_xxxxxx（xxxxxx はタッチコントローラのパーツ番号で置換します）。

手順 7. カスタムドライバーコードをコピーして ssp_supplemental/ touch_drivers/ touch_panel_i2c_xxxxxx). にペーストし、このディレクトリに追加します。

手順 8. ビルドに既存のドライバーが存在する場合は除外します（.c ファイルを右クリック > [Exclude from build…] > [Select all] > [OK]）。

手順 9. コードをビルドします。

- ユーザーコールバックは、I2C バス通信手順では使用できません。
- sf_touch_panel_api_t::reset API は、次の条件が満たされた場合にのみ使用できます：（1）sf_touch_panel_api_t::stop API がコールされている。（2）タッチイベントが発生し、タッチパネルフレームワークがタッチイベントメッセージをポストしている。
- タッチパネルドライバーのアドオンディレクトリは（SSPv1.2.0-b.1 の）renesas_sybd から ssp_supplemental（SSPv1.2.0）に変更されました。SSPv1.2.0-b.1 と SSPv1.2.0 の両方が開発環境にインストールされている場合は、両方のタッチパネルモジュールが [Component] タブで選択されます。プロジェクトで、ビルドから renesas_sybd フォルダを除外します。'renesas_sybd' フォルダを右クリック -> [Exclude from build] -> [Select all] -> [OK]。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.31.4 アプリケーションへのタッチパネルフレームワークワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してタッチパネルフレームワークモジュールをアプリケーションに組み込む方法について説明します。

注：このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

タッチパネルフレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。（タッチパネルフレームワークモジュールのデフォルト名は g_sf_touch_panel_i2c0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

タッチパネルフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_touch_panel_i2c0 Touch Panel Framework on sf_touch_panel_i2c	Threads	New Stack> Framework> Input> Touch Panel Framework on sf_touch_panel_i2c

次の図に示すように、sf_touch_panel_i2c のタッチパネルフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

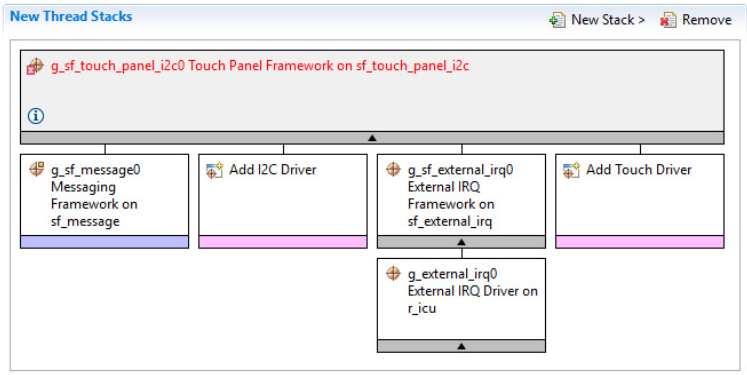


図 126:タッチパネルフレームワークモジュールのスタック

4.1.31.5 タッチパネルフレームワークモジュールの構成

ユーザーは必要な動作に合わせてタッチパネルフレームワークモジュールを構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます (ブロックが赤で強調表示される)。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成表の設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_touch_panel_i2c 上のタッチパネルフレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_touch_panel_i2c0	Module name.
Thread Priority	3	Thread priority selection.
Hsize Pixels	800	Hsize pixels selection.
Vsize Pixels	480	Vsize pixels selection.
Update Hz	10	Update hz selection.
Reset Pin	IOPORT_PORT_10_PIN_02	Reset pin selection.
Touch Event Class Instance Number	0	Touch event class instance number selection.
Touch Coordinate Rotation Angle (Clockwise)	0, 90 (select this if 'Screen Rotation Angle' in GUIX Port is '270'), 180, 270 (select this if 'Screen Rotation Angle' in GUIX Port is '90') Default: 0	Touch coordinate rotation angle selection.
Name of generated initialization function	sf_touch_panel_i2c_init0	Name of generated initialization function
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

デフォルト以外の設定が望ましい場合もあります。たとえば、異なる画面サイズの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

タッチパネルフレームワークのローレベルモジュールの構成

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、ローレベルモジュールの [properties] セクションのすべての設定を示します。

sf_message 上のメッセージフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Message Queue Depth (Total number of messages to be enqueued in a Message Queue)	16	Specify the size of Thread X Message Queue in bytes for Message Subscribers. This value is applied to all the Message Queues.
Name	g_sf_message0	The name of Messaging Framework module control block instance.
Work memory size in bytes	2048	Specify the work memory size in bytes. Choosing a small number results a small number of buffers which can be allocated at the same time (You can confirm the total buffer number on: sf_message_ctrl_t::number_of_buffers). If the value is smaller than the peak number of messages to be posted at the same time, the Framework occurs a buffer allocation failure affecting system performance.
Pointer to subscriber list array	p_subscriber_lists	Specify the name of pointer to the Subscriber List array.

ISDE Property	Value	Description
name of the block pool internally used in the messaging framework	sf_msg_blk_pool	The name of memory block memory the Framework creates in the control block. This parameter might be useful for debugging purpose but NULL can be specified for saving memory.
Name of generated initialization function	sf_message_init0	Name of generated initialization function
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_riic 上の I2C マスタ ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.
Name	g_i2c0	Module name.
Channel	0	Specify the IIC channel to be used with this configuration.
Rate	Standard, Fast-mode, Fast-mode Plus Default: Standard	Standard, Fast, and Fast-plus. (See IIC Rate Calculation.)
Slave Address	0x00	Set the address of the slave device the I2C master will be communicating with.

ISDE Property	Value	Description
Address Mode	7-Bit, 10-Bit Default: 7-Bit	Only 7-bit addresses are currently supported.
Callback	NULL	A user callback function can be registered in i2c_api_master_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) for each of the conditions defined in i2c_event_t. Warning: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Receive interrupt priority selection.
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Transmit interrupt priority selection.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Transmit end interrupt priority selection.

ISDE Property	Value	Description
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Error interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event IIC0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Enabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection

ISDE Property	Value	Description
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event IIC0 TXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event IIC0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer1	Module name
Mode	Normal	Mode selection

ISDE Property	Value	Description
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event IIC0 RXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sci_i2c 上の I2C マスタ ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.
Name	g_i2c0	Module name.
Channel	0	Specify the SCI channel to be used with this configuration. SCI has channels as follows: Series S7 : 0 1 2 3 4 5 6 7 8 9; Series S3 : 0 1 2 3 4 - - - - 9; Series S1 : 0 1 - - - - - - - 9 .
Rate	Standard, Fast-mode Default: Standard	Standard and Fast.
Slave Address	0x00	Address of the slave device.
Address Mode	7-Bit, 10-Bit Default: 7-Bit	Only 7-bit addresses are currently supported.
SDA Output Delay (nano seconds)	300	SDA output delay in nanoseconds.
Bit Rate Modulation Enable	Enable, Disable Default: Enable	Enables bitrate modulation function.

ISDE Property	Value	Description
Callback	NULL	A user callback function can be registered in i2c_api_master_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) for each of the conditions defined in i2c_event_t. Warning: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Receive interrupt priority selection.
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Transmit interrupt priority selection.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Transmit end interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Enabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SCI0 TXI	Activation source selection
Auto Enable	FALSE	Auto enable selection

ISDE Property	Value	Description
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection

ISDE Property	Value	Description
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SCI0 RXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_external_irq 上の外部 IRQ フレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Controls whether to include code for API parameter checking.
Name	g_sf_external_irq0	Framework name.

ISDE Property	Value	Description
Event	None, Semaphore Put Default: Semaphore Put	Event selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_icu 上の外部 IRQ ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter checking setting enables or disables the addition of parameter checking code.
Name	g_external_irq0	Module name.
Channel	0	Specifies the hardware IRQ channel used.
Trigger	Falling, Rising, Both Edges, Low Level Default: Rising	Configures edge or level triggering.
Digital Filtering	Enabled, Disabled Default: Disabled	Digital filter enable/disable.
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCLK/1, PLCK/8, PLCK/32, PCLK/64 Default: PCKL/64	Sets noise filter sampling period.
Interrupt enabled after initialization	True, False Default: True	Determines if the interrupt is enabled immediately after initialization.

ISDE Property	Value	Description
Callback	NULL	A user callback function can be registered in external_irq_api_t::open. If this callback function is provided, it is called from the interrupt service routine (ISR) each time the IRQn triggers. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Pin Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	An Interrupt priority can be registered in external_irq_cfg_t::irq_ipl.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

touch_panel_ft5x06 上のタッチパネルドライバーの構成設定

ISDE Property	Value	Description
Name	g_sf_touch_panel_i2c_chip_ft5x06	Module name.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

touch_panel_sx8654 上のタッチパネルドライバーの構成設定

ISDE Property	Value	Description
Name	g_sf_touch_panel_i2c_chip_sx8654	Module name.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。注 : モジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

タッチパネルフレームワークモジュールのクロック構成

ここで説明する I2C インタフェースの実装例では、SCI 周辺機器を使用しています。その他の実装にはさまざまな選択肢があり、次の例から推測できます。SCI ペリフェラルモジュールは PCLKB をそのクロックソースとして使用します。PCLKB 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。構成時に、I2C 転送レートは、ユーザー選択 PCLB レートとユーザー選択転送レートに基づいて、ドライバーにより内部で計算および設定されます。ユーザー選択レートを実現できないような構成が PCLKB で行われた場合は、ドライバーを初期化するときにエラーが返されます。

タッチパネルフレームワークモジュールのピン構成

ここで説明する I2C インタフェースの実装例では、SCI 周辺機器を使用しています。その他の実装にはさまざまな選択肢がありますが、以下の例から推測できます。Synergy キット固有のピン設定を以下のセクションに示します。SCI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

タッチパネルフレームワークモジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SCI0	Pins	Select Peripherals > Connectivity:SCI > SCI0

注 : 選択シーケンスでは、SCI0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_dac 上の DAC ドライバーのピン構成設定

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Asynchronous UART, Synchronous UART, Simple I2C, Simple SPI, SmartCard Default: Disabled **Simple I2C**	Select Simple I2C as the Operation Mode for SPI on SCI
RXD1_SCL1_MISO1	None, P212, P708 Default: None	SCL Pin
TXD1_SDA1_MOSI1	None, P213, P709 Default: None	SDA Pin

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

タッチパネルフレームワークモジュールの Synergy キット固有の設定

前のセクションに示した選択項目に加えて、使用されるタッチチップに基づいていくつかのキット固有の設定が必要な場合があります。これらの設定を以下に例としてまとめます。この例では、独自のカスタムタッチコントローラデバイスに対してこれらを構成する方法が示されます。

外部 IRQ 設定の構成 :

DK-S7G2（タッチチップ SX8654）では、以下を選択します。

- チャネル : 7
- トリガ : Falling

SK-S7G2 (タッチ チップ SX8654) では、以下を選択します。

- チャネル : 9
- トリガ : Falling

PE-HMI1-S7G2 (タッチ チップ FT5x06) では、以下を選択します。

- チャネル : 12
- トリガ : Falling

すべてのボードの共通設定 :

- デジタル フィルター設定 : Any

- 初期化後に割り込みを有効にする : True
- コールバック : NULL

I2C ドライバー設定の構成：

DK-S7G2 の場合は、r_sci_i2c: を選択します。

- チャネル : 7
- 速度 : Standard
- スレーブ アドレス : 0x48
- アドレス モード : 7-bit

SK-S7G2 の場合は、r_riic: を選択します。

- チャネル : 2
- 速度 : Standard
- スレーブ アドレス : 0x48
- アドレス モード : 7-bit

PE-HMI1-S7G2 の場合は、r_riic: を選択します。

- チャネル : 1
- 速度 : 高速モード
- スレーブ アドレス : 0x38
- アドレス モード : 7-bit

すべてのボードの共通設定 :

- コールバック : NULL

タッチパネルフレームワーク設定の構成：

DK-S7G2 の場合 :

- タッチチップ：g_sf_touch_panel_i2c_chip_sx8654
- 幅のサイズ ピクセル : 480
- 高さのサイズ ピクセル : 272
- リセットピン：IOPORT_PORT_07_PIN_11

SK-S7G2 の場合 :

- タッチチップ：g_sf_touch_panel_i2c_chip_sx8654
- 幅のサイズ ピクセル : 240
- 高さのサイズ ピクセル : 320
- リセットピン：IOPORT_PORT_06_PIN_09

PE-HMI1-S7G2 の場合 :

- タッチチップ：g_sf_touch_panel_i2c_chip_ft5x06
- 幅のサイズ ピクセル : 800
- 高さのサイズ ピクセル : 480

- リセットピン：IOPORT_PORT_10_PIN_02

ボードの共通設定 :

- スレッドのプライオリティ : Any

SX8654 および FT5x06 のタッチチップドライバーは SSP のビルトインドライバーで、モジュールを構成するときに選択できます。

4.1.31.6 アプリケーションでのタッチパネルフレームワークモジュールの使用

アプリケーションでタッチパネルフレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) sf_touch_panel_api_t::start API を使用してタッチパネルフレームワークモジュールを初期化します。（SSP で自動的に実行）
2) sf_message_api_t::pend API を使用してタッチメッセージを待機します。
3) アプリケーションコードを使用してタッチメッセージを解析します。
4) アプリケーションコードを使用して、必要に応じて受信したデータを操作します。さらに必要なデータがある場合は、手順 2 に戻ります。
5) sf_touch_panel_api_t::close API を使用して、モジュールを閉じます（必要な場合）。

これらの一般的な手順を、次の図の通常の動作フローに示します。

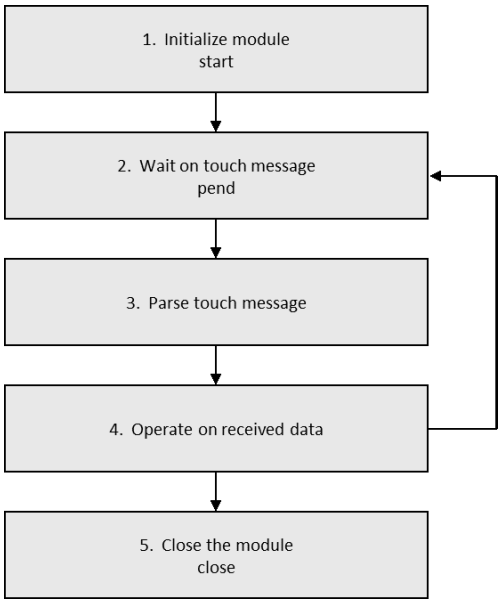


図 127:通常のタッチパネルフレームワークモジュールアプリケーションのフロー図

4.1.32 タッチパネル V2 フレームワーク

タッチパネル V2 フレームワークモジュールは、タッチコントローラから座標上のタッチデータとイベントを読み取るためのハイレベル API を提供します。タッチパネル V2 フレームワークモジュールは、タッチパネルチップドライバー SSP 補足モジュールを使用して、タッチパネルと通信します。

4.1.32.1 タッチパネル V2 フレームワークモジュールの特長

- 位置データ（X および Y 座標）を提供します。
- タッチ座標のローテーションを提供します。
- タッチイベントタイプ（下、上、移動、保持、無効）を提供します。
- コールバックを登録するか、API 関数を使用してタッチデータを取得できます。
- スカラ値、ローテーション、機械的な移動といった重要な要素に対するタッチパネルの較正をサポートします。
- タッチパネルチップドライバーへの共通の API インタフェースを提供します。
- 調整可能な更新周波数をサポートします。

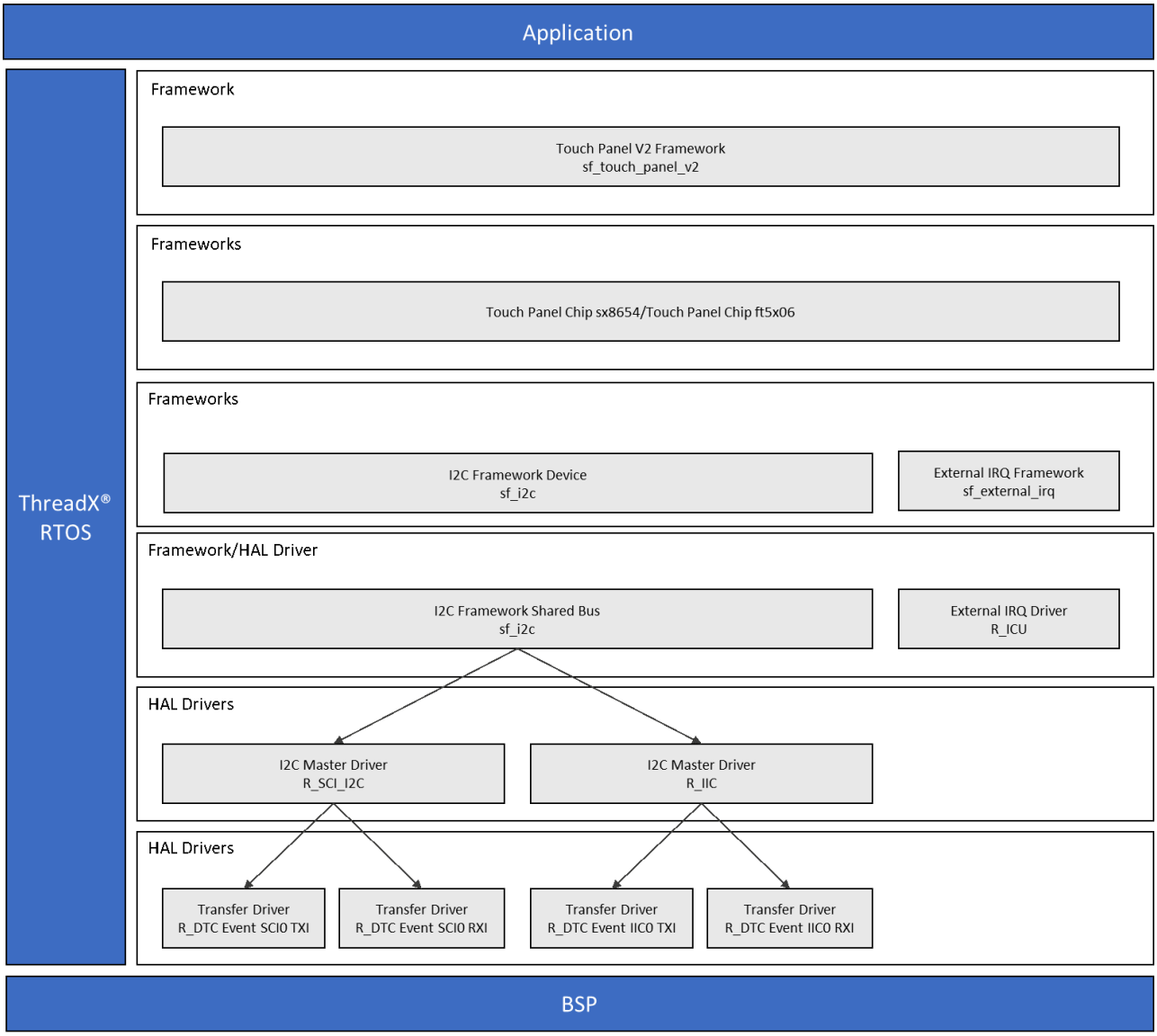


図 128:タッチパネル V2 フレームワークモジュールのブロック図

4.1.32.2 タッチパネル V2 フレームワークモジュールの API の概要

タッチパネル V2 フレームワークモジュールは、オープン、較正、開始、停止、クローズなどの API 関数を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

タッチパネル V2 フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_touch_panel_v2_0.p_api->open(g_sf_touch_panel_v2_0.p_ctrl, g_sf_touch_panel_v2_0.p_cfg); Create required RTOS objects, call a lower level module for hardware specific initialization and create a thread to post touch data to a user application.
calibrate	g_sf_touch_panel_v2_0.p_api->calibrate(g_sf_touch_panel_v2_0.p_ctrl, p_display, p_touchscreen, timeout); Begin calibration routine based on provided expected and actual coordinates.
start	g_sf_touch_panel_v2_0.p_api->start(g_sf_touch_panel_v2_0.p_ctrl); Start scanning for touch events.
stop	g_sf_touch_panel_v2_0.p_api->stop (g_sf_touch_panel_v2_0.p_ctrl); Stop scanning for touch events.
touchDataGet	g_sf_touch_panel_v2_0.p_api->touchDataGet (g_sf_touch_panel_v2_0.p_ctrl, p_payload, timeout); Reads the touch data and fills in the touch payload data.
reset	g_sf_touch_panel_v2_0.p_api->reset(g_sf_touch_panel_v2_0.p_ctrl); Resets touch controller if reset pin is provided.

Function Name	Example API Call and Description
close	g_sf_touch_panel_v2_0.p_api->close(g_sf_touch_panel_v2_0.p_ctrl); Terminates touch thread and closes channel at HAL layer.
versionGet	g_sf_touch_panel_v2_0.p_api->versionGet(g_sf_touch_panel_v2_0.p_version); Retrieves API version and stores it in the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API call successful.
SSP_ERR_ASSERTION	A pointer parameter was NULL, or a lower level driver reported this error.
SSP_ERR_INTERNAL	The touch panel thread or event flags could not be created, or a lower level driver reported this error.
SSP_ERR_CALIBRATE_FAILED	Actual touch value was not in expected range.
SSP_ERR_NOT_OPEN	Touch panel is not configured. Call SF_TOUCH_PANEL_I2C_Open.
SSP_ERR_IN_USE	Mutex was not available, or a lower level driver reported this error.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.32.3 タッチパネル V2 フレームワークモジュールの動作の概要

タッチパネル V2 フレームワークモジュールは、タッチコントローラからデータをスキャンし、ユーザーが登録したコールバックを呼び出します。ユーザーコールバックが登録されていない場合は、

sf_touch_panel_v2_api_t::touchDataGet API 関数を使用してデータを取得できます。タッチデータには、タッチ座標（X および Y 座標）や、タッチイベントタイプ（上、下、移動、保持、または無効）が含まれます。タッチパネル V2 フレームワークは、内部スレッドを使用し、タッチコントローラを読み取ります。

タッチパネル V2 フレームワークモジュールの動作に関する重要な注意事項と制限事項

タッチパネル V2 フレームワークの動作に関する重要な要素を以下で説明します。

自動初期化とオートスタート

フレームワークを自動的に初期化するには、[Synergy configurator] ウィンドウで [Auto Initialization] プロパティを有効にする必要があります。タッチパネルチップから自動的にタッチデータを受信するには、コンフィギュレータのウィンドウで [Auto start] プロパティを有効にする必要があります。

ユーザーコールバック

タッチデータを取得するには、ユーザーコールバックを登録する必要があります。ユーザーコールバックが登録されている場合は、タッチイベントの発生時にフレームワークによってコールバックが呼び出されます。

タッチデータを取得するための API

sf_touch_panel_v2_api_t::touchDataGet API 関数はタッチデータに使用できます。この関数は（API に渡されたタイムアウト引数に基づいて）新しいタッチイベントデータが使用可能になるかタイムアウトするまで待機します。

更新周波数

アプリケーションは、Synergy コンフィギュレータの [property] ウィンドウで構成された指定の更新周波数（[Update Hz]）に従い、反復的なタッチイベント（タッチイベント下、タッチイベント保持、タッチイベント無効）によって通知を受け取ります。

注 : タッチイベント上およびタッチイベント下は、更新周波数に関係なく通知されます。

以下に例を示します。

[Update Hz] プロパティが 10Hz に設定されている場合は、アプリケーションに通知される反復的なタッチイベントは 1 秒間に 10 個のみです。

較正

タッチパネル V2 フレームワークは、スカラ値、ローテーション、機械的な移動に関するタッチパネルでの課題に対処するためにタッチデータの較正をサポートします。

較正されたデータを取得するには、ユーザーは sf_touch_panel_v2_api_t::calibrate API 関数をコールし、3 つの期待座標と取得座標のセットを渡す必要があります。

*注 :SF_TOUCH_PANEL_V2_API_T::CALIBRATE API をコールする前に、フレームワークを初期化する必要があります。*以下に例を示します。

分解能 480 x 272 のタッチパネルを使用している場合を例に説明します。タッチパネルの左上、右上、右下をタッチしたときのタッチパネル位置の取得座標（x, y）とそれらの期待座標は以下のとおりです。

タッチパネル座標

Location on Touch Panel	Expected Coordinates	Obtained Coordinates
Upper left corner	(0,0)	(17, 20)
Upper right corner	(480, 0)	(464, 17)
Lower right corner	(480, 272)	(463, 258)

上の 3 セットの値（期待座標と取得座標）が、期待されるフォーマットで calibrate 関数に渡されます。sf_touch_panel_v2_api_t::calibrate API 関数がコールされると、必要な較正係数が計算され、制御構造体に格納されます。これで、取得されたタッチデータが較正されます。

注 :SF_TOUCH_PANEL_V2_API_T::CLOSE API 関数がコールされると、格納された較正係数は消去され、再度 calibrate API をコールして較正されたタッチデータを取得する必要があります。

カスタムタッチパネルチップドライバーの作成

カスタムタッチパネルチップドライバーを作成するには、synergy/ssp_supplemental/touch_drivers. にある、SSP の既存のタッチチップドライバーコードを参照します。

注 : このディレクトリは、既存のタッチパネルチップドライバー touch_panel_chip_ft5x06 または touch_panel_chip_sx8654 が Synergy コンフィギュレータで選択されている場合にのみ表示されます。そのためには、Synergy コンフィギュレータに移動し、[New] > [Framework] > [Input] > [Touch Panel V2 Framework on sf_touch_panel_v2] > [Add Touch Driver] の順に選択します。参照する既存のタッチドライバーを選択し、プロジェクトコンテンツを生成します。

注 : カスタムドライバーの作成の詳細については、『r11an0132eu0102 SSP Module Development Guide』を参照してください。このドキュメントは、次のリンクを使用して参照できます。https://www.renesas.com/en-us/doc/products/renesas-synergy/apn/r11an0132eu0102-synergy-ssp-module-development-guide.pdf

以下の手順を使用して、カスタムタッチパネルチップドライバーをタッチパネル V2 フレームワークに接続します。

- タッチパネルチップドライバーインスタンス (synergy/ssp_supplemental/inc/framework/instances) とソースファイル (synergy/ssp_supplemental/touch_drivers). を実装します。
- 使用されるタッチスクリーンとアプリケーションのユースケースに基づいてチップ固有の設定が構成されていることを確認します。
- タッチパネルコントローラからタッチイベントとタッチ座標を取得する payloadGet 関数を実装します。
- タッチパネルチップをリセットするリセット関数を実装します。
- タッチパネルチップが I2C 以外の通信プロトコルを使用している場合、または外部 IRQ をサポートしていない場合は、open、payloadGet、reset および close 関数を変更します。また、タッチパネルチップドライバー構成 XML（".module_descriptions" フォルダにあります）を変更する必要もあります。

カスタムタッチパネルチップドライバーの構成

プロジェクトに対してカスタムタッチパネルチップドライバーを構成するには、以下の手順に従います。

1) Synergy プロジェクトを作成します。
2) 以下の手順に従って、既存のタッチドライバー XML をカスタムタッチドライバー用に更新します。以下の説明に従って、既存のタッチ XML を変更します。
   - プロジェクトルートフォルダの下にある .module_descriptions フォルダに移動します。
   - タッチパネルチップドライバー XML Renesas##HAL Drivers##touch panel##touch_panel_chip_ft5x06####< バージョン >OrRenesas##HAL Drivers##touch panel##touch_panel_chip_sx8654####< バージョン >

   を編集します。
   - インスタンス構造体の名前をカスタムドライバーインスタンス構造体の名前に変更します。たとえば、"g_touch_panel_chip_xxxxx" に変更します。

- 新しいインスタンス宣言と API 宣言を </header> タブに追加します。たとえば、<header>extern const sf_touch_panel_chip_instance_t**g_touch_panel_chip_xxxxx;extern const** sf_touch_panel_chip_api_tg_sf_touch_panel_chip_xxxxx;</header>

のようにします。

- XML 内のすべてのタッチパネルチップ番号をカスタムタッチチップ番号に変更します。
- I2C 以外の通信プロトコルが使用されている場合は、<requires> タグの interface と id を変更します。たとえば、<requires>id="module.external.ex_touch_panel_chip_ft5x06.requires.spi" interface="interface.framework. sf_spi_v2_on_sf_spi " display="Add framework"</requires >

のようにします。

- XML ファイルを保存して閉じます。

3) プロジェクトコンフィギュレータを開き、[New] > [Framework] > [Input] > [Touch Panel V2 Framework on sf_touch_panel_v2] の順に選択してタッチパネル V2 フレームワークを追加します（コンフィギュレータが既に開いていた場合は、いったん閉じてから再び開く必要があります）。
4) すべてのコンポーネントを構成します。
5) カスタムタッチパネルチップドライバーを [Add Touch Driver] ボックスに追加します。（XML を正しく変更した場合は、開いているカスタムタッチドライバーが表示されます）。
6) プロジェクトコンテンツを生成し、新規ディレクトリ (たとえば、touch_panel_chip_xxxxxx) 構造体を synergy/ ssp_supplemental/ touch_drivers, の下に追加します。ディレクトリ構造体は次のようになります : synergy/ ssp_supplemental/ touch_drivers/ touch_panel_chip_xxxxxx（xxxxxx はタッチコントローラのパーツ番号で置換します）。
7) カスタムドライバーコードをこのディレクトリ（ssp_supplemental/ touch_drivers/ touch_panel_chip_xxxxxx）. に追加します。
8) ビルドに既存のドライバーが存在する場合は除外します（.c ファイルを右クリック > [Exclude from build…] > [Select all] > [OK]）。
9) コードをビルドします。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.32.4 アプリケーションへのタッチパネル V2 フレームワークワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してタッチパネル V2 フレームワークモジュールをアプリケーションに組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

タッチパネル V2 フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。(タッチパネル V2 フレームワークモジュールのデフォルト名は g_sf_touch_panel_v2_0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

タッチパネル V2 フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_touch_panel_v2_0 Touch Panel V2 Framework on sf_touch_panel_v2	Threads	New Stack> Framework> Input> Touch Panel V2 Framework on sf_touch_panel_v2

次の図に示すように、sf_touch_panel_v2 のタッチパネル V2 フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

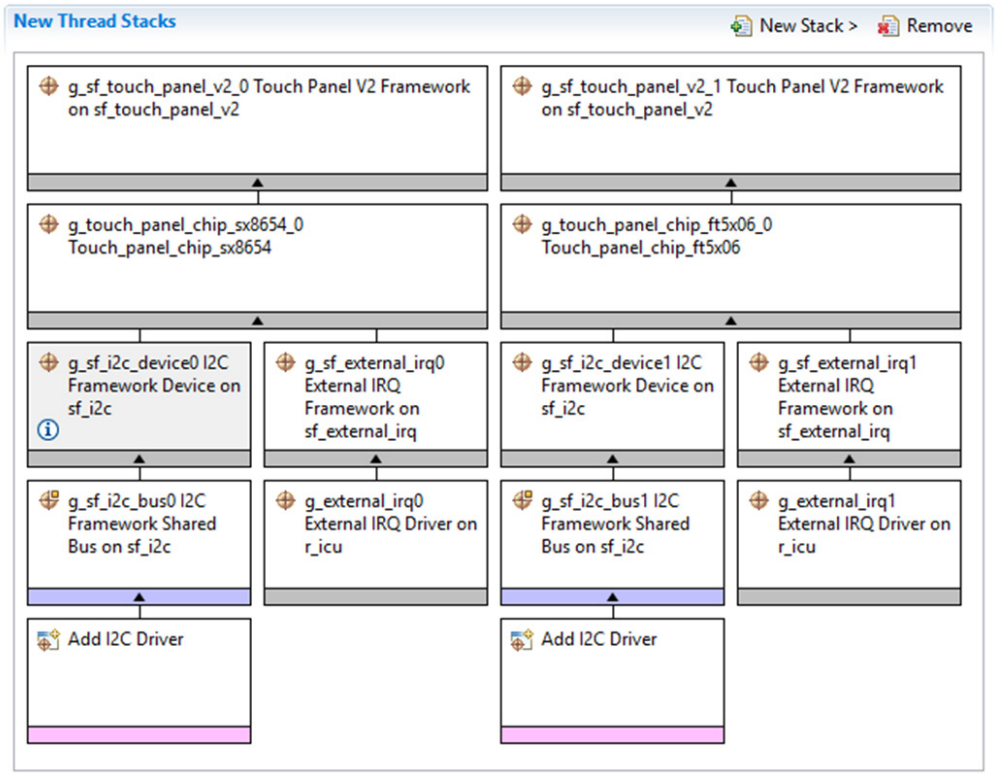


図 129:タッチパネル V2 フレームワークモジュールのスタック

4.1.32.5 タッチパネル V2 フレームワークモジュールの構成

ユーザーは必要な動作に合わせてタッチパネル V2 フレームワークモジュールを構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成テーブルの設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_touch_panel_v2 のタッチパネル V2 フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Thread Stack Size	512	Specify the touch panel thread stack size.
Name	g_sf_touch_panel_v2_0	Module name.
Thread Priority	3	Specify the thread priority.
Update Hz	10	Specify the update rate in Hertz.
Touch Coordinate Rotation Angle (Clockwise)	0, 90 (Select this if "Screen Rotation Angle in GUIX Port is 270), 180, 270 (Select this if "Screen Rotation Angle in GUIX Port is 90) Default: 0	Select the touch coordinate rotation angle.
Name of generated initialization function	sf_touch_panel_v2_init0	Specify the name of the generated initialization function.

ISDE Property	Value	Description
Auto Initialization	Enable, Disable Default: Enable	Select if sf_touch_panel_v2 will be initialized during startup.
Auto Start	Enable, Disable Default: Enable	Enabling this will start to get the touch data.
Name of touch panel callback function to be defined by user	NULL	Touch panel callback function name.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

デフォルト以外の設定が望ましい場合もあります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

タッチパネル V2 フレームワークのローレベルモジュールの構成

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、ローレベルモジュールの [properties] セクションのすべての設定を示します。

タッチパネルチップ sx8654 の構成設定

ISDE Property	Value	Description
Name	g_touch_panel_chip_sx8654_0	Module name.
Horizontal pixel count	480	Specify the number of horizontal pixels.
Vertical pixel count	272	Specify the number of vertical pixels.
Reset Port	00:11 Default: 07	Select the chip reset port.

ISDE Property	Value	Description
Reset Pin	00:15 Default: 11	Select the chip reset pin.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

タッチパネルチップ ft5x06 の構成設定

ISDE Property	Value	Description
Name	g_touch_panel_chip_ft5x06	Module name.
Horizontal pixel count	800	Specify the number of horizontal pixels.
Vertical pixel count	480	Specify the number of vertical pixels.
Reset Port	00:11 Default: 10	Select the chip reset port.
Reset Pin	00:15 Default: 02	Select the chip reset pin.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_i2c 上の I2C フレームワークデバイスの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: Enabled	Selects if code for parameter checking is to be included in the build.

ISDE Property	Value	Description
Name	g_sf_i2c_device0	Give a name to identify the I2C Framework device. API, Config and Control instances will be created based on this name.
Slave Address	0x00	Specify the address of the I2C slave device.
Address Mode	7-Bit, 10-Bit Default: 7-Bit	Select the I2C address mode.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_i2c 上の I2C 共有バスの構成設定

ISDE Property	Value	Description
Name	g_sf_i2c_bus0	Module name

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_riic 上の I2C マスタ ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Name	g_i2c0	Module name.
Channel	0	Specify the IIC channel to be used with this configuration.

ISDE Property	Value	Description
Rate	Standard, Fast-mode, Fast-mode Plus Default: Standard	Standard, Fast, and Fast-plus. (See IIC Rate Calculation.)
Slave Address	0x00	Set the address of the slave device the I2C master will be communicating with.
Address Mode	7-Bit, 10-Bit Default: 7-Bit	Only 7-bit addresses are currently supported.
Timeout Mode	Short Mode, Long Mode Default: Short Mode	Select the timeout mode.
Callback	NULL	A user callback function can be registered in i2c_api_master_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) for each of the conditions defined in i2c_event_t. Warning: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Default: Priority 12	Select the receive interrupt priority.

ISDE Property	Value	Description
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Default: Priority 12	Select the transmit interrupt priority.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Default: Priority 12	Select the transmit end interrupt priority.
Error Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Default: Priority 12	Select the error interrupt priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event IIC0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Software Start	Enabled, Disabled Default: Disabled	Include code for software start in the build.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Section to place the DTC vector table.
Name	g_transfer0	Module name.
Mode	Normal	Specify the hardware channel.

ISDE Property	Value	Description
Transfer Size	1 Byte	Select the transfer mode.
Destination Address Mode	Fixed	Select the transfer size.
Source Address Mode	Incremented	Select the address mode for the destination.
Repeat Area (Unused in Normal Mode)	Source	Select the address mode for the source.
Interrupt Frequency	After all transfers have completed	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.
Number of Transfers	0	Specify the number of transfers.
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source (Must enable IRQ)	Event IIC0 TXI	Select the DTC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback (Only valid with Software start)	NULL	A user callback that is called at the end of the transfer.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event IIC0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Software Start	Enabled, Disabled Default: Disabled	Include code for software start in the build.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Section to place the DTC vector table.
Name	g_transfer1	Module name.
Mode	Normal	Specify the hardware channel.
Transfer Size	1 Byte	Select the transfer mode.
Destination Address Mode	Incremented	Select the transfer size.
Source Address Mode	Fixed	Select the address mode for the destination.
Repeat Area (Unused in Normal Mode	Destination	Select the address mode for the source.
Interrupt Frequency	After all transfers have completed	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.

ISDE Property	Value	Description
Number of Transfers	0	Specify the number of transfers.
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source (Must enable IRQ)	Event IIC0 RXI	Select the DTC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback (Only valid with Software start)	NULL	A user callback that is called at the end of the transfer.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sci_i2c 上の I2C マスタドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Name	g_i2c0	Module name.
Channel	0 to 9	Specify the SCI channel to be used with this configuration. SCI has channels as follows: Series S7: 0 1 2 3 4 5 6 7 8 9; Series S3 : 0 1 2 3 4 - - - - 9; Series S1 : 0 1 - - - - - - - 9 .

ISDE Property	Value	Description
Rate	Standard, Fast-mode, Fast-mode plus Default: Standard	Select the I2C data rate.
Slave Address	0x00	Specify the slave address.
Address Mode	7-Bit, 10-Bit Default: 7-Bit	Only 7-bit addresses are currently supported.
SDA Output Delay (nano seconds)	300	SDA output delay in nanoseconds.
Bit Rate Modulation Enable	Enable, Disable Default: Enable	Enables bitrate modulation function.
Callback	NULL	A user callback function can be registered in i2c_api_master_t::open. If this callback function is provided, it will be called from the interrupt service routine (ISR) for each of the conditions defined in i2c_event_t. Warning: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Select the receive interrupt priority.

ISDE Property	Value	Description
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Select the transmit interrupt priority.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Select the transmit end interrupt priority.
Error Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Select the error interrupt priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Software Start	Enabled, Disabled Default: Disabled	Include code for software start in the build.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Section to place the DTC vector table.
Name	g_transfer0	Module name.
Mode	Block	Specify the hardware channel.

ISDE Property	Value	Description
Transfer Size	1 Byte	Select the transfer mode.
Destination Address Mode	Fixed	Select the transfer size.
Source Address Mode	Incremented	Select the address mode for the destination.
Repeat Area (Unused in Normal Mode	Source	Select the address mode for the source.
Interrupt Frequency	After all transfers have completed	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.
Number of Transfers	0	Specify the number of transfers.
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source (Must enable IRQ)	Event SCI0 TXI	Select the DTC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback (Only valid with Software start)	NULL	A user callback that is called at the end of the transfer.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Software Start	Enabled, Disabled Default: Disabled	Include code for software start in the build.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Section to place the DTC vector table.
Name	g_transfer1	Module name.
Mode	Normal	Specify the hardware channel.
Transfer Size	1 Byte	Select the transfer mode.
Destination Address Mode	Incremented	Select the transfer size.
Source Address Mode	Fixed	Select the address mode for the destination.
Repeat Area (Unused in Normal Mode	Destination	Select the address mode for the source.
Interrupt Frequency	After all transfers have completed	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.

ISDE Property	Value	Description
Number of Transfers	0	Specify the number of transfers.
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source (Must enable IRQ)	Event SCI0 RXI	Select the DTC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback (Only valid with Software start)	NULL	A user callback that is called at the end of the transfer.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_external_irq 上の外部 IRQ フレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Controls whether to include code for API parameter checking.
Name	g_sf_external_irq0	Framework name.
Event	None, Semaphore Put Default: Semaphore Put	Event selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_icu 上の外部 IRQ ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter checking setting enables or disables the addition of parameter checking code.
Name	g_external_irq0	Module name.
Channel	0	Specifies the hardware IRQ channel used.
Trigger	Falling, Rising, Both Edges, Low Level Default: Rising	Configures edge or level triggering.
Digital Filtering	Enabled, Disabled Default: Disabled	Digital filter enable/disable.
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCLK/1, PLCK/8, PLCK/32, PCLK/64 Default: PCKL/64	Sets noise filter sampling period.
Interrupt enabled after initialization	True, False Default: True	Determines if the interrupt is enabled immediately after initialization.

ISDE Property	Value	Description
Callback	NULL	A user callback function can be registered in external_irq_api_t::open. If this callback function is provided, it is called from the interrupt service routine (ISR) each time the IRQn triggers. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Pin Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	An Interrupt priority can be registered in external_irq_cfg_t::irq_ipl.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

タッチパネル V2 フレームワークモジュールのクロック構成

ここで説明する I2C インタフェースの実装例では、SCI 周辺機器を使用しています。その他の実装にはさまざまな選択肢があり、次の例から推測できます。SCI ペリフェラルモジュールは PCLKB をそのクロックソースとして使用します。PCLKB 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。構成時に、I2C 転送レートは、ユーザー選択 PCLB レートとユーザー選択転送レートに基づいて、ドライバーにより内部で計算および設定されます。ユーザー選択レートを実現できないような構成が PCLKB で行われた場合は、ドライバーを初期化するときにエラーが返されます。

タッチパネル V2 フレームワークモジュールのピン構成

ここで説明する I2C インタフェースの実装例では、SCI 周辺機器を使用しています。その他の実装にはさまざまな選択肢がありますが、以下の例から推測できます。Synergy キット固有のピン設定を以下のセクションに示します。SCI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

タッチパネル V2 フレームワークモジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SCI0	Pins	Select Peripherals > Connectivity:SCI > SCI0

*注 : この選択シーケンスでは SCI0 がドライバーに必要なハードウェアターゲットであることを想定しています。*シリアル接続のピン構成は以下のとおりです。

r_dac での DAC ドライバーのピン構成設定

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Asynchronous UART, Synchronous UART, Simple I2C, Simple SPI, SmartCard Default: Disabled Simple I2C	Select Simple I2C as the Operation Mode for SPI on SCI
RXD1_SCL1_MISO1	None, P212, P708 Default: None	SCL Pin
TXD1_SDA1_MOSI1	None, P213, P709 Default: None	SDA Pin

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.1.32.6 アプリケーションでのタッチパネル V2 フレームワークモジュールの使用

アプリケーションでタッチパネル V2 フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) タッチデータを受信するために、アプリケーションコードでユーザーコールバックを登録します（必要な場合）。
2) sf_touch_panel_v2_api_t::open API 関数を使用してタッチパネル V2 フレームワークモジュールを初期化します（[auto initialization] プロパティが有効になっている場合は、SSP で自動的に行われます）。
3) sf_touch_panel_v2_api_t::calibrate API 関数を使用して、タッチデータの較正を行うことができます（必要な場合）。
4) sf_touch_panel_v2_api_t::start API 関数を使用してタッチパネルフレームワークモジュールを開始し、タッチパネルチップからのタッチデータのスキャンを開始します（[auto start] プロパティが有効になっている場合は、SSP で自動的に行われます）。
5) タッチイベントが発生すると、登録されたユーザーコールバックがフレームワークによって呼び出されて、アプリケーションコードで処理されます。
6) sf_touch_panel_v2_api_t::touchDataGet API 関数を使用してデータを取得します（ユーザーコールバックが登録されていない場合）。
7) アプリケーションコードを使用して、必要に応じて受信したタッチデータを操作します。
8) sf_touch_panel_v2_api_t::stop API 関数を使用してタッチデータの受信を終了します。
9) sf_touch_panel_v2_api_t::close API 関数を使用して、モジュールを閉じます（必要な場合）。

これらの一般的な手順を、次の図の通常の動作フローに示します。

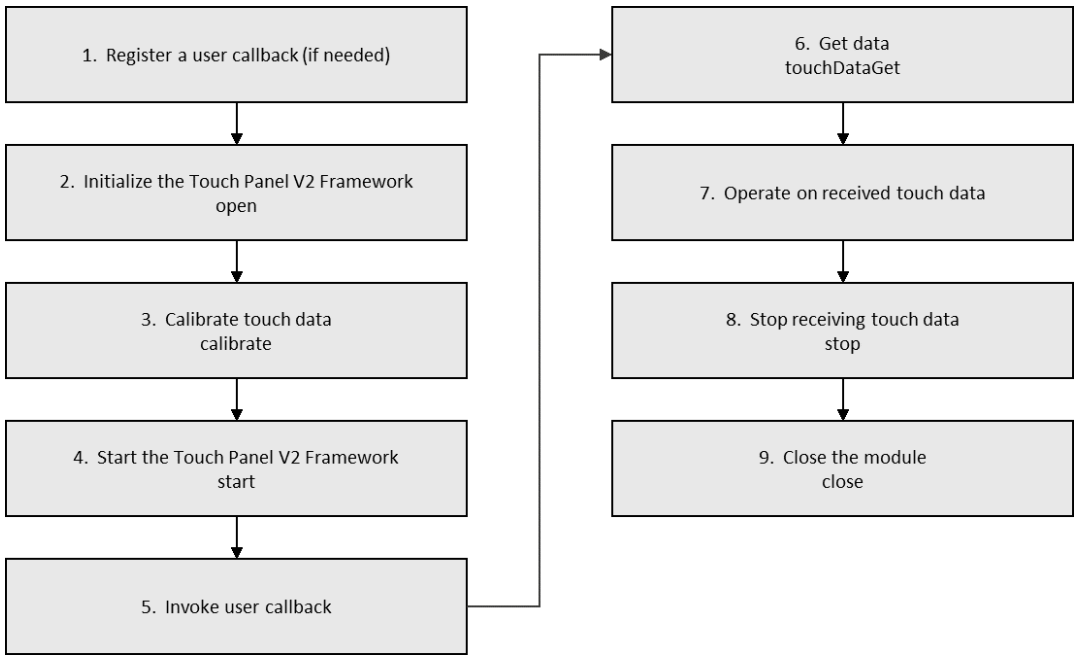


図 130:通常のタッチパネル V2 フレームワークモジュールアプリケーションのフロー図

4.1.33 UART 通信フレームワーク

UART 通信フレームワークは、UART 準拠の Synergy MCU ペリフェラル上で業界標準の UART プロトコルをサポートするシリアル通信用のハイレベル API を実装します。このフレームワークでは、r_sci_uart HAL ドライバーを使用し、UART モードで Synergy MCU の SCI ペリフェラルを操作します。

4.1.33.1 UART 通信フレームワークモジュールの特長

このモジュールは ThreadX 対応通信フレームワークです。ThreadX オブジェクトを使用して動作がスレッドセーフになるようにします。

主な特長は次のとおりです。

- UART 通信プロトコルのサポート
- 専用アクセスを確保するためのチャネルロックのサポート
- ThreadX 対応実装

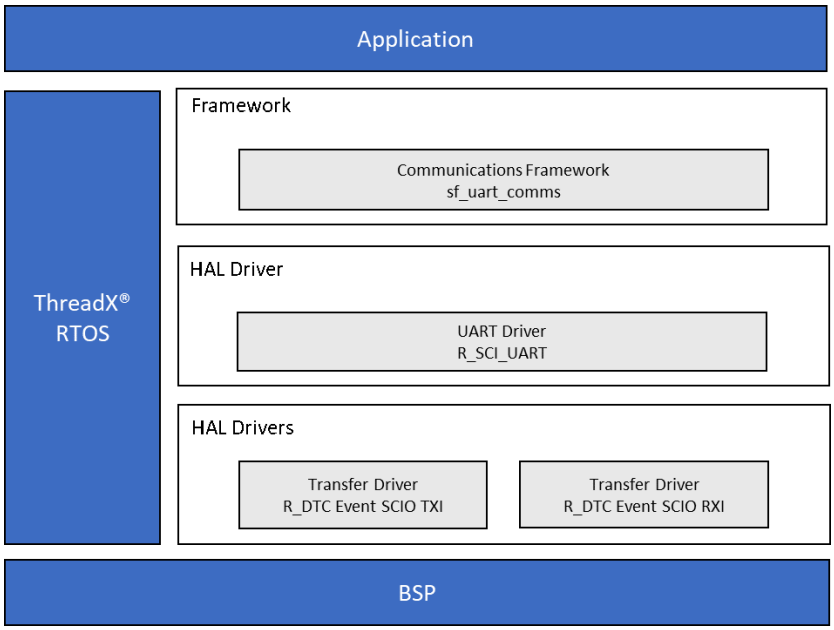


図 131:UART 通信フレームワークモジュールのブロック図

4.1.33.2 UART 通信フレームワークモジュールの API の概要

UART 通信フレームワークモジュールは、通信チャネルのオープン、クローズ、リード、ライトの API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

UART 通信フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_comms0.p_api->open(g_sf_comms0.p_ctrl, g_sf_comms0.p_cfg); Initialize communications driver.
close	g_sf_comms0.p_api->close(g_sf_comms0.p_ctrl); Clean up communications driver.
read	g_sf_comms0.p_api->read(g_sf_comms0.p_ctrl, &destination, bytes, timeout); Read data from communications driver. This call will return after the number of bytes requested is read or if a timeout occurs while waiting for access to the driver.
write	g_sf_comms0.p_api->write(g_sf_comms0.p_ctrl, &source, bytes, timeout); Write data to communications driver. This call will return after all bytes are written or if a timeout occurs while waiting for access to the driver.
lock	g_sf_comms0.p_api->lock(g_sf_comms0.p_ctrl, lock_type, timeout); Lock the communications driver. Reserve exclusive access to the communications driver.
unlock	g_sf_comms0.p_api->unlock(g_sf_comms0.p_ctrl, lock_type); Unlock the communications driver. Release exclusive access to the communications driver.

Function Name	Example API Call and Description
versionGet	g_sf_comms0.p_api->version(&version); Store the driver version in the provided version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Channel opened successfully.
SSP_ERR_IN_USE	Channel already in use.
SSP_ERR_ASSERTION	Pointer to UART control block or configuration structure is NULL.
SSP_ERR_HW_LOCKED	Channel is locked.
SSP_ERR_INVALID_MODE	Channel is used for non-UART mode or illegal mode is set.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter setting found in the configuration structure.
SSP_ERR_QUEUE_UNAVAILABLE	Cannot open transmit or receive queue or both.
SSP_ERR_INTERNAL	Internal error occurs.
SSP_ERR_TIMEOUT	Timeout error.
SSP_ERR_INSUFFICIENT_DATA	Not enough data in receive circular buffer.
SSP_ERR_RXBUF_OVERFLOW	Receive queue overflow.
SSP_ERR_OVERFLOW	Hardware overflow.
SSP_ERR_FRAMING	Framing error.
SSP_ERR_PARITY	Parity error.

Name	Description
SSP_ERR_INSUFFICIENT_SPACE	Not enough space in transmission circular buffer.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.1.33.3 UART 通信フレームワークモジュールの動作の概要

UART フレームワークによって、標準 UART プロトコルを使用した、簡単に使用できる通信フレームワークが提供されます。UART デバイスのデータをスレッドセーフ方式でリードおよびライトを実行するためのハイレベル関数に加え、このフレームワークでは、アプリケーションが UART チャネルをスレッドに対してロック（およびロック解除）するための API 関数も提供されます。これが特に役立つのは、複数のアプリケーションスレッドが同じ UART デバイスとの通信を試行するときに、コンテキストスイッチのために、UART に実装されているハイレベルのアプリケーションプロトコルまたは状態マシン（あるいは両方）（たとえば Kermit）で異常が発生する可能性がある場合です。

UART 通信フレームワークモジュールの動作に関する重要な注意事項と制限事項

- UART フレームワークモジュールはすべてのチャネルで再入可能です。
- UART フレームワークは、UART デバイスと通信するために r_sci_uart モジュール上の UART ドライバーを使用します。(ISDE で Synergy プラットフォームコンフィギュレータを使用して) UART ドライバーに DTC ドライバーを追加して拡張すると、CPU の割り込みを行わずに、UART デバイスに対するリードトランザクションまたはライトトランザクションを実行できます。UART フレームワークを使用して UART デバイスからデータを読み取るとき、UART ドライバーのコールバック機能が利用され、DTC は使用されません（DTC モジュールのサポートがコンフィギュレータで UART ドライバーに追加されている場合でも）。これは、ドライバーが DTC を使用してデバイスのデータを読み取る際に発生する可能性があるタイミングや同期の問題を回避するためです。UART フレームワークを使用して UART デバイスにデータを書き込むときは、トランザクションを実行するために DTC が使用されます（ドライバーが DTC を使用するように構成されている場合）。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.33.4 アプリケーションへの UART 通信フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに UART 通信フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

UART 通信フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(UART 通信フレームワークのデフォルト名は g_sf_comms0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

UART 通信フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_comms0 Communications Framework on sf_uart_comms	Threads	Framework > Connectivity > Communications Framework on sf_uart_comms

次の図に示すように、sf_uart_comms 上の UART 通信フレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

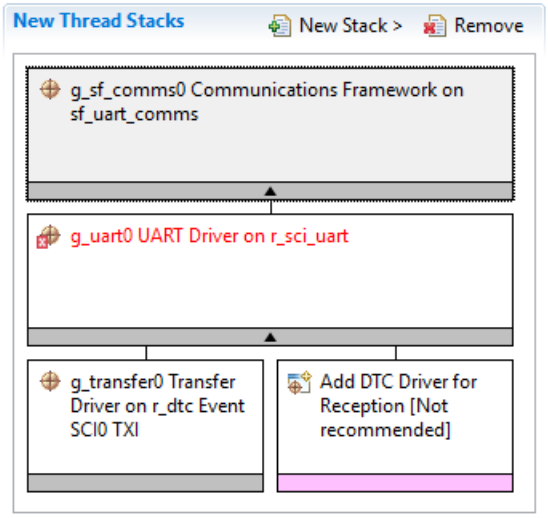


図 132:UART 通信フレームワークモジュールのスタック

4.1.33.5 UART 通信フレームワークモジュールの構成

ユーザーは必要な動作に合わせて、UART 通信フレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_uart_comms 上の UART 通信フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if parameter checking is included.
Read Input Queue Size (4-Byte Words)	15	Buffer size for data reception queue. sf_uart_comms utilizes the ThreadX Queue for the queue management.
Name	g_sf_comms0	Name of UART communications framework module.
Name of generated initialization function	sf_comms_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

UART 通信フレームワークモジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_sci_uart の UART HAL モジュールの構成設定

ISDE Property	Value	Description
External RTS Operation	Enable, Disable Default: Disable	Enable an IOPORT pin to be used as RTS signal. For RTS functionality set this configuration parameter to "Enable" and specify the configuration "Name of UART callback function for the RTS external pin control".
Reception	Enable, Disable Default: Enable	Enable or disable UART reception for all UART channels on SCI. Setting this configuration parameter to "Disable" reduces code size because the portion of code for UART reception is not compiled. You cannot set this parameter for individual UART channels.
Transmission	Enable, Disable Default: Enable	Enable or disable UART transmission for all UART channels on SCI. Setting "Disable" to this configuration allows to get smaller code size due to the portion of code for UART transmission is compiled out, however, you can only set "Disable" to this configuration if any other SCI channels which work as UART ports do not perform the transmission.
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_uart0	The name to be used for UART on SCI module control block instance. This name is also used as the prefix of the other variable instances.

ISDE Property	Value	Description
Channel	0	SCI channel number.
Baud Rate	9600	Baud rate selection.
Data Bits	7 bits, 8, bits, 9 bits Default: 8 bits	UART data bits.
Parity	None, Odd, Even Default: None	UART parity bits.
Stop Bits	1 bit, 2 bits Default: 1 bit	UART stop bits.
CTS/RTS Selection	CTS (Note that RTS is available when enabling External RTS Operation mode which uses 1 GPIO pin), RTS (CTS is disabled) Default: RTS (CTS is disabled)	Select CTS or RTS for the CTSn/RTSn pin of SCI channel n. The SCI hardware supports either the CTS or the RTS control signal on this pin but not both. For an application that uses both CTS and RTS, select "CTS" for this configuration parameter and enable the configuration "External RTS Operation" specifying the configuration "Name of UART callback function for the RTS external pin control".
Name of UART callback function to be defined by user	NULL	Name must be a valid C symbol.
Name of UART callback function for the RTS external pin control to be defined by user	NULL	Name must be a valid C symbol.

ISDE Property	Value	Description
Clock Source	Internal Clock, External Clock 8x baudrate, External Clock 16x baudrate Default: Internal Clock	Selection of the clock source to be used in the baud-rate clock generator block.
Baudrate Clock Output from SCK pin	Enable, Disable Defaualt: Disable	Optional setting to output the baud-rate clock on the SCKn pin for the selected channel n.
Start bit detection	Fallling Edge, Low Level Default: Falling Edge	Start bit detection mode in the reception, usually set "Falling Edge" to this configuration.
Noise Cancel	Enable, Disable Defaualt: Disable	Enable the digital noise cancellation on RXDn pin. The digital noise filter block in SCI consists of two-stage flip-flop circuits. For detail, refer to the Noise cancellation section in the Renesas Synergy hardware manual.
Bit Rate Modulation Enable	Enable, Disable Defaualt: Enable	Bit rate modulation enable selection.
Receive FIFO Trigger Level	One, Max Default: Max	Receive FIFO trigger level selection.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Receive interrupt priority selection.

ISDE Property	Value	Description
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit interrupt priority selecition.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit end interrupt priority selection.
Error Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Error interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Defaualt: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection

ISDE Property	Value	Description
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SCI0 TXI	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SPI0 RXI	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection

ISDE Property	Value	Description
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

UART 通信フレームワークモジュールのクロック構成

UART 通信フレームワークには、固有のクロック構成の要件はありません。

UART 通信フレームワークモジュールのピン構成

UART 通信フレームワークは、MCU のピンを使用し、選択したローレベル実装に基づいて外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はローレベル実装のピンの選択例を示しています。

注 : 一部のペリフェラルでは、動作モードの選択によって使用可能なペリフェラル信号が決まり、それに従って必要な MCU ピンが決定されます。

sf_uart_comms 上の UART 通信フレームワークモジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
UART	Pins	Select Peripherals > Connectivity: SCI > SCI8

sf_uart_comms 上の UART 通信フレームワークモジュールのピン構成設定

Property	Value	Description
Pin Group Selection	Mixed, _A Only, _B Only	Pin group selection
Operation Mode	Disabled, Custom, Asynchronous UART, Synchronous UART, Simple I2C, Simple SPI, SmartCard Default: Disabled	Select Asynchronous UART as the Operation Mode for a UART Receiver implementation

Property	Value	Description
TXD_MOSI	None, PB05, P105 Default: None	TXD Pin P105
RXD_MISO	None, PB05, P104 Default: None	RXD Pin P104

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

UART、USB、または Telnet（トランスミッタ）通信フレームワークのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
UART	Pins	Select Peripherals > Connectivity: SCI > SCI3

注 : これらの選択シーケンスは選択した実装に対応する例です。ターゲットハードウェアによっては他の選択シーケンスも可能です。

UART のピン構成設定

Pin Configuration Property	Settings	Description
Pin Group Selection	Mixed, _A Only, _B Only	Pin group selection
Operation Mode	Disabled, Custom, Asynchronous UART, Synchronous UART, Simple I2C, Simple SPI, SmartCard Default: Disabled	Select Asynchronous UART as the Operation Mode for a UART Transmitter implementation
TXD_MOSI	None, P707, P409 Default: None	TXD Pin P707

Pin Configuration Property	Settings	Description
RXD_MISO	None, P706, P408 Default: None	RXD Pin P706

注 : これらの選択シーケンスは選択した実装に対応する例です。ターゲットハードウェアによっては他の選択シーケンスも可能です。

4.1.33.6 アプリケーションでの UART 通信フレームワークモジュールの使用

一般的なアプリケーションで sf_audio_record_adc 上の UART 通信フレームワークモジュールを使用する際の手順は次のとおりです。

1) sf_comms_api_t::open API を使用して UART 通信フレームワークを初期化します。
2) sf_comms_api_t::lock API を使用して、連続通信のためにチャネルをロックします（必要な場合）。
3) sf_comms_api_t::read API を使用して、データを受信します。
4) sf_comms_api_t::write API を使用して、データを送信します。
5) sf_comms_api_t::unlock API を使用して連続通信用のチャネルのロックを解除します（必要な場合）。
6) sf_comms_api_t::close API を使用して、チャネルを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

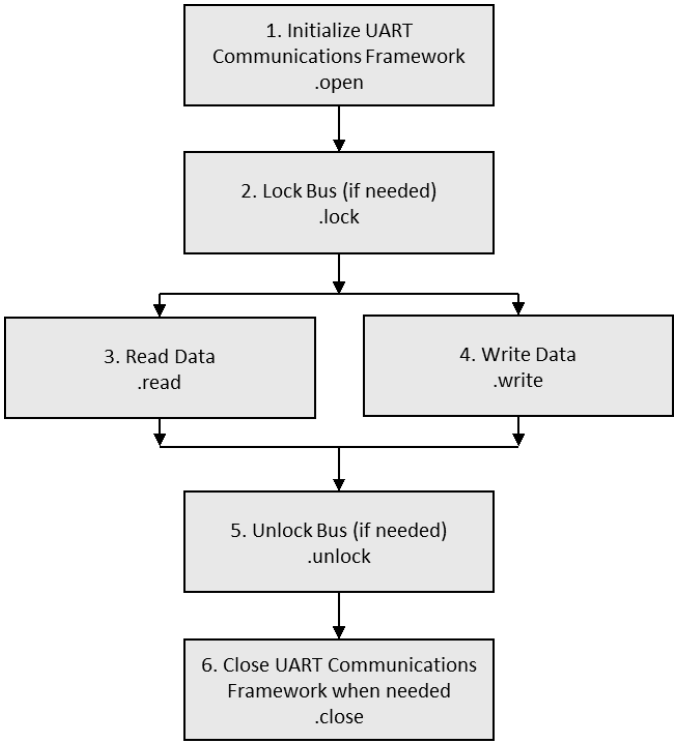


図 133:通常の UART 通信フレームワークモジュールアプリケーションのフロー図

4.1.34 Wi-Fi フレームワーク

Wi-Fi フレームワークが提供するハイレベルの API モジュールによって、Wi-Fi モジュールの構成とプロビジョニングを行い、オンチップネットワーク機能の有無にかかわらずデータ転送を実行できます。現在、Qualcomm GT202 モジュールのみがサポートされています。Wi-Fi フレームワークは、SPI を介して基礎となる GT202 モジュールと通信します。

4.1.34.1 WiFi フレームワークモジュールの特長

- WiFi モジュールを構成し、プロビジョニングするための高レベルの API を提供する
- 以下に対して 4 つの異なる実装を提供する :
  - sf_wifi_gt202 フレームワークを使用した Wi-Fi デバイスドライバースタック
  - sf_wifi_onchip_stack フレームワークを使用したオンチップスタック
  - sf_wifi_onchip_stack フレームワークを使用した BSD ソケットスタック
  - sf_wifi_nsal_nx フレームワークを使用した NetX および NetX Duo ポート
- NetX および NASL の使用 :
  - 同じアプリケーションコードを異なる Wi-Fi モジュールに使用可能

  - イーサネットベースのアプリケーションを Wi-Fi ベースのアプリケーションに容易に移行可能
  - アプリケーションの要件に従ってアプリケーションと TCP/IP スタックのデバッグと微調整が可能。
  - 現在の NSAL 実装では、NetX NSAL のみを提供。新しいネットワークスタックのサポートを追加するには、適切な NSAL の実装が必要です。
- オンチップネットワークスタックの使用 :
  - MCU を小さなメモリフットプリントで使用する場合に有益
  - オンチップ TCP/UDP を使用してソケットベースのアプリケーションを作成するための BSD ソケットインタフェースを提供
- NetX スタックを使用せずに TCP/IP 上の MQTT および COAP などのサードパーティアプリケーションプロトコルを統合するオプションを提供。

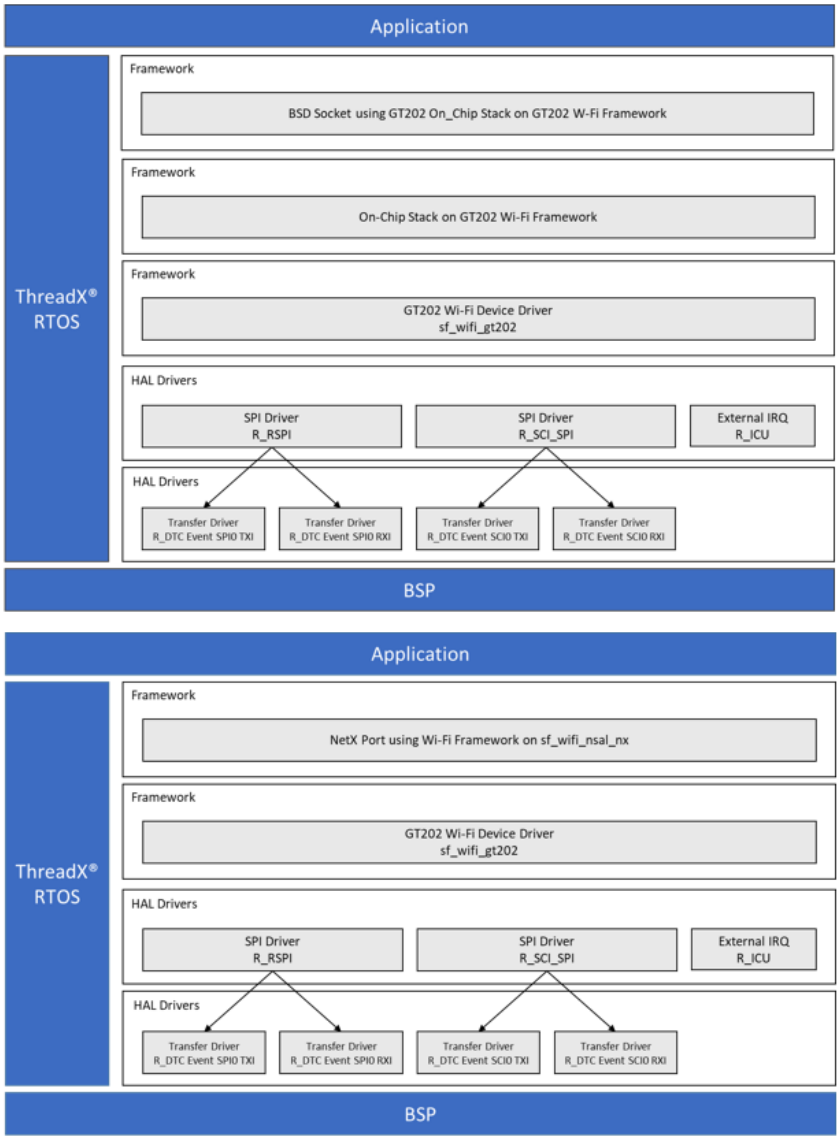


図 134:WiFi フレームワークモジュールのブロック図

注 :GT202 Wi-Fi フレームワーク上のオンチップスタックは、BSD ソケットフレームワークのローレベル実装として、または単体で使用できます。GT202 Wi-Fi デバイスドライバーとそのローレベルモジュールは、その他すべての Wi-Fi フレームワーク実装の下に含まれています。

4.1.34.2 WiFi フレームワークモジュールの API の概要

Wi-Fi フレームワークモジュールは、関連する各モジュールの API 関数を定義します。この後で各 API の動作について説明します。一般的なリターンコードの表はセクションの終わりに記載されています。その他の詳細については、対応するモジュールの API リファレンスセクションを参照してください。

さらに、Renesas Web サイトには、API の詳しい説明と、実際に動作するサンプルアプリケーション（この例は長いため本書には記載されていません）が掲載されています。次のサイトのトップページに表示される検索バーで、関連するアプリケーションノートのドキュメント番号 r11an0226eu を検索してください：http://www.Renesas.com. 本書で要約に示されている説明の補足としてアプリケーションノートを使用することを強く推奨します。

Wi-Fi フレームワークモジュールの API

WiFi フレームワークモジュール API の要約

Function Name	Example API Call and Description
open	g_sf_wifi0.p_api->open (g_sf_wifi0.p_ctrl, g_sf_wifi0.p_cfg); This API initializes and enables the Wi-Fi module. The open function returns the Wi-Fi control structure, uniquely identifying the instance of the Wi-Fi framework.
close	g_sf_wifi0.p_api->close(g_sf_wifi0.p_ctrl); This API un-initializes the Wi-Fi module and powers it off.
infoGet	g_sf_wifi0.p_api->infoGet (g_sf_wifi0.p_ctrl, wifi_info); This API takes the WiFi control structure as an argument and returns the following information obtained from the WiFi module: - Chipset/driver information string - RSSI value (unsigned integer 16 bits) - Noise level (unsigned integer 16 bits) - Link Quality (unsigned integer 16 bits)

Function Name	Example API Call and Description
statisticsGet	g_sf_wifi0.p_api->statisticsGet (g_sf_wifi0.p_ctrl, p_stats); This API gets the data statistics from the Wi-Fi module. It takes the Wi-Fi control structure as an argument and returns the following statistics: - Received packets (unsigned integer 32 bits) - Transmitted packets (unsigned integer 32 bits) - Transmit packet errors (unsigned integer 32 bits)
transmit	g_sf_wifi0.p_api->transmit (g_sf_wifi0.p_ctrl, p_buffer, length); This API sends the data/packet out. This function takes the Wi-Fi control structure as an argument. It takes the network packet buffer, and the network packet buffer length as arguments. The Wi-Fi framework transmit function passes the packet buffer to the Wi-Fi driver for transmission.
provisioningGet	g_sf_wifi0.p_api->provisioningGet (g_sf_wifi0.p_ctrl, &sf_wifi_provisioninfo); This API takes the Wi-Fi control structure as an argument and returns the following parameters: - Mode (enumeration, that is, AP or client) - Channel (unsigned integer 8 bits) - SSID (string) - Security type (enumeration) - Encryption type (enumeration) - Security key (string)

Function Name	Example API Call and Description
provisioningSet	g_sf_wifi0.p_api->provisioningSet (g_sf_wifi0.p_ctrl, &g_sf_wifi_provisioninfo); This API sets the Wi-Fi module in the given mode AP/Station. The provisioningSet function uses the following parameters to provision the WiFi module: - Mode (enumeration, that is, AP or client) - Channel (unsigned integer 8 bits), only used in AP mode - SSID (string) - Security type (enumeration) - Encryption type (enumeration) - Security key (sting). - Connection Status Notification Callback function: Used to get connection status change notification *See note at the end of this table.
scan	g_sf_wifi0.p_api->scan (g_sf_wifi0.p_ctrl, p_scan, p_count); This API scans the available SSIDs (that is, access points) in range. This function takes the Wi-Fi control structure as an argument and returns a list of SSIDs scanned by the WiFi module with the following parameters: - HW mode (enumeration a/b/g/n) - RSSI (unsigned integer 16 bits) - SSID (string) - BSSID (byte array of size 6 bytes) - Channel (unsigned integer 8 bits) - Security type (enumeration) - Encryption type(enumeration) - BSS type (enumeration) The Wi-Fi framework scan function takes the SSID count as an argument, which acts as an in/out parameter. It specifies the size of the scan result array and the Wi-Fi framework sets it to count the indicating number of scan results stored in the array.

Function Name	Example API Call and Description
ACLAdd	g_sf_wifi0.p_api->ACLAdd (g_sf_wifi0.p_ctrl, peer_device_mac); This API adds the given MAC address to the access control list. This function takes the Wi-Fi control structure and the MAC address as arguments.
ACLDelete	g_sf_wifi0.p_api->ACLDelete (g_sf_wifi0.p_ctrl, peer_device_mac); This API deletes the given MAC address from the access control list. This function takes the Wi-Fi control structure and MAC address as arguments.
multicastListAdd	g_sf_wifi0.p_api->multicastListAdd (g_sf_wifi0.p_ctrl, p_mac_addr); This API adds the given Multicast IP address to the multicast filer list. This function takes the Wi-Fi control structure and MAC address as arguments.
multicastListDelete	g_sf_wifi0.p_api->multicastListDelete (g_sf_wifi0.p_ctrl, p_mac_addr); This API deletes the given Multicast IP address from the multicast filer list. This function takes the Wi-Fi control structure and MAC address as arguments.
macAddressGet	g_sf_wifi0.p_api->macAddressGet (g_sf_wifi0.p_ctrl, p_mac); This API reads MAC address from WiFi module. This function takes the Wi-Fi control structure as argument and returns MAC address read from Wi-Fi module.

Function Name	Example API Call and Description
macAddressSet	g_sf_wifi0.p_api->macAddressSet (g_sf_wifi0.p_ctrl, p_mac); This API sets Wi-Fi module's MAC address. This function takes the Wi-Fi control structure and MAC address as arguments.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

注 : プロビジョニング：デバイスがクライアントモードでプロビジョニングされる場合は、AP との接続が切断されてから再度確立されるときにコールバックがコールされます。デバイスが AP モードでプロビジョニングされる場合は、クライアントが AP と接続するとき、または AP から切断するときにこのコールバックがコールされます。デバイスがクライアントモードでプロビジョニングされる場合は、コールバックに渡される引数には、次の有効なフィールドのみが含まれます。

注 :- event = SF_WIFI_EVENT_AP_CONNECT または SF_WIFI_AP_DISCONNECT

注 : デバイスが AP モードでプロビジョニングされる場合は、コールバックに渡される引数には、次の有効なフィールドのみが含まれます。注 :- event = SF_WIFI_EVENT_CLIENT_CONNECT または SF_WIFI_CLIENT_DISCONNECT

注 :- mac_addr = クライアントの MAC アドレス。

注 :the sf_wifi_api_t::provisioningSet API 関数をコールしてクライアントモードでデバイスをプロビジョニングする間、AP との接続の成功または失敗に関係なく、フレームワークはコールバック関数をコールしません。デバイスが AP モードでプロビジョニングされる場合、クライアントが誤ったパスワードを使用して AP との接続を試みると、コールバックは 2 回コールされます。最初は接続されたイベントによってコールされ、その直後には切断されたイベントによってコールされます。

オンチップネットワークスタックサポート API

これらの API を使用して、オンチップネットワークスタックを使用するときに Wi-Fi モジュールを構成できます。これは、インタフェースの IP アドレスの構成と、DHCP サーバーの起動と停止（AP モードで構成されている場合）に役立ちます。

オンチップネットワークスタックサポート Wi-Fi フレームワークモジュールの API の要約

Function Name	Example API Call and Description
open	g_sf_wifi_onchip_stack0.p_api->open (g_sf_wifionchip_stack0.p_ctrl, g_sf_wifi_onchip_stack0.p_cfg); This API calls the WiFi framework open API which initializes the Wi-Fi module.

Function Name	Example API Call and Description
close	g_sf_wifi_onchip_stack0.p_api->close(g_sf_wifi_onchip_stack0.p_ctrl); This API calls the Wi-Fi framework close API which un-initializes the Wi-Fi module.
ipAddressCfg	g_sf_wifi_onchip_stack0.p_api->ipAddressCfg (g_sf_wifi_onchip_stack0.p_ctrl, p_cfg); This API configures the IP address of the interface using an on-chip networking stack. It provides facility to configure static IP address or using DHCP.
dhcpServerStart	g_sf_wifi_onchip_stack0.p_api->dhcpServerStart (g_sf_wifi_onchip_stack0.p_ctrl, p_start_ip, p_end_ip); This API starts the DHCP server on the interface (when configured in AP mode) using on-chip networking stack. It takes the range of IP addresses to be used by DHCP server.
dhcpServerStop	g_sf_wifi_onchip_stack0.p_api->dhcpServerStop (g_sf_wifi_onchip_stack0.p_ctrl); This API stops the DHCP server.
versionGet	g_sf_wifi_onchip_stack0.p_api->versionGet (&version); Retrieves the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

BSD ソケット API

これらの API は、GT202 オンチップスタック実装を使用した BSD ソケットサポートに使用できます。

GT202 オンチップスタック Wi-Fi フレームワークモジュールを使用した BSD ソケットの API の要約

Function Name	Example API Call and Description
open	g_sf_socket0.p_api->open (g_sf_socket0.p_ctrl, g_sf_socket0.p_cfg); This API initializes the networking interface.
close	g_sf_socket0.p_api->close(g_sf_socket0.p_ctrl); This API closes the network interface.
versionGet	g_sf_socket0.p_api->versionGet (&version); Retrieves the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

この実装には、BSD API に準拠したソケット API が含まれます。これらの API は、アプリケーションがソケットを使用したデータ転送を実行するために使用できます。次の API を使用できます。

- socket
- close
- bind
- listen
- accept
- connect
- send
- recv
- recvfrom
- sendto
- setsockopt
- getsockopt
- select

注 : アプリケーションは、オンチップネットワークスタックを使用するときに、すべての BSD ソケット API、すべてのオンチップネットワークスタックサポート API、およびいくつかの Synergy Wi-Fi フレームワーク API を使用できます。アプリケーションで使用できる Synergy Wi-Fi フレームワーク API は、provisioningSet()、provisioningGet()、scan()、macAddressGet()、macAddressSet()、infoGet() です。

これらの API の詳細については、『NetX BSD 4.3 Sockets API Compliancy Wrapper for NetX User Guide』を参照してください。このガイドは、SSP ページの Synergy ギャラリーで、[documentation] タブに掲載されている Renesas Synergy の X-Ware ™コンポーネントドキュメントの zip ファイルにあります。

Wi-Fi NSAL

Synergy Wi-Fi フレームワークは、NetX または NetX-Duo ネットワークサービス抽象化レイヤーをサポートします。これらには、NetX または NetX-Duo ドライバー、パケット送信および受信コールバック関数の実装が含まれます。

NetX または NetX-Duo ドライバー関数

NetX または NetX-Duo ドライバー関数は NetX IP インスタンス、WiFi フレームワークインスタンス、NSAL 構成を引数として受け取ります。NSAL 構成は、送信および受信コールバック関数の動作を制御します。NSAL 構成には、ゼロコピーサポートが送信および受信パスで有効か無効かを示すフラグが含まれます。NetX または NetX-Duo ドライバー関数は、NetX または NetX-Duo で使用される各種 IP ドライバーコマンドを実装します。interface attach コマンドは、WiFi フレームワークの open API をコールして Wi-Fi モジュールを初期化します。initialize コマンドは、Wi-Fi フレームワークの macAddressGet API をコールして、WiFi モジュールから MAC アドレスを読み取ります。un-initialize コマンドは、WiFi フレームワークの close API を呼び出して WiFi モジュールを初期化解除します。multicast-join コマンドは、Wi-Fi フレームワークの multicastListAdd API をコールして、特定の MAC アドレスをマルチキャストリストに追加します。multicast-leave コマンドは、Wi-Fi フレームワークの multicastListDelete API をコールして、特定の MAC アドレスをマルチキャストリストから削除します。Send および Broadcast コマンドは、WiFi フレームワークの transmit API をコールしてパケットを送信します。

NSAL 送信関数

NSAL transmit 関数は NetX IP インスタンス、NetX パケット、Wi-Fi フレームワークインスタンス、NSAL 構成を引数として受け取ります。ゼロコピーサポートが有効になっている場合、NetX パケットは NetX から WiFi ドライバーに転送されます。ゼロコピーがサポートされていない場合は、NetX パケットからドライバーバッファにデータをコピーします。バッファやパケットを Wi-Fi ドライバーに渡してさらに送信する Wi-Fi フレームワークの transmit API をコールします。

NSAL 受信コールバック

NSAL 受信コールバック関数は、NetX IP インスタンス、パケットバッファ、パケットバッファ長、NSAL 構成を引数として受け取ります。このコールバックは Wi-Fi デバイスドライバーからコールされます。ゼロコピーサポートが有効になっている場合、NetX パケットは WiFi ドライバーから NetX に転送されます。ゼロコピーがサポートされていない場合は、ドライバーバッファから NetX パケットにデータをコピーし、さらに処理するために NetX スタックに渡します。バッファを Wi-Fi ドライバーに渡してさらに送信する Wi-Fi フレームワークの transmit API をコールします。

これらの API の詳細については、『NetX ユーザーガイド』を参照してください。このガイドは、SSP ページの Synergy ギャラリーで、[documentation] タブに掲載されている Renesas Synergy の X-Ware ™コンポーネントドキュメントの zip ファイルにあります。

Wi-Fi フレームワークのエラーコード

次の表に、Wi-Fi フレームワーク固有のエラーコードを示します。これらのエラーコードは ssp_err_t. に含まれます。

Wi-Fi フレームワークのエラーコード

Error Codes	Description
SSP_ERR_WIFI_CONFIG_FAILED	Configuration failed
SSP_ERR_WIFI_INIT_FAILED	Initialization failed

Error Codes	Description
SSP_ERR_WIFI_TRANSMIT_FAILED	Transmission failed
SSP_ERR_WIFI_INVALID_MODE	Invalid mode specified
SSP_ERR_WIFI_FAILED	Wifi failed

4.1.34.3 WiFi フレームワークモジュールの動作の概要

Wi-Fi フレームワークは、Wi-Fi モジュールの構成と Wi-Fi モジュールのプロビジョニングを行ってデータ転送を実行するアプリケーションのためにハイレベルインタフェースを提供します。これにより、アプリケーション開発が簡素化され、同じアプリケーションコードをさまざまな Wi-Fi モジュールで使用できます。

次の図に、Synergy Wi-Fi フレームワーク階層化アーキテクチャの概要を示します。

- Wi-Fi フレームワークには、NSAL、Wi-Fi フレームワーク API、Wi-Fi オンチップスタック API、BSD ソケット API、および Wi-Fi デバイスドライバーインタフェースという、アーキテクチャ図の中央で囲まれている 5 つのブロックが含まれます。
- ベンダ提供のWi-Fiデバイスドライバーは、SSP_Supplemental.の下のSSPパッケージに含まれています。

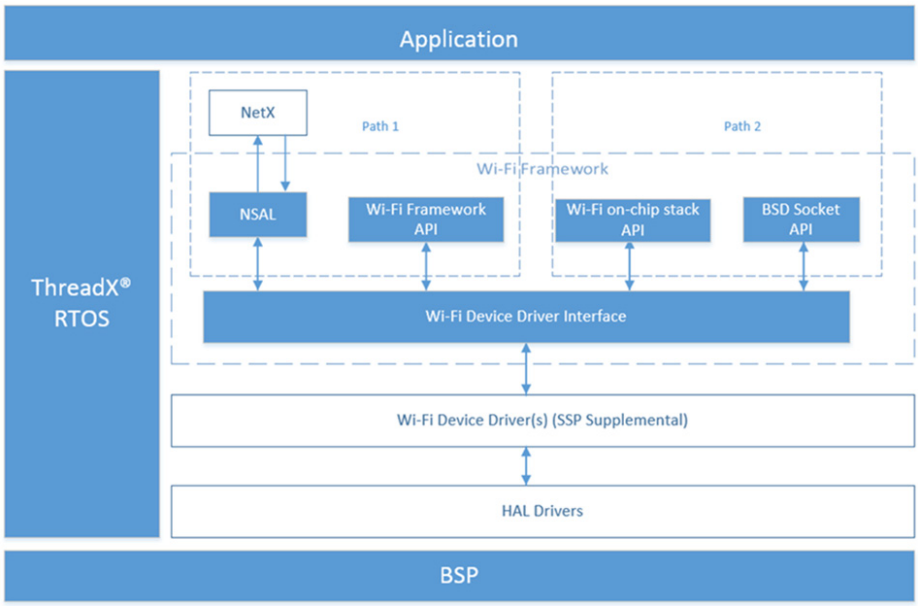


図 135:**WiFi フレームワークの編成、オプション、スタックの実装**

Wi-Fi フレームワークの実装では、オンチップネットワークスタックサポートの有無にかかわらず、Wi-Fi モジュールと SSP ローレベルサポートブロックとの統合を行うことができます。

- パス 1：前の図で示したとおりに、Wi-Fi フレームワーク API ブロックに加えて NetX ™または NetX Duo ™と、NSAL を使用します。説明を簡素にするために、本書ではコメントが NetX と NetX-Duo の両方に当てはまる場合は、まとめて NetX と表記します。

- パス 2：前の図で示したとおりに、オンチップネットワークスタックサポート API と BSD ソケット API を使用します。

WiFi フレームワークモジュールの動作に関する重要な注意事項と制限事項

- Wi-Fi モジュールには、802.11 規格によって指定されたさまざまなパラメータがあります。個別のデバイスドライバーおよび Wi-Fi チップセットですべての関数の構成がサポートされるとは限りません。
- Wi-Fi インタフェースがアクティブになるためには、少なくともチャネル、サービスセット識別子（SSID）、セキュリティスキーム、セキュリティ資格情報を構成する必要があります。
- 現在の NSAL 実装には、NetX（IPv4）と NetX-Duo（IPv6）のサポートが含まれます。NetX と NetX Duo は IPv4 をサポートしています。また、NetX Duo は IPv6 もサポートしています。新しいネットワークスタックのサポートを追加するには、適切な NSAL の実装が必要です。
- セキュリティ設定については、WEP キーは ASCII 形式または Hex 形式で入力できます。また、40 ビットキーまたは 104 ビットキーを使用するよう構成できます。WEP キーは、秘密キーと 24 ビットの初期化ベクタで構成されます。そのため、ベンダによっては、64 ビットの WEP キーを 40 ビットキー、128 ビットの WEP キーを 104 ビットキーと呼んでいる場合があります。Wi-Fi フレームワークでは、特定の形式およびタイプの WEP キーを 1 ～ 4 個許可します。プロビジョニング構造体では、セキュリティタイプを SF_WIFI_SECURITY_TYPE_WEP として指定し、最低 1 つ（最大 4 つ）の WEP キーをキーバッファに入力する必要があります。

Wi-Fi フレームワークモジュールの API の使用に関する注意事項

Open：

Wi-Fi フレームワークモジュールと NSAL（つまり、NetX または NetX Duo）を併用する場合、アプリケーションでは Wi-Fi フレームワークモジュール sf_wifi_api_t::open API を直接コールしないでください。代わりに、内部で NetX ドライバーから sf_wifi_api_t::open API をコールする NetX nx_ip_create() API をコールする必要があります。

オンチップネットワークスタックを使用する場合は、アプリケーションは BSD ソケットインタフェースから sf_wifi_api_t::open API をコールする必要があります。この API は内部で Wi-Fi フレームワーク sf_wifi_api_t::open API をコールします。

Close：

Wi-Fi フレームワークモジュールと NSAL（つまり、NetX）を併用する場合、アプリケーションでは Wi-Fi フレームワークモジュール sf_wifi_api_t::close API を直接コールしないでください。代わりに、内部で NetX ドライバーから close をコールする NetX nx_ip_delete() API をコールする必要があります。

オンチップネットワークスタックを使用する場合は、アプリケーションは BSD ソケットインタフェースから sf_wifi_api_t::close API をコールする必要があります。この API は内部で Wi-Fi フレームワークモジュール sf_wifi_api_t::close API をコールします。

ProvisioningSet：

デバイスがクライアントモードでプロビジョニングされる場合は、AP との接続が切断されて再度確立されるときに、コールバックがコールされます。デバイスが AP モードでプロビジョニングされる場合は、クライアントが AP と接続するとき、または AP から切断するときにこのコールバックがコールされます。

デバイスがクライアントモードでプロビジョニングされる場合は、コールバックに渡される引数には、以下の有効なフィールドのみが含まれます。

event = SF_WIFI_EVENT_AP_CONNECT または SF_WIFI_AP_DISCONNECT

デバイスが AP モードでプロビジョニングされる場合は、コールバックに渡される引数には、以下の有効なフィールドのみが含まれます。

event = SF_WIFI_EVENT_CLIENT_CONNECT または SF_WIFI_CLIENT_DISCONNECT

mac_addr = クライアントの MAC アドレス。

sf_wifi_api_t::provisioningSet API をコールしてクライアントモードでデバイスをプロビジョニングする間、AP との接続の成功または失敗に関係なく、フレームワークはコールバック関数をコールしません。

デバイスが AP モードでプロビジョニングされる場合、クライアントが誤ったパスワードを使用して AP との接続を試みると、コールバックは 2 回コールされます。最初は接続されたイベントによってコールされ、その直後には切断されたイベントによってコールされます。

InfoGet：

デバイスがクライアントモードでプロビジョニングされる場合は、sf_wifi_api_t::infoGet API コールを使用して取得された RSSI 値は dB 単位の SNR を表します。sf_wifi_api_t::infoGet API コールによって返される noise_level および link_quality フィールドには情報は格納されず、0 に設定されます。

BSD ソケット API：

アプリケーションは、オンチップネットワークスタックを使用するときに、すべての BSD ソケット API、すべてのオンチップネットワークスタックサポート API、およびいくつかの Synergy Wi-Fi フレームワーク API を使用できます。使用可能な Wi-Fi フレームワーク API は、sf_wifi_api_t::provisioningSet API、sf_wifi_api_t::provisioningGet API、

sf_wifi_api_t::scan API、sf_wifi_api_t::macAddressGet, sf_wifi_api_t::macAddressSet API、および sf_wifi_api_t::infoGet API です。

- メモリに制約があるため、Wi-Fi フレームワークは Synergy S1 MCU シリーズをサポートしていません。
- メモリに制約があるため、S3A6 および S128 MCU では、オンチップネットワークスタック（つまり、パス 2 で示した Wi-Fi の使用。Wi-Fi チップセットでネットワークスタックを実行します）のみをサポートします。
- Wi-Fi モジュールドライバーを RSPI に使用するときは、WiFi モジュールの SPI ドライバーでの制限のため、依存関係に自動的に設定される DTC コンポーネントを削除する必要があります。DTC を RSPI で使用するときは 32 ビット転送が必要ですが、GT202 ベンダコード（つまり、WiFi モジュールの SPI ドライバー）は 8 ビット転送のみを使用します。
- GT-202 用に実装された Synergy Wi-Fi フレームワーク API は、再入可能ではありません。これらすべての API は、ドライバー API をコールして要求された操作を行います。GT-202 ドライバーが Wi-Fi フレームワーク API に代わって動作しているときに他の Wi-Fi フレームワークがコールされる場合は、このドライバーは継続中の操作が終了するまで SPP_ERR_IN_USE エラーを返します。
- ペリフェラルおよび IO ピンのドライブ能力が [Medium] に設定されていない場合は、GT-202 用の Synergy Wi-Fi フレームワークの sf_wifi_api_t::provisioningSet API でエラーが発生する可能性があります。
- r_rspi, 上の SPI ドライバーを使用して GT-202 を構成する一方で、スレーブ選択ピン、リセットピンおよび SPI ピン（MISO、MOSI、RSPCK）のドライブ能力を [Medium] に設定します。r_sci_spi, 上の SPI ドライバーを使用して GT-202 を構成する一方で、スレーブ選択ピン、リセットピンおよび SCI ピン（TXD_MOSI および RXD_MISO）のドライブ能力を [Medium] に設定します。r_sci_spi ドライバーを使用する場合は、SCK ピンのドライブ能力を変更しないでください。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.1.34.4 アプリケーションへの WiFi フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに WiFi フレームワークモジュールを組み込む方法について説明します。

注：このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

WiFi フレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用して HAL/Common スレッドに追加します。（WiFi フレームワークモジュールのデフォルト名は g_sf_wifi0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

WiFi フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_wifi0 GT202 Wi-Fi Device Driver on sf_wifi_gt202	Threads	New Stack> Framework> Networking> WiFi> GT202 Wi-Fi Device Driver on sf_wifi_gt202
g_sf_wifi_onchip_stack0 On-Chip Stack on Gt202 Wi-Fi Framework	Threads	New Stack> Framework> Networking> WiFi> On-Chip Stack on Gt202 Wi-Fi Framework
g_sf_socket0 BSD Socket using On-Chip Stack on Gt202 Wi-Fi Framework	Threads	New Stack> Framework> Networking> WiFi> BSD Socket using On-Chip Stack on Gt202 Wi-Fi Framework
g_sf_el_nx0 NetX Port using Wi-Fi Framework on sf_wifi_nsal_nx	Threads	New Stack> Framework> Networking> WiFi> NetX Port using Wi-Fi Framework on sf_wifi_nsal_nx

次の図に示すように、sf_wifi の WiFi フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

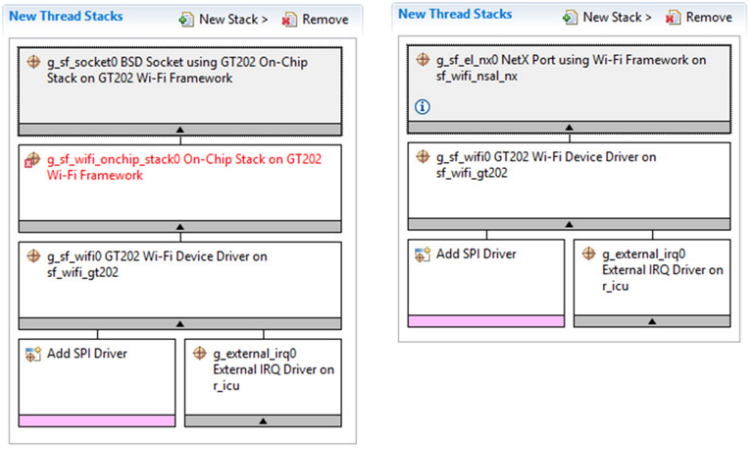


図 136:**WiFi フレームワークモジュールのスタック**

4.1.34.5 WiFi フレームワークモジュールの構成

ユーザーは必要な動作に合わせて WiFi フレームワークモジュールを構成する必要があります。SSP 構成ウィンドウでは、割り込みや動作モードなど必須の設定選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 :ISDE を開き、次に示す構成表の設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

GT202 Wi-Fi フレームワークでの GT202 オンチップスタックを使用した BSD ソケットの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name (Must be a valid C Symbol)	g_sf_socket0	Module name.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

GT202 Wi-Fi フレームワークでのオンチップスタックの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name (Must be a valid C Symbol)	g_sf_wifi_onchip_stack0	Module name.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_wifi_nsal_nx で Wi-Fi フレームワークを使用する NetX ポートの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name (Must be a valid C Symbol)	g_sf_el_nx0	Module name

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_wifi_gt202 上の Wi-Fi デバイスドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
On-Chip Stack Support	Enabled, Disabled Default: Disabled	On-chip stack support selection
Driver Heap Size in Bytes (Minimum 8192 bytes)	8192	Driver heap size selection
Name (Must be a valid C symbol)	g_sf_wifi0	Module name
Hardware Mode	802.11a, 802.11b, 802.11g, 802.11n Default: 802.11n	Hardware mode selection
Transmit (TX) Power (Valid Range 1-17)	10	Transmit power selection

ISDE Property	Value	Description
Ready/Clear to Send (RTS/CTS) Flag	Enabled, Disabled Default: Enabled	Ready/Clear to send selection
Delivery Traffic Indication Message (DTIM) Interval (Valid Range: 1-255)	3	Delivery traffic indication message interval selection
Broadcast SSID (AP mode only)	Enabled, Disabled Default: Enabled	Broadcast SSID selection
Beacon Interval in Microseconds (AP mode only and must be greater than 1023)	1024	Beacon interval in microseconds selection
Station inactivity timeout in seconds (AP mode only and must be greater than 0)	100	Station inactivity timeout selection
Requested High Throughput	Enabled, Disabled Default: Disabled	Requested high throughput selection
Reset Pin (must be a valid C symbol)	IOPORT_PORT_06_PIN_0 0	Reset pin selection
Slave Select Pin (SSL)(Must be a valid C symbol)	IOPORT_PORT_01_PIN_0 3	Slave select pin selection
GT202 Driver Task Thread Priority (Modifying Task Thread Priority may cause Driver to malfunction)	5	GT202 driver task thread priority selection
Callback	NULL	Callback selection
Support NetX Packet Chaining	Enabled, Disabled Default: Enabled	Support NetX packet chaining selection

注:例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

デフォルト以外の設定が望ましい場合もあります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注: ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

WiFi フレームワークのローレベルモジュールの構成

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、ローレベルモジュールの [properties] セクションのすべての設定を示します。

r_rspi 上の SPI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_spi0	Module name.
Channel	0	SCI or SPI Channel number to which the device has been connected.
Operating Mode	Master	Configure as a Master or Slave device. Note: Current version of SSP supports only SPI Master mode.
Clock Phase	Data sampling on even edge, data variation on odd edge	Data sampling on odd or even clock edge.
Clock Polarity	High when idle	Clock level when idle.
Mode Fault Error	Disable	Indicates Mode fault error (master/slave conflict) flag.
Bit Order	MSB First	Select transmit order MSB/LSB first.

ISDE Property	Value	Description
Bitrate	500000	Transmission or reception rate. Bits per second.
Callback	NULL	Optional Callback function pointer.
SPI Mode	Clock synchronous operation	Select spi or clock syn mode operation.
Slave Select Polarity(SSL0)	Active Low	Select SSL0 signal polarity.
Slave Select Polarity(SSL1)	Active Low	Select SSL1 signal polarity.
Slave Select Polarity(SSL2)	Active Low	Select SSL2 signal polarity.
Slave Select Polarity(SSL3)	Active Low	Select SSL3 signal polarity.
Select Loopback1	Normal	Select loopback1.
Select Loopback2	Normal	Select loopback2.
Enable MOSI Idle State	Disable	Select MOSI idle fixed value and selection.
MOSI Idle State	MOSI Low	Select mosi idle fixed value and selection.
Enable Parity	Disable	Enable/disable parity.
Parity Mode	Parity Even	Select parity.
Select SSL(Slave Select)	SSL0	Select which slave to use; 0-SSL0; 1-SSL1; 2-SSL2; 3-SSL3.
Select SSL Level After Transfer	SSL Level Do Not Keep	Select SSL level after transfer completion; 0-negate; 1-keep.
Clock Delay Enable	Clock Delay Disable	Clock delay enable selection.
Clock Delay Count	Clock Delay 1 RSPCK	Clock delay count selection.
SSL Negation Delay Enable	Negation Delay Disable	SSL negation delay enable selection.
Negation Delay Count	Negation Delay 1 RSPCK	Negation delay count selection.

ISDE Property	Value	Description
Next Access Delay Enable	Next Access Delay Disable	Next access delay enable selection.
Next Access Delay Count	Next Access Delay 1 RSPCK	Next access delay count selection.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Receive interrupt priority selection.
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit interrupt priority selection.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit end interrupt priority selection.
Error Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Error interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SPI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build

ISDE Property	Value	Description
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte, 2 Bytes, 4 Bytes Default: 2 Bytes	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SPI0 TXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection

ISDE Property	Value	Description
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SPI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte, 2 Bytes, 4 Bytes Default: 2 Bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection

ISDE Property	Value	Description
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SPI0 RXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sci_spi 上の SPI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_spi0	Module name
Channel	0	SCI or SPI Channel number to which the device has been connected.

ISDE Property	Value	Description
Operating Mode	Master	Configure as a Master or Slave device. Note: Current version of SSP supports only SPI Master mode.
Clock Phase	Data sampling on even edge, data variation on odd edge	Data sampling on odd or even clock edge.
Clock Polarity	High when idle	Clock level when idle.
Mode Fault Error	Disable	Indicates Mode fault error (master/slave conflict) flag.
Bit Order	MSB First	Select transmit order MSB/LSB first
Bitrate	100000	Transmission or reception rate. Bits per second.
Bit Rate Modulation Enable	Enable, Disable Default: Enable	Bitrate Modulation Function enable or disable
Callback	NULL	Optional Call back function pointer.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Bitrate Modulation Function enable or disable. Note: This is applicable only for SCI SPI.
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit interrupt priority selection.

ISDE Property	Value	Description
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit end interrupt priority selection
Error Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Error interrupt priority selection

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

Event SCI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection

ISDE Property	Value	Description
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency seleciton
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SCI0 TXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build

ISDE Property	Value	Description
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SCI0 RXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection

ISDE Property	Value	Description
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_icu 上の外部 IRQ ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter checking setting enables or disables the addition of parameter checking code.
Name	g_external_irq0	Module name.
Channel	0	Specifies the hardware IRQ channel used.
Trigger	Falling	Selection for trigger event mode
Digital Filtering	Disabled	Digital filter enable/disable.
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCKL/64	Sets noise filter sampling period.
Interrupt enabled after initialization	TRUE	Determines if the interrupt is enabled immediately after initialization.

ISDE Property	Value	Description
Callback	custom_hw_irq_isr	A user callback function can be registered in external_irq_api_t::open. If this callback function is provided, it is called from the interrupt service routine (ISR) each time the IRQn triggers. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Interrupt priority selection.

注 : 例の値とデフォルトは、Synergy S7G2 を使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注 : モジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

WiFi フレームワークモジュールのクロック構成

Wi-Fi フレームワークモジュールは、選択された特定のローレベルモジュール（SPI など）に必要なクロックを使用します。

WiFi フレームワークモジュールのピン構成

Wi-Fi フレームワークモジュールは、選択されたローレベルモジュール（SPI や IRQ など）に依存する入力ピンと出力ピンを使用します。

4.1.34.6 アプリケーションでの WiFi フレームワークモジュールの使用

以下の説明では、一般的な Wi-Fi のユースケースの一部について大まかな概要を示します。Renesas Web サイトには、詳しい説明のほか、実際に動作するアプリケーションプロジェクト（このプロジェクトは長いため本書には記載されていません）が掲載されています。次のサイトのトップページに表示される検索バーで、関連するアプリケーションノートのドキュメント番号 r11an0226eu を検索してください：http://www.renesas.com. 本書で要約に示されている説明の補足としてアプリケーションノートを使用することを強く推奨します。

Wi-Fi フレームワークは、実装ごとにターゲットアプリケーションによる扱いが異なります。初期化、NetX/NetX Duo を使用したパケット送信、NetX/NetX Duo を使用したパケット受信、およびオンチップネットワークスタックの使用に関する一般的な制御フローは、特に重要です。これらの機能の一般的な実装のフロー例を次の図に示します。

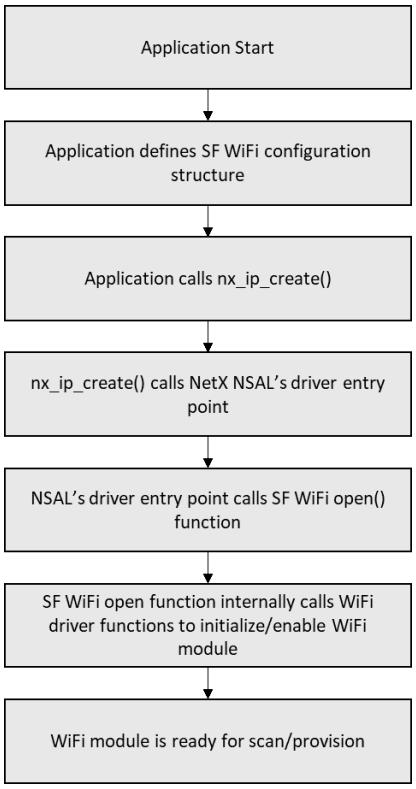


図 137:**Wi-Fi モジュールの初期化を利用するアプリケーションの制御フロー**

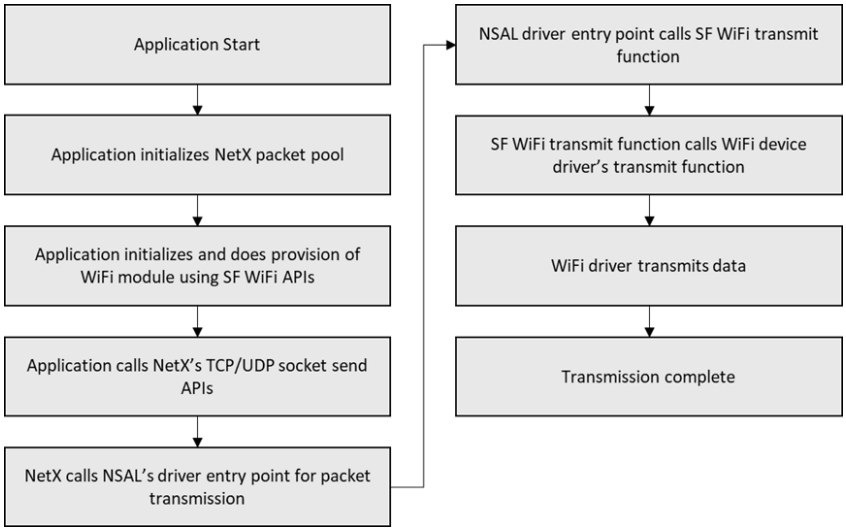


図 138:NetX を使用してパケット送信を実行するアプリケーションの制御フロー

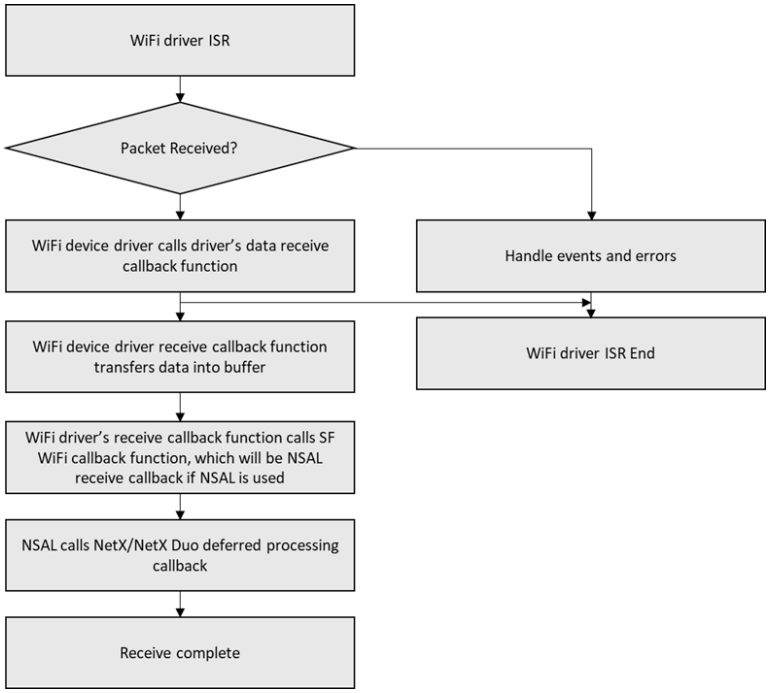


図 139:NetX を使用してパケットを受信するアプリケーションの制御フロー

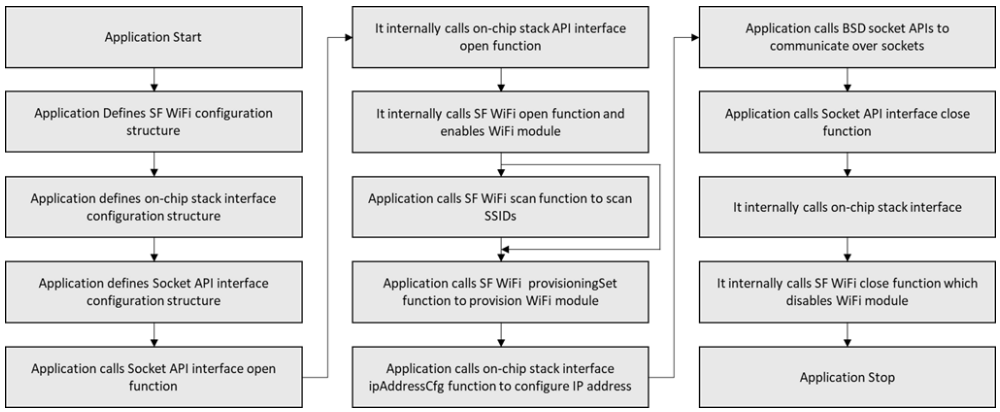


図 140:オンチップネットワークスタックを使用するアプリケーションの制御フロー

4.2 HAL ドライバー

ACMPHS ドライバーー
ACMPLP ドライバーー
ADC ドライバー
AGT ドライバー
CAC ドライバー
CAN ドライバー
CGC ドライバー
CRC ドライバー
CTSU ドライバー
DAC ドライバー
DAC8 ドライバー
ディスプレイドライバー
データ操作回路ドライバー
DMAC ドライバー
DTC ドライバー
ELC ドライバー
外部 IRQ ドライバー
フラッシュ ドライバー
FMI ドライバー
GPT ドライバー

I2C SCI ドライバー
I2C マスタドライバーー
I2C スレーブドライバーー
I2S ドライバー
入力キャプチャ ドライバー
I/O ポート ドライバー
独立ウォッチドッグドライバーー
JPEG デコード ドライバー
JPEG エンコードドライバーー
Key Matrix ドライバー
ローパワー モード ドライバー
低消費電力モード V2 ドライバーー
低電圧検出ドライバー
OPAMP ドライバーー
PDC ドライバー
QSPI ドライバー
RTC ドライバー
SCE 暗号化ドライバーー
SDADC ドライバーー
SD/MMC ドライバーおよび SDIO ドライバー
セグメント LCD ドライバー
SCI SPI ドライバー
SPI ドライバー
UART ドライバー
ウォッチドッグ ドライバー

4.2.1 ACMPHS ドライバー

ACMPHS HAL モジュールは、信号処理アプリケーション用のコンパレータ API を実装し、Synergy マイクロコントローラハードウェア上で使用可能な ACMPHS ペリフェラルをサポートしています。コールバックを使用して、遷移イベントでユーザーアプリケーションに信号を送信できます。

4.2.1.1 ACMPHS HAL モジュールの機能

- 立ち上がりエッジ、立ち下がりエッジ、または両方でのコールバック
- 構成可能なデバウンスフィルタ
- VCOUT ピンでコンパレータ出力を含めるオプション

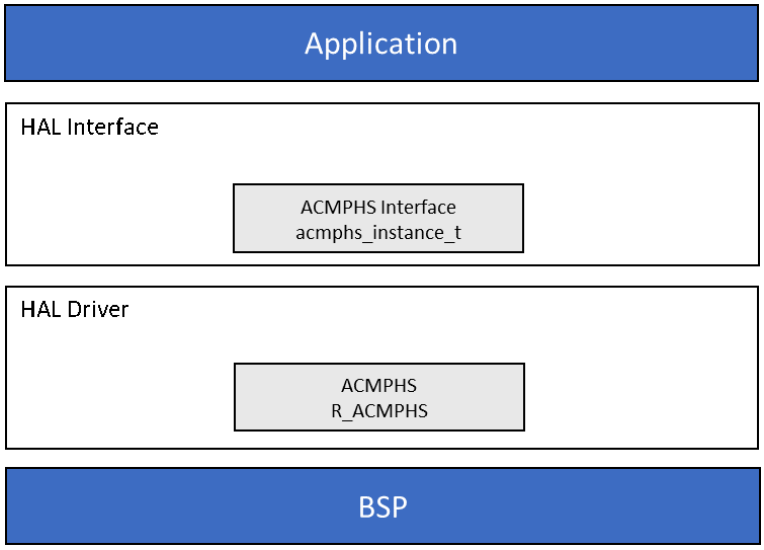


図 141:**ACMPHS HAL モジュールのブロック図**

4.2.1.2 ACMPHS HAL モジュールの API の概要

ACMPHS HAL モジュールは、モジュールのオープン、有効化、ステータスの取得、クローズの API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

ACMPHS HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_comparator0.p_api->open(g_comparator0.p_ctrl, g_comparator0.p_cfg); Configures the comparator and starts operation. Callbacks and pin output are not active until outputEnable() is called. outputEnable() should be called after the output has stabilized.

Function Name	Example API Call and Description
outputEnable	g_comparator0.p_api->outputEnable(g_comparator0.p_ctrl); Enables the comparator output, which can be polled using statusGet(). Also enables pin output and interrupts as configured during open().
infoGet	g_comparator0.p_api->infoGet(g_comparator0.p_ctrl, p_info); Provides the minimum stabilization wait time in microseconds.
statusGet	g_comparator0.p_api->statusGet(g_comparator0.p_ctrl, p_status); Provides the operating status of the comparator.
close	g_comparator0.p_api->close(g_comparator0.p_ctrl); Close the module.
versionGet	g_comparator0.p_api->read(&version); Retrieves the version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.

Name	Description
SSP_ERR_NOT_OPEN	Unit is not open
SSP_ERR_ASSERTION	An input pointer is NULL.
SSP_ERR_IN_USE	Peripheral is in use or hardware lock is taken.
SSP_ERR_TIMEOUT	The debounce filter is off and 2 consecutive matching values were not read within 1024 attempts.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.1.3 ACMPHS HAL モジュールの動作の概要

ACMPHS HAL モジュールは、Synergy マイクロコントローラ上の高速アナログコンパレータペリフェラルを制御します。RTOS 要素を使用しないで ACMPHS ハードウェアを直接制御し、開発が簡単になる便利な API を提供します。

ACMPHS HAL モジュールの動作に関する重要な注意事項と制限事項

VCOUT ピンでのコンパレータ出力

VCOUTピンの信号は、出力ピンが有効なすべてのコンパレータ(ACMPHSおよびACMPLP)の出力の論理和(OR)です。

割り込みとコールバック

コンパレータイベントが発生すると、r_acmphs HAL モジュール上のコンパレータドライバーはコールバック引数（comparator_callback_args_t）を指定してコールバック（p_callback）をコールします。

- このモジュールは、一部の Synergy MCU でのみ使用できます。現在の SSP リリースのリリースノートを参照して、このモジュールでサポートされている MCU を確認してください。使用可能なペリフェラルが MCU ハードウェアマニュアルに示されています。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.1.4 アプリケーションへの ACMPHS HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに ACMPHS HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

コンパレータドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(コンパレータドライバーのデフォルトの名前は g_acmphs0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

ACMPHS HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_comparator0 Comparator Driver on r_acmphs	Threads	New Stack> Driver> Analog> Comparator Driver on r_acmphs

次の図に示すように、r_acmphs のコンパレータドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

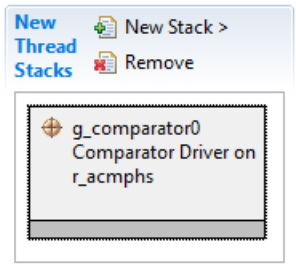


図 142:**ACMPHS HAL モジュールのスタック**

4.2.1.5 ACMPHS HAL モジュールの構成

ユーザーは必要な動作に合わせて ACMPHS HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_acmphs での ACMPHS HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Controls whether to include code for API parameter checking.
Name	g_comparator	Module name.
Channel	0	Select the hardware channel.
Trigger Edge	Rising, Falling, Both Edge Default: Both Edge	The trigger specifies when a comparator callback event should occur. Unused if the interrupt priority is disabled or the callback is NULL.
Debounce Filter	No Filter, 8, 16, 32 Default: No Filter	Select the PCLK divisor for the hardware digital debounce filter. Larger divisors provide a longer debounce and take longer for the output to update.
Invert	Not Inverted, Inverted Default: Not Inverted	Turns this on to invest comparator output. This affects the output read from StatusGet(), the pin output level, and the edge trigger.
Pin Output	Disabled, Enabled Default: Disabled	Turn this on to include the output from this comparator on VCOUT. The comparator output on VCOUT is ORed with output from all other ACMPHS and ACMPLP comparators.
Callback	NULL	Define this function in the application. It is called when the Trigger event occurs.

ISDE Property	Value	Description
Comparator Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for the comparator interrupt.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ACMPHS HAL モジュールのクロック構成

ACMPHS HAL モジュールは PCLKB をそのクロックソースとして使用します。

実行時にクロック周波数を変更するには、CGC インタフェースを使用します。

ACMPHS HAL モジュールのピン構成

ACMPHS HAL モジュールを使用するには、アナログ入力を受信するチャネルのポートピンを、ISDE のピンコンフィギュレータで入力ピンとして設定する必要があります。次の表では、ISDE [Configuration] ウィンドウでのピンの選択方法を示します。

r_acmphs の ACMPHS HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
ACMPLP	Pins	Select Peripherals > Analog:ACMP

4.2.1.6 アプリケーションでの ACMPHS HAL モジュールの使用

アプリケーションで ACMPHS HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、ACMPLP を初期化します。
2) 出力を有効にする前に、ハードウェアマニュアルを参照し、直接 COMPSELn レジスタを設定して内部接続を構成します。内部基準電圧が使用される場合は、COMPMDR.CiVRF を設定します。
3) モジュールと内部接続を構成した後で、最小安定性待機時間を待機してから、出力を有効にします。最小安定性待機時間は、infoGet API を使用して照会できます。
4) outputEnable API を使用してコンパレータ出力を有効にします。これによって open API コール時に、構成したとおりにピン出力、割り込み、statusGet API が有効になります。
5) [オプション] statusGet API を使用して、コンパレータステータスをポーリングします。
6) [オプション] close API を使用して、コンパレータを無効化し、ペリフェラルの電源をオフにします。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

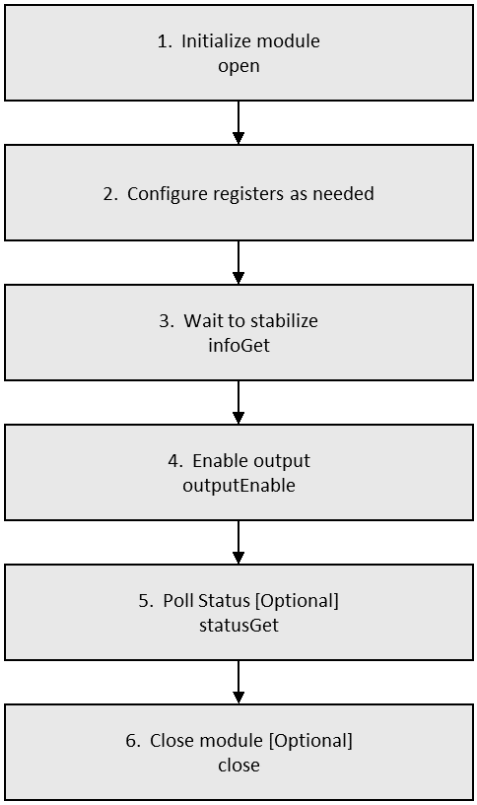


図 143:通常の ACMPHS HAL モジュールアプリケーションのフロー図

4.2.2 ACMPLP ドライバー

ACMPLP HAL モジュールは、信号処理アプリケーション用のコンパレータ API を実装し、Synergy マイクロコントローラハードウェア上で使用可能な ACMPLP ペリフェラルをサポートしています。コールバックを使用して、遷移イベントでユーザーアプリケーションに信号を送信できます。

4.2.2.1 ACMPLP HAL モジュールの機能

- ノーマルモードまたはウィンドウモード
- 立ち上がりエッジ、立ち下がりエッジ、または両方でのコールバック
- 構成可能なデバウンスフィルタ
- VCOUT ピンでコンパレータ出力を含めるオプション

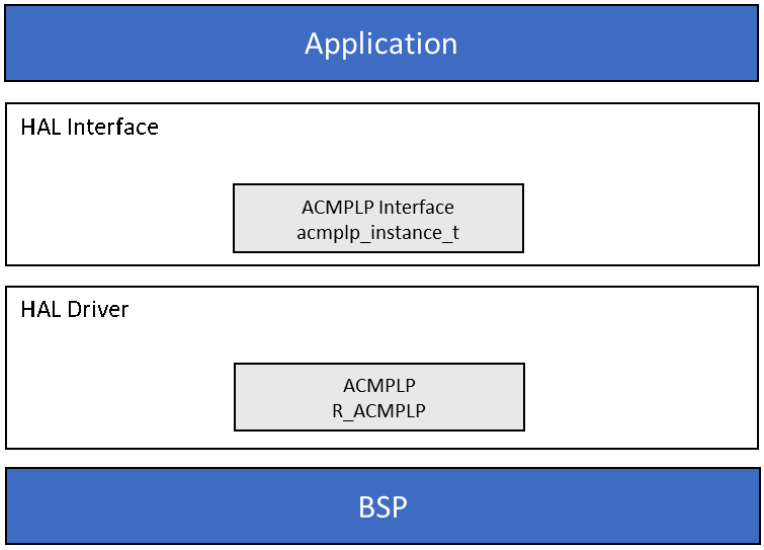


図 144:**ACMPLP HAL モジュールのブロック図**

4.2.2.2 ACMPLP HAL モジュールの API の概要

ACMPHS HAL モジュールは、モジュールのオープン、有効化、ステータスの取得、クローズの API 関数を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

ACMPLP HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_comparator0.p_api->open(g_comparator0.p_ctrl, g_comparator0.p_cfg); Configures the comparator and starts operation. Callbacks and pin output are not active until is called. should be called after the output has stabilized.
outputEnable	g_comparator0.p_api->outputEnable(g_comparator0.p_ctrl); Enables the comparator output, which can be polled using statusGet(). Also enables pin output and interrupts as configured during open().

Function Name	Example API Call and Description
infoGet	g_comparator0.p_api->infoGet(g_comparator0.p_ctrl, p_info); Provides the minimum stabilization wait time in microseconds.
statusGet	g_comparator0.p_api->statusGet(g_comparator0.p_ctrl, p_status); Provides the operating status of the comparator.
close	g_comparator0.p_api->close(g_comparator0.p_ctrl); Close the module.
versionGet	g_comparator0.p_api->read(&version); Retrieves the version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_NOT_OPEN	Unit is not open
SSP_ERR_ASSERTION	An input pointer is NULL.
SSP_ERR_IN_USE	Peripheral is in use or hardware lock is taken.

Name	Description
SSP_ERR_TIMEOUT	The debounce filter is off and 2 consecutive matching values were not read within 1024 attempts.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.2.3 ACMPLP HAL モジュールの動作の概要

コンパレータドライバー（r_acmplp）HAL モジュールは、Synergy マイクロコントローラ上のローパワーアナログコンパレータ（ACMPLP）ペリフェラルを制御します。RTOS 要素を使用しないで ACMPLP ハードウェアを直接制御し、開発が簡単になる便利な API を提供します。

ACMPLP HAL モジュールの動作に関する重要な注意事項と制限事項

VCOUT ピンでのコンパレータ出力

VCOUTピンの信号は、出力ピンが有効なすべてのコンパレータ（ACMPHSおよびACMPLP）の出力の論理和（OR）です。

割り込みとコールバック

コンパレータイベントが発生すると、R_ACMPLP HAL モジュールはコールバック引数（comparator_callback_args_t）を指定してコールバック（p_callback）をコールします。

- このモジュールは、一部の Synergy MCU でのみ使用できます。現在の SSP リリースのリリースノートを参照して、このモジュールでサポートされている MCU を確認してください。各 MCU で使用可能なペリフェラルは、MCU ハードウェアマニュアルに示されています。
- このモジュールの動作に関するその他の制限事項については、最新のSSPリリースノートを参照してください。

4.2.2.4 アプリケーションへの ACMPLP HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに ACMPLP HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

コンパレータドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（コンパレータドライバーのデフォルトの名前は g_acmplp0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

ACMPLP HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_comparator0 Comparator Driver on r_acmplp	Threads	New Stack> Driver> Analog> Comparator Driver on r_acmplp

次の図に示すように、r_acmplp のコンパレータドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

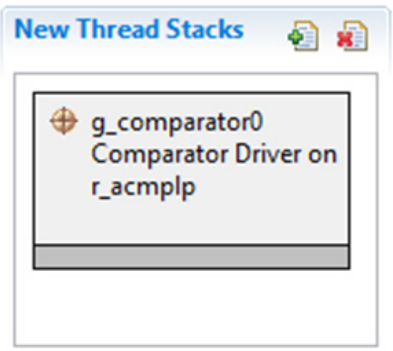


図 145:ACMPLP HAL モジュールのスタック

4.2.2.5 ACMPLP HAL モジュールの構成

ユーザーは必要な動作に合わせて ACMPLP HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_acmplp での ACMPLP HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Controls whether to include code for API parameter checking.
Name	g_comparator	Module name.
Channel	0	Select the hardware channel.
Mode	Mode Normal, Mode Window Default: Mode Normal	In normal mode, comparator output is high if VCMP > VREF. In window mode, comparator output is high if VCMP is outside the range of VREF0 to VREF1.
Trigger	Trigger Rising, Trigger Falling, Trigger Both Edge Default: Trigger Both Edge	The trigger specifies when a comparator callback event should occur. Unused if the interrupt priority is disabled or the callback is NULL.
Filter	Filter Off, Filter 1, Filter 8, Filter 16, Filter 32 Default: Filter Off	Select the PCLK divisor for the hardware digital debounce filter. Larger divisors provide a longer debounce and take longer for the output to update.
Invert	Off, On Default: Off	Turns this on to invest comparator output. This affects the output read from StatusGet(), the pin output level, and the edge trigger.
Pin Output	Off, On Default: Off	Turn this on to include the output from this comparator on VCOUT. The comparator output on VCOUT is OR'd with output from all other ACMPHS and ACMPLP comparators.
Callback	NULL	Define this function in the application. It is called when the Trigger event occurs.

ISDE Property	Value	Description
Comparator Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for the comparator interrupt.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ACMPLP HAL モジュールのクロック構成

ACMPHS HAL モジュールは PCLKB をそのクロックソースとして使用します。

実行時にクロック周波数を変更するには、CGC インタフェースを使用します。

ACMPLP HAL モジュールのピン構成

ACMPLP HAL モジュールを使用するには、アナログ入力を受信するチャネルのポートピンを、ISDE のピンコンフィギュレータで入力ピンとして設定する必要があります。次の表では、ISDE [Configuration] ウィンドウでのピンの選択方法を示します。

r_acmplp の ACMPLP HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
ACMPLP	Pins	Select Peripherals > Analog:ACMP

4.2.2.6 アプリケーションでの ACMPLP HAL モジュールの使用

アプリケーションで ACMPLP HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、ACMPLP を初期化します。
2) 出力を有効にする前に、ハードウェアマニュアルを参照し、直接 COMPSELn レジスタを設定して内部接続を構成します。内部基準電圧が使用される場合は、COMPMDR.CiVRF を設定します。
3) モジュールと内部接続を構成した後で、最小安定性待機時間を待機してから、出力を有効にします。最小安定性待機時間は、infoGet API を使用して照会できます。
4) outputEnable API を使用してコンパレータ出力を有効にします。これによって open API コール時に、構成したとおりにピン出力、割り込み、statusGet API が有効になります。
5) [オプション] statusGet API を使用して、コンパレータステータスをポーリングします。
6) [オプション] close API を使用して、コンパレータを無効化し、ペリフェラルの電源をオフにします。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

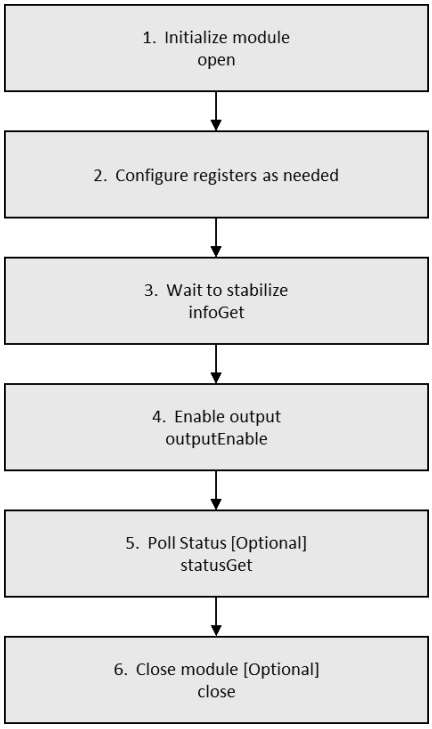


図 146:通常の ACMPLP HAL モジュールアプリケーションのフロー図

4.2.3 ADC ドライバー

ADC HAL モジュールでは、アナログ / デジタル変換アプリケーション用 API を実装します。このモジュールは、Synergy MCU 上で利用可能な関連周辺機能用に ADC12、ADC14、および ADC16 をサポートします。ユーザー定義のコールバックを使用して、新しいサンプルが完了するたびにデータを処理することができます。

4.2.3.1 ADC HAL モジュールの特長

- 12 ビット A/D コンバータ
- 14 ビット A/D コンバータ
- 16 ビット A/D コンバータ
- 複数の動作モード
  - シングルスキャン
  - グループスキャン
  - 連続スキャン
- 複数チャネル
  - ターゲット MCU に応じて 12 ～ 28 のアナログチャネル

- 温度センサーチャネル
- 電圧センサーチャネル

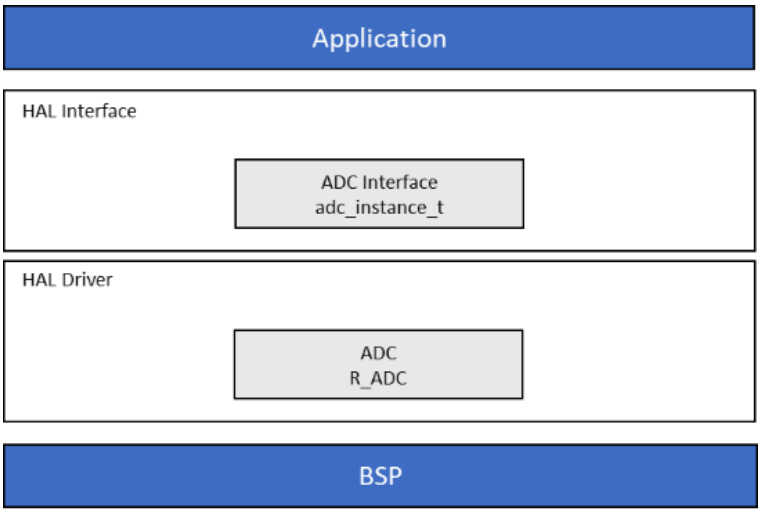


図 147:ADC HAL モジュールのブロック図

4.2.3.2 ADC HAL モジュールの API の概要

ADC HAL モジュールでは、ADC ユニットを開き、スキャンの構成、開始、停止を行い、ADC スキャンの変換結果を読み取り、ADC ユニットを閉じるための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

ADC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_adc.p_api->open(g_adc.p_ctrl, g_adc.p_cfg); Initialize ADC Unit; apply power, set the operational mode, trigger sources, interrupt priority, and configurations common to all channels and sensors.

Function Name	Example API Call and Description
scanCfg	g_adc.p_api->scanCfg(g_adc.p_ctrl, g_adc.p_channel_cfg); Configure the scan including the channels, groups and scan triggers to be used for the unit that was initialized in the open call.
scanStart	g_adc.p_api->scanStart(g_adc.p_ctrl); Start the scan (in case of a software trigger), or enable the hardware trigger.
scanStop	g_adc.p_api->scanStop(g_adc.p_ctrl); Stop the ADC scan (in case of a software trigger), or disable the hardware trigger.
scanStatusGet	g_adc.p_api->scanStatusGet(g_adc.p_ctrl); Check scan status.
read	g_adc.p_api->read(g_adc.p_ctrl, ADC_REG_CHANNEL_13, &adc_data); Read ADC conversion result(s).
sampleStateCountSet	g_adc.p_api->sampleStateCountSet(g_adc.p_ctrl,&adc_sample); Set the sample state count for the specified channel.
close	g_adc.p_api->close(g_adc.p_ctrl); Close the specified ADC unit by ending any scan in progress, disabling interrupts, and removing power to the specified A/D unit.

Function Name	Example API Call and Description
infoGet	g_adc.p_api->infoGet(g_adc.p_ctrl, &adc_info); Return the ADC data register address of the first (lowest number) channel and the total number of bytes to be read for the DTC/DMAC to read the conversion results of all configured channels.
versionGet	g_adc.p_api->versionGet(&version); Retrieve the API version with the version pointer.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_NOT_OPEN	Unit is not open
SSP_ERR_ASSERTION	The parameter p_ctrl or p_sample is NULL.
SSP_ERR_IN_USE	Peripheral is still running in another mode; perform R_ADC_Close first.
SSP_ERR_INVALID_POINTER	The parameter p_data is NULL.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.3.3 ADC HAL モジュールの動作の概要

r_adc HAL モジュール上の ADC ドライバーは、ユーザーの構成に従って、Synergy マイクロコントローラ上の ADC ペリフェラルを制御します。RTOS 要素を使用しないで ADC ハードウェアを直接制御します。また、開発が簡単になる便利な API を提供します。

このドライバーは、シングルスキャン、連続スキャン、グループスキャンの 3 つの動作モードをサポートしています。

シングルスキャンモード

シングルスキャンモードでは、1 つ以上の指定したチャネルがトリガごとに 1 回スキャンされます。チャネルプロパティ構成設定でチャネルビットマスクが使用されて、スキャンされたチャネルが示されます。シングルスキャンモードは、チャネル番号の昇順にチャネルを順番に選択して、アナログ入力を変換します。すべての選択されたチャネルが変換処理を完了した後に、コールバックイベントが生成されます。

連続スキャンモード

連続スキャンモードは、チャネル番号の昇順にチャネルを連続で選択して、アナログ入力を変換します。スキャンを開始するには、トリガが 1 つ必要です。このモードではコールバックは使用されず、割り込みを無効にする必要があります。scanStatusGet API 関数を使用して、データが使用できる時点を特定します。

グループスキャンモード

グループスキャンモードでは、アプリケーションは 2 つのグループ（A と B）のどちらかにチャネルを割り当てることができます。選択したチャネルのアナログ入力は、チャネル番号の昇順でグループごとに変換されます。対象のグループに対して指定した開始トリガを受信すると、変換が開始されます。関連付けられたグループで選択されたすべてのチャネルが変換処理を完了すると、コールバック割り込みが生成されます。割り込みイベントによって、変換を完了したグループが示されます。

グループ モードではハードウェア トリガのみを使用できるのに対し、通常モードでは、ソフトウェア トリガまたは外部トリガを使用できます。プライオリティ構成パラメータを使用すると、以下のことを指定できます。

- あるグループのトリガが、他のグループの進行中のスキャンを中断できるかどうか。
- 中断されたスキャンを、再開するか、最初から開始し直すか、単に現在のスキャンを中止して次のトリガを待つかどうか。

割り込みおよびコールバックの概要

（サポートされる MCU 上で）スキャンまたは較正が完了し、コールバックがアプリケーションコードで定義されている場合（さらに割り込みが有効な場合）は、モジュールは定義されたコールバックを呼び出し、ADC ユニット、イベント、変換されたデータのアドレス、およびチャネルを示す引数を指定します。

モジュールは 2 つの割り込みをサポートしています。

- 通常 / グループ A 割り込み（スキャン終了割り込み）は、シングルスキャンモードでスキャンが完了したとき、グループモードでグループ A のスキャンが完了したとき、またはサポートされる MCU の較正が終了したときに発生します。
- グループ B 割り込み（スキャン終了グループ B 割り込み）は、グループモードでグループ B のスキャンが完了すると発生します。

割り込みの機能はモードによって異なります。

- シングルスキャンモードでは、通常割り込み（スキャン終了割り込み）はスキャンが完了するときにトリガされます。
- 連続スキャンモードでは、選択されたチャネルをハードウェアが常にスキャンします。このモードでは、割り込みが有効になっているとドライバーがエラーを返します。したがって、このモードでは割り込みを無効にする必要があります。
- グループモードでは、ADC ユニットによって 2 つの割り込みが行われます（有効になっている場合）。これらは、通常割り込み（このモードでは通常割り込みとベクトルが同じでも、グループ A 割り込みと呼ばれます）とグループ B 割り込みです。グループ A 割り込み（スキャン終了割り込み）は、グループ A スキャンが完了するときにトリガされます。グループ B 割り込み（スキャン終了グループ B 割り込み）は、グループ B スキャンが完了するときにトリガされます。

注 : 選択したユニットのプロパティウィンドウで、[Scan End Interrupt Priority] および [Scan End Group B Interrupt Priority] 構成設定を、デフォルトの [Disabled] 設定から目的のプライオリティレベルに変更して、関連付けられた割り込みを有効にする必要があります。

割り込みが有効でない場合

割り込みが有効になっていない場合は、scanStatusGet API を使用して ADC をポーリングして、スキャンが完了した時点を特定します。read API 関数を使用して、変換後の ADC の結果にアクセスします。

較正をサポートする MCU では、割り込みが有効でない場合、アプリケーションプログラムは 24 ミリ秒待機してから、infoGet API を使用して較正関数のステータスを確認する必要があります。較正が完了すると、別の API を使用できます。

ADC HAL モジュールの動作に関する重要な注意事項と制限事項

サンプル状態カウントの設定

アプリケーションプログラムは、sampleStateCountSet API 関数をコールすることにより、サンプル状態カウントの設定を変更できます。アプリケーションプログラムに必要なのは、サンプル状態カウントの設定をデフォルト値から変更して、サンプリング時間を長くすることだけです。入力信号のインピーダンスが高すぎてデフォルト設定では十分なサンプリング時間が確保できない場合か、ADCLK が低すぎる場合に、この変更が必要になることがあります。指定したチャネルのサンプル状態カウントを変更するには、sampleStateCountSet API 関数をコールするときに、チャネル番号と状態の数を設定します。有効なサンプル状態カウントは 7 ～ 255 です。

注 : ハードウェアがサンプル状態の最小数 5 をサポートする場合でも、一部の Synergy MCU では 7 つの状態を必要とするため、最小数は 7 に設定されます。サポートする最小 ADC 変換クロックレート（1MHz）では、この特別な状態によって、最悪のケースでは変換時間が 2 ミリ秒長くなります。60MHz では、この特別な状態により、変換時間が 33.4 ナノ秒長くなります。

サンプル状態カウントを複数のチャネルに対して変更する必要がある場合、アプリケーションプログラムでは、チャネルごとに引数を適切に変更して sampleStateCountSet API 関数を繰り返しコールする必要があります。

ADC でのデータ転送のトリガ

ADC スキャンが完了したときにデータの転送をトリガするには、activation_source を ELC_EVENT_ADCn_SCAN_END または ELC_EVENT_ADCn_SCAN_END_B（n は ADC チャネル番号）に設定してデータ転送を構成します。infoGet API 関数をコールして、転送 API で使用する ADC ユニット固有の情報を取得できます。詳細については、ELC モジュールの概要を参照してください。

ADC を使用した ELC イベントのトリガ

ADC ユニットは、ELC を使用して他のペリフェラルの起動をトリガできます。詳細については、ELC モジュールの概要を参照してください。

ADC での温度センサーの使用

ADC HAL モジュールは、オンチップ温度センサーからのデータの読み取りをサポートしています。センサーから返された値は、T = (Vs – V1)/ スロープ + T1 という式を使用して、アプリケーションプログラムでセ氏またはカ氏に変換できます。変数の意味は次のとおりです。

- T: 測定温度 (°C)
- Vs: 温度センサーによる温度測定時の電圧出力（ボルト）
- T1: 1 地点で実験的に測定された温度 (°C)
- V1: 温度センサーによる T1 測定時の電圧出力（ボルト）
- T2: 別の地点で実験的に測定された温度 (°C)
- V2: 温度センサーによる T2 測定時の電圧出力（ボルト）
- スロープ : 温度センサーの温度勾配 (V/°C)、スロープ = (V2 – V1)/(T2 – T1)

注 : スロープの値は、各デバイスのハードウェアマニュアルの電気特性の章に記載されている TSN 特性の表、温度スロープエントリから取得できます。スロープは、S7 および S5 デバイスでは正、S3 および S1 デバイスでは負です。

モジュールを構成するときに、他の使用可能なチャネルも選択する場合は、温度センサーと電圧センサーを選択しないでください。温度センサーと電圧センサーを一緒に使用することはできますが、いずれかの通常 ADC チャネルを使用する場合はどちらも使用できません。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.3.4 アプリケーションへの ADC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに ADC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

ADC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(ADC ドライバーのデフォルト名は g_adc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

ADC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_adc0 ADC Driver on r_adc	Threads	New Stack> Driver> Analog> ADC Driver on r_adc

次の図に示すように、r_adc の ADC ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

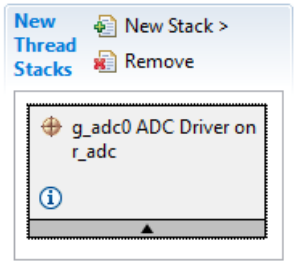


図 148:ADC HAL モジュールのスタック

4.2.3.5 ADC HAL モジュールの構成

ユーザーは必要な動作に合わせて ADC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になりま

す。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_adc での ADC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: Enabled	If selected, code for parameter checking is included in the build.
Name	g_adc0	Module name.
Unit	0, 1 (S7G2, S5D9 and S5D5) Default: 0	Specify the ADC Unit to be used. The s7g2 has two units; 0 and 1.
Resolution (resolution varies by MCU)	14-Bit, 12-Bit, 10-Bit, 8-Bit Default: 8-Bit	Specify the conversion resolution for this unit.
Alignment	Right, Left Default: Right	Specify the conversion result alignment.
Clear after read	Off, On Default: On	Specify if the result register must be automatically cleared after the conversion result is read. Note: If this is enabled, then watching the result register using a debugger always results in a 0.

ISDE Property	Value	Description
Mode	Single Scan, Continuous Scan, Group Scan Default: Single Scan	Specify the mode that this ADC unit is used in.
Internal Calibration During Open()	Disabled, Enabled Default: Enabled	Internal calibration during open selection.
Channels 0-13	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channel 14 (S3 Series Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channel 15 (S3A7/S3A3 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.

ISDE Property	Value	Description
Channels 16-20	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channel 21 (Unit 0 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channels 22-24	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channel 25 (S3 series only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.

ISDE Property	Value	Description
Channel 26 (S3A7/S3A3 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Channel 27 (S3A7/S3A3 Only)	Unused, Use in Normal/Group A, Use in Group B Default: Unused	In Normal mode of operation, this bitmask field is used to specify the channels that are enabled in that ADC unit. For example, if it is set to 0x101, then channels 0 and 2 are enabled. In group mode, this field is used to specify which channels belong to group A.
Temperature Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	Temperature sensor use selection for Channel Scan Mask
Voltage Sensor	Unused, Use in Normal/Group A, Use in Group B Default: Unused	Voltage sensor use selection for Channel Scan Mask
Normal/Group A Trigger	None, Asynchronous External Trigger 0, ELC Event, Software Default: Software	Specify the trigger type to be used for this unit. If group mode is used , then this field is used to set the Group A trigger. Note: The only valid option in group mode is the ELC trigger.

ISDE Property	Value	Description
Group B Trigger (Valid Only in Group Scan Mode)	ELC Event (The only valid trigger for either group in Group Scan Mode)	Specify the group B trigger. This option is only valid if group mode is chosen in .
Group Priority (Valid only in Group Scan Mode)	Group A cannot interrupt Group B, Group A can interrupt Group B; Group B scan restarts at next trigger, Group A can interrupt Group B; Group B scan restarts immediately, Group A can interrupt Group B; Group B scan restarts immediately and scans continuously Default: Group A cannot interrupt Group B	Determines whether an ongoing group B scan can be interrupted by a group A trigger, whether it should abort on a group A trigger, or if it should pause to allow group A scan and restart immediately after group A scan is complete. Note: This field is valid only in group mode.
Add/Average Count	Disabled, Add two samples, Add three samples, Add four samples, Add sixteen samples, Average two samples, Average four samples Default: Disabled	Specify if addition or averaging needs to be done for any of the channels in this unit. The actual channels are specified by using a channel mask .
Channels 0-27	Disabled, Enabled Default: Disabled	This field is valid only if is enabled. This field determines what channels results are to be averaged or summed.
Temperature Sensor	Disabled, Enabled Default: Disabled	Temperature sensor use selection for Addition/Averaging Mask
Voltage Sensor	Disabled, Enabled Default: Disabled	Voltage sensor use selection for Addition/Averaging Mask
Sample and Hold Mask	Select channels for which individual sample and hold circuit is to be enabled	Sample and hold mask selection

ISDE Property	Value	Description
Channels 0-2	Disabled, Enabled Default: Disabled	Determines which of channels 0, 1 and 2 are using the updated sample-and-hold states value specified in . This field must only be set if it is desired to modify the default sample and hold count value for channels 0, 1 and 2.
Sample Hold States (Applies only to the 3 channels selected above)	24	Specifies the updated sample-and-hold count for the channel dedicated sample-and-hold circuit. This field is valid only if is not 0. Only channels 0, 1 and 2 have dedicated sample and hold circuits. Note: Use this to modify the default number of states (24) for which the value is sampled. Each state is equal to 1/ADCLK time.
Callback	NULL	A user callback function can be registered in . If this callback function is provided, it is called from the interrupt service routine (ISR) each time the ADC scan completes. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Scan End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Scan End Interrupt Priority selection

ISDE Property	Value	Description
Scan End Group B Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Scan End Group B Interrupt Priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ADC HAL モジュールのクロック構成

ADC HAL モジュールは PCLKC をクロックソースとして使用します（ADCLK）。このクロックを構成するときの唯一の制限事項は、最大 ADC クロックより小さい値に設定することです。また、PCLKC クロックと PCLKB クロックの比率には、ハードウェアマニュアルで指定されている制限があります。

ADC の変換時間は、PCLKC の設定に依存します。

PCLKB と PCLKC の周波数を設定するには、ISDE のクロックコンフィギュレータを使用します。

実行時にクロック周波数を変更するには、CGC インタフェースを使用します。

ADC HAL モジュールのピン構成

ADC HAL モジュールを使用するには、アナログ入力を受信するチャネルのポートピンを、ISDE のピンコンフィギュレータで入力ピンとして設定する必要があります。次の表では、ISDE [Configuration] ウィンドウでのピンの選択方法を示します。

r_adc での ADC HAL モジュール用のピンの選択

Resource	ISDE Tab	Pin selection Sequence
ADC	Pins	Select Peripherals > Analog Pins>ADC0\1>AN_XX

4.2.3.6 アプリケーションでの ADC HAL モジュールの使用

アプリケーションで ADC HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して ADC を初期化します。（構成で較正が有効になっている場合は、較正をサポートする MCU に対する open コールの一部として較正が実行されます。）
2) scanCfg API を使用してチャネルを構成します。較正をサポートする MCU では、構成で較正が有効ではなかった場合は、最初のスキャンを開始する前に較正を実行する必要があります。calibrate API を使用して、（サポートされる MCU に対して）較正を開始します。
   a) 割り込みが無効になっている場合は、最低 24 ミリ秒（32MHz PCLKB の場合）待機し、infoGet API を使用してステータスを確認し、較正が完了してから他の ADC API が使用されるようにします。
   b) 割り込みが有効になっている場合は、較正が完了するとコールバックが呼び出されます。

3) scanStart API で必要なトリガを使用して変換を開始します。
   a) ハードウェアトリガを使用する場合、この呼び出しにより、ADC ユニットをハードウェアトリガでトリガできるようになります。ソフトウェア トリガを使用する場合、この呼び出しにより ADC スキャンが開始されます。
4) 割り込みが無効になっている場合は、scanStatusGet API を使用して、スキャンが完了したかどうかを特定します。
5) 割り込みが有効になっている場合は、スキャンが完了するとコールバックが呼び出されます。
6) read API を使用して変換結果を読み取ります。
7) scanStop API をコールして ADC のスキャンを停止します。
   a) これにより、ADC は外部トリガまたはハードウェアトリガによってトリガされなくなります。また、進行中のソフトウェアトリガスキャンがある場合は強制的に停止されます。
8) 受信したデータをアプリケーションの必要に応じて操作します。
9) close API を使用して、ペリフェラルの電源をオフにします。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

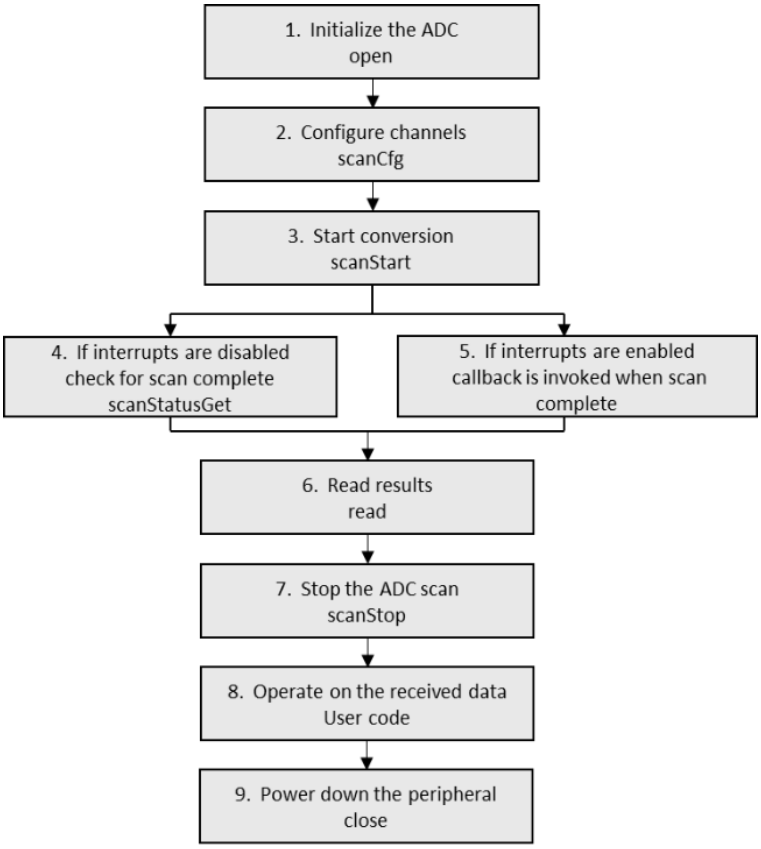


図 149:一般的な ADC HAL モジュールアプリケーションのフロー図

4.2.4 AGT ドライバー

AGT HAL モジュールは、タイミングアプリケーション用のハイレベル API を実装し、Synergy MCU 上の AGT ペリフェラルを使用します。ユーザー定義のコールバックを作成し、タイマイベントに応答できます。

4.2.4.1 AGT HAL モジュールの特長

- 設定した期間に対してタイマを較正し、期間が切れたときに以下のいずれかのイベントを生成します。
  - ユーザーコールバック関数をコールする CPU への割り込み（指定されている場合）
  - ポートピンのトグル
  - DMAC/DTC を使用したデータ転送（転送インタフェースで構成されている場合）
  - 別のペリフェラルの開始（イベントおよびペリフェラル定義で構成されている場合）
- 複数チャネル : 16 ビットのチャネルが 2 個
  - チャネル 1 はチャネル 0 のアンダーフローによってクロック供給することができ、カスケードされた 32 ビットタイマが作成されます。
- コアクロック : PCLKB、LOCO、または Fsub を使用してクロックを供給できます。LOCO または Fsub でクロックが供給された場合、それを使用して MCU をスリープモードから起動できます。

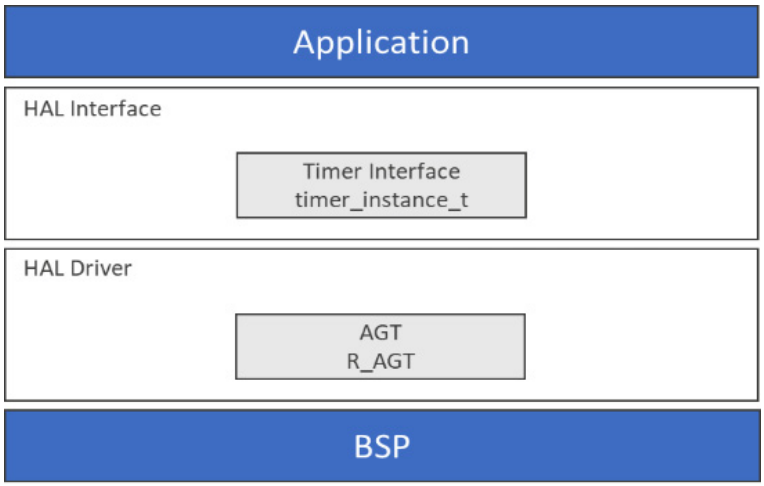


図 150:AGT HAL モジュールのブロック図

4.2.4.2 AGT HAL モジュールの API の概要

AGT HAL モジュールでは、タイマをオープン、クローズ、開始、停止するための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

AGT HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_timer0.p_api->open(g_timer0.p_ctrl, g_timer0.p_cfg) Initial configuration.
stop	g_timer0.p_api->stop(g_timer0.p_ctrl) Stop the counter.
start	g_timer0.p_api->start(g_timer0.p_ctrl) Start the counter.
reset	g_timer0.p_api->reset(g_timer0.p_ctrl) Reset the counter initial value.
counterGet	g_timer0.p_api->counterGet(&value) Get current counter value and store it in the provided pointer, value.
periodSet	g_timer0.p_api->periodSet(g_timer0.p_ctrl, period, unit) Set the time until the timer expires.
dutyCycleSet	g_timer0.p_api->dutyCycleSet(g_timer0.p_ctrl, period, unit, pin) Sets the time until the duty cycle expires.
infoGet	g_timer0.p_api->infoGet(&info) Get the time until the timer expires in clock counts and store it in provided pointer, info.

Function Name	Example API Call and Description
close	g_timer0.p_api->close(g_timer0.p_ctrl) Allows driver to be reconfigured and may reduce power consumption.
versionGet	g_timer0.p_api->versionGet(g_timer0.p_ctrl, &version) Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Operation is successful
SSP_ERR_ASSERTION	Parameter is NULL or configuration setting is not allowed
SSP_ERR_IN_USE	The channel specified is already open
SSP_ERR_IRQ_BSP_DISABLED	A required interrupt is not enabled in the BSP
SSP_ERROR_NOT_OPEN	The channel is not open
SSP_ERR_INVALID_ARG	Invalid argument provided
SSP_ERR_INVALID_HW_CONDITION	Invalid hardware setting detected
SSP_ERR_INVALID_PTR	A pointer parameter was NULL, but needed a non-NULL value

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.4.3 AGT HAL モジュールの動作の概要

AGT HAL モジュールは、ユーザーが指定した時間にタイマを構成します。時間が経過した時点で、CPU の割り込み、ポートピンの切り替え、DMAC または DTC を使用したデータ転送の開始、別の周辺機器の動作開始のトリガなどを行うことができます。

次の図では、指定した時間後にポートピンの切り替えまたは CPU 割り込みの生成を行うためのフローチャートを示します。このフローチャートは、AGT カウンタと GPT カウンタの両方に適用されます。(AGT 動作の場合は GPT の参照を AGT の参照に置き換えます。AGT はダウンカウンタであるため、オーバーフローをアンダーフローに変更します。)

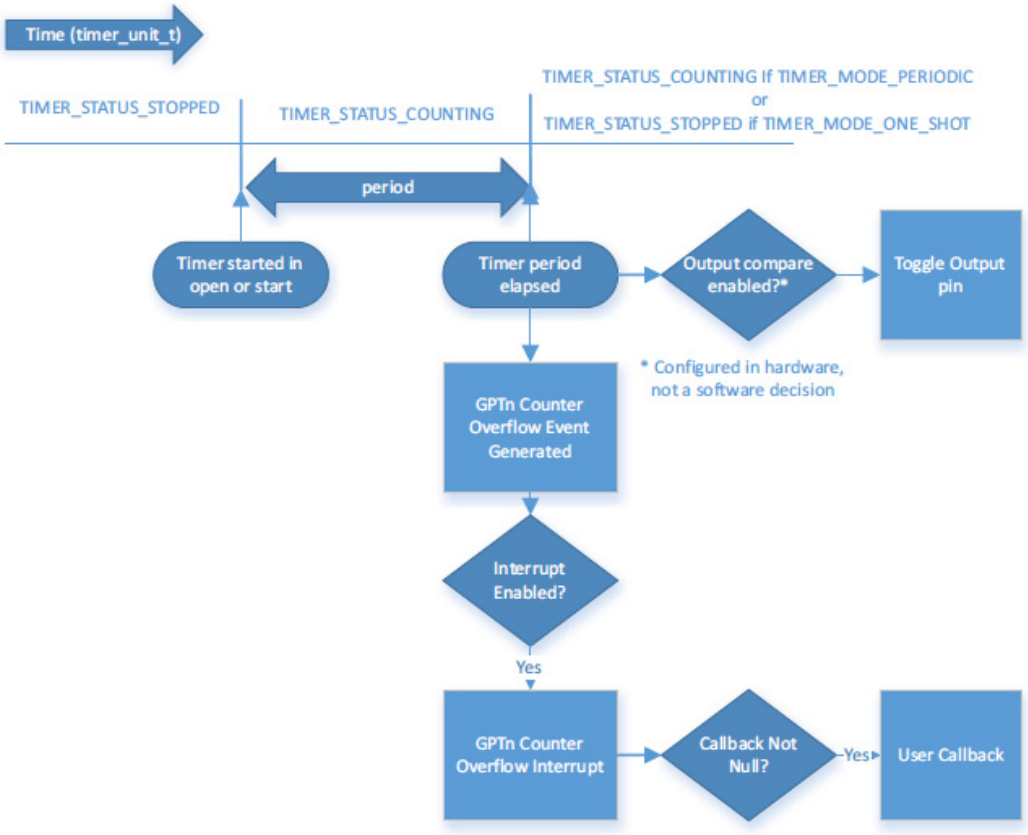


図 151:AGT HAL モジュールのフローチャート

SSP では、GPT と AGT の 2 種類のタイマモジュールがサポートされます。以下のセクションでは、開発者が特定のアプリケーションに対する各モジュールの機能を比較して対照できるように、両方のモジュールについての情報を提供します。GPT の追加情報については、『GPT ユーザーズガイド』を参照してください。

GPT モジュールは、ほとんどの汎用タイマアプリケーションに推奨されますが、基本的なタイマ機能にはどちらのモジュールでも使用できます。一方のタイマモジュールが他方より推奨されるユースケースを以下で説明します。

GPT タイマー モジュールの選択

GPT モジュールは、PCLKA のみでクロック供給可能な、高解像度の 32 ビット カウンタを使用します。Synergy デバイスでは AGT チャネルより GPT チャネルの方が多いので、GPT を使用することでリソースが競合する可能性が低くなります。

AGT タイマー モジュールの選択

AGT モジュールは、PCLKB、LOCO、または Fsub でクロック供給可能な 16 ビットカウンタを使用します。LOCO または Fsub でクロック供給された場合、AGT 割り込みを使用して MCU をスリープ モードから起動できます。2 つのチャネルがあり、チャネル 1 はチャネル 0 のアンダーフローによってクロック供給できるため、実質的に 32 ビットのカスケードされたタイマが作成されます。

AGT HAL モジュールの動作に関する重要な注意事項と制限事項

最大の時間間隔は、タイマの種類と入力クロック周波数に依存します。

- 120MHz で動作する PCLKA を使用した 32 ビット解像度の GPT では、最大期間は約 36650 秒であり、10 時間を少し超えます。
- 32MHz で動作する PCLKA を使用した 16 ビット解像度の GPT では、最大期間は約 2.09 秒です。
- 60MHz で動作する PCLKB を使用した 16 ビット解像度の AGT では、最大期間は約 8.7ms です。
- 32kHz で動作する Fsub を使用した 16 ビット解像度の AGT では、最大周期は約 2.0 秒です。

以下の状況では、選択した使用チャネルに対する AGT カウンタアンダーフロー割り込みを BSP で有効にする必要があります。

- タイマ期間が経過したらソフトウェア割り込みを取得するため。
- ワンショットモードを使用する。

BSP で AGTn AGTI 割り込みが有効になっている場合、対応する ISR がタイマドライバーで定義されます。ISR は、ユーザーコールバック関数が open で登録されていれば、それを呼び出します。

注 : 割り込みは、IRQ を TRANSFER_IRQ_END. に設定した DTC ペリフェラルで使用された場合、スキップされることがあります。

AGT 出力タイマ信号

タイマ出力が構成されている場合 ([AGTO Output Enabled] が true に設定されている)、出力ピンは、出力反転が True に構成されている場合は高レベルで開始し、False に構成されている場合はローレベルで開始します。出力ピンは、タイマ開始後最初の時間間隔が経過して以降、時間間隔が経過するたびに切り替わります。

ワンショット モードでは、タイマーのカウントが開始されるときにも切り替わるように出力が設定されます。これによりパルスが生成されます。タイマーは、カウントが始まるときに停止レベルから切り替わり、カウントが終了するときに停止レベルに戻ります。

タイマー期間の計算

タイマー期間は、タイマーが期限切れになるまでの時間として定義されます。出力比較を使用する場合、出力ピンが時間間隔ごとに 1 回切り替わるため、従来の時間間隔 (立ち上がりエッジから立ち上がりエッジ) はソフトウェアで指定された時間間隔の 2 倍になります。

現在のクロック設定に基づく実行時期間計算は、open および periodSet から使用できます。

指定されたタイマ時間間隔が raw カウントと異なる場合は、現在のタイマクロック周波数を使用して時間間隔が計算されます (「GPT クロックの設定」または「AGT クロックの設定」を参照してください)。現在のタイマクロック周波数を確認するには、systemClockFreqGet を使用します。この周波数は、次の表の適切な式で、clk_freq_hz. として使用されます。

タイマー期間の計算

Timer Units	Description
TIMER_UNIT_PERIOD_NSEC	Counts = (period * clk_freq_hz)/ (1000000000 * channel_0_period)
TIMER_UNIT_PERIOD_USEC	Counts = (period * clk_freq_hz)/ (1000000 * channel_0_period)
TIMER_UNIT_PERIOD_MSEC	Counts = (period * clk_freq_hz)/ (1000 * channel_0_period)
TIMER_UNIT_PERIOD_SEC	Counts = (period * clk_freq_hz) / (channel_0_period)
TIMER_UNIT_FREQUENCY_HZ	Counts = (clk_freq_hz)/ (period * channel_0_period)
TIMER_UNIT_FREQUENCY_KHZ	Counts = (clk_freq_hz)/ (1000 * period * channel_0_period)

注 : 通常モードでは、channel_0_period の値は 1 です。カスケードモードでは、channel_0_period の値はタイマ T0 のカウント値です。

タイマ期間の計算

必要な時間間隔がカウンタのサイズ（32 ビットまたは 16 ビット）より長い場合、ドライバーは結果がカウンタのサイズに収まる最も小さい除数を選択します。カウンタ値が、最大の除数（1024）を持つカウンタサイズよりも大きい場合は、エラーコード (SSP_ERR_INVALID_ARGUMENT) が返されます。

GPT を使用した DMAC/DTC のトリガー

タイマの時間間隔が経過したときに、DMAC または DTC の周辺機器を使用してデータの転送をトリガするには、activation_source を ELC_EVENT_GPTn_COUNTER_OVERFLOW（n は GPT チャネル番号）に設定して DMAC/DTC 転送を設定します。詳細については、DMAC または DTC のガイドを参照してください。

注 :DTC でワンショットモードのタイマを使用した場合、IRQ が TRANSFER_IRQ_END. に設定されていると、割り込みによりタイマが停止する前に、転送全体が完了します。タイマ期間が経過した後で 1 回だけ転送を生成するには、IRQ を TRANSFER_IRQ_EACH に設定するか、DMAC を使用して転送します。

GPT による ELC イベントのトリガ

GPT タイマは、他の周辺機器の起動をトリガできます。ELC のガイドには、すべての使用可能な周辺機器のリストがあります。

AGT を使用した DMAC/DTC のトリガー

タイマの時間間隔が経過したときに、DMAC または DTC の周辺機器を使用してデータの転送をトリガするには、activation_source を ELC_EVENT_AGTn_AGTI（n は AGT チャネル番号）に設定して DMAC/DTC 転送を設定します。詳細については、「転送インタフェース」を参照してください。

*注 :DTC でワンショットモードのタイマを使用した場合、IRQ が TRANSFER_IRQ_END. に設定されていると、割り込みによりタイマが停止する前に、転送全体が完了します。タイマ期間が経過した後で 1 回だけ転送を生成するには、IRQ を TRANSFER_IRQ_EACH, に設定するか、DMAC を使用して転送します。*AGT による ELC イベントのトリガ

AGT タイマは、他の周辺機器の起動をトリガできます。ELC のガイドには、elc_peripheral_t. に列記されているすべての使用可能な周辺機器のリストがあります。（詳細については、イベントと周辺機器の定義を参照してください）。

32 ビットタイマを作成するカスケードされた AGT タイマ

AGT チャネル 1 は AGT チャネル 0 のアンダーフローによってクロック供給することができ、カスケードされた 32 ビットタイマが作成されます。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.4.4 アプリケーションへの AGT HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに AGT HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

タイマドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（タイマドライバーのデフォルト名は g_agt0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

AGT HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
r_agt0 Timer Driver on r_agt	Threads	New Stack> Driver> Timers> Timer Driver on r_agt

次の図に示すように、r_agt のタイマドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

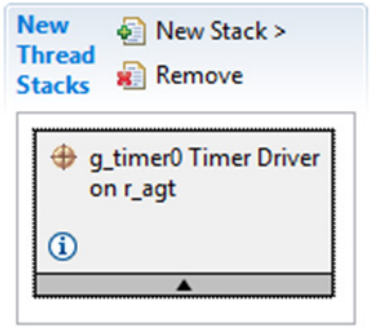


図 152:AGT HAL モジュールのスタック

4.2.4.5 AGT HAL モジュールの構成

ユーザーは必要な動作に合わせて AGT HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_agt での AGT HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables parameter checking.
Name	g_timer0	Module name.
Channel	0	Physical hardware channel.

ISDE Property	Value	Description
Mode	Periodic, One Shot Default: Periodic	Warning: One Shot functionality is not available in the GPT hardware, so it is implemented in software by stopping the timer in the ISR called when the period expires. For this reason, ISR's must be enabled for one-shot mode even if the callback is unused.
Period Value	10	See Timer Period Calculation
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz Default: Microseconds	See Timer Period Calculation
Auto Start	True, False Default: True	Set to true to start the timer after configuring or false to leave the timer stopped until is called.
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub Default: PCLKB	The clock source for the AGT counter.
AGTO Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for AGT (AGTO pin). Set to false for no output of the timer signal.
AGTIO Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for AGT (AGTIO pin). Set to false for no output of the timer signal.
Output Inverted	True, False Default: False	Set to false to start the output signal low. Set to true to start the output signal high.

ISDE Property	Value	Description
Enable comparator A output pin	True, False Default: False	Enable comparator A output pin selection
Enable comparator B output pin	True, False Default: False	Enable comparator B output pin selection
Callback	NULL	A user callback function can be registered in . If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the timer period elapses. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Timer interrupt priority. 0 is the highest priority.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

AGT HAL モジュールのクロック構成

AGT タイマは、PCLKB、LOCO、Fsub、または AGT アンダーフロー周波数に基づいてクロック供給されます。AGT クロックは、e^2 studio の [Properties] ウィンドウで選択できます。クロック周波数を設定するには、実行時に e^2 studio または CGC インタフェースのクロックコンフィギュレータを使用します。

AGT HAL モジュールのピン構成

AGT 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は関連するピンの選択例を示しています。

注 : 動作モードの選択によって、使用可能なペリフェラル信号および必要な MCU ピンが決定されます。

r_agt の AGT HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
AGT	Pins	Select Peripherals > Timer: AGT>AGT0

注 : この選択シーケンスでは、AGT0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_agt での AGT HAL モジュールのピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Timer Output, Compare Match, Count Measurement, Gated Count Default: Disabled	Select timer operation mode
AGTIO	None Default: None	AGTIO Pin
AGTO	None, P102 Default: P102	AGTO Pin
AGTOA	None Default: None	AGTOA Pin
AGTOB	None Default: None	AGTOB Pin
AGTEE	None, P101 Default: P101	AGTEE Pin

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

4.2.4.6 アプリケーションでの AGT HAL モジュールの使用

アプリケーションで AGT HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、AGT HAL モジュールを初期化します。
2) start API をコールして、AGT HAL モジュールを開始します。
3) counterGet API をコールして、カウンタ値を読み取ります。
4) periodSet API をコールして、期間値を読み取ります。
5) dutyCycleSet API を使用して、デューティサイクルを設定します。
6) infoGet API を使用して、タイマの情報を取得します。
7) 必要に応じて、AGT HAL モジュールのコールバックに応答します。
8) reset API を使用して、カウンタ値をリセットします。
9) stop API を使用して、AGT チャネルを停止します。
10) close へのコールを使用して、ペリフェラルの電源をオフにします。

注 : アプリケーションのニーズに基づいて、timer-period および duty-cycle パラメータを再構成できます。

これらの共通の手順を、以下の図に示す一般的な動作フローで説明します。

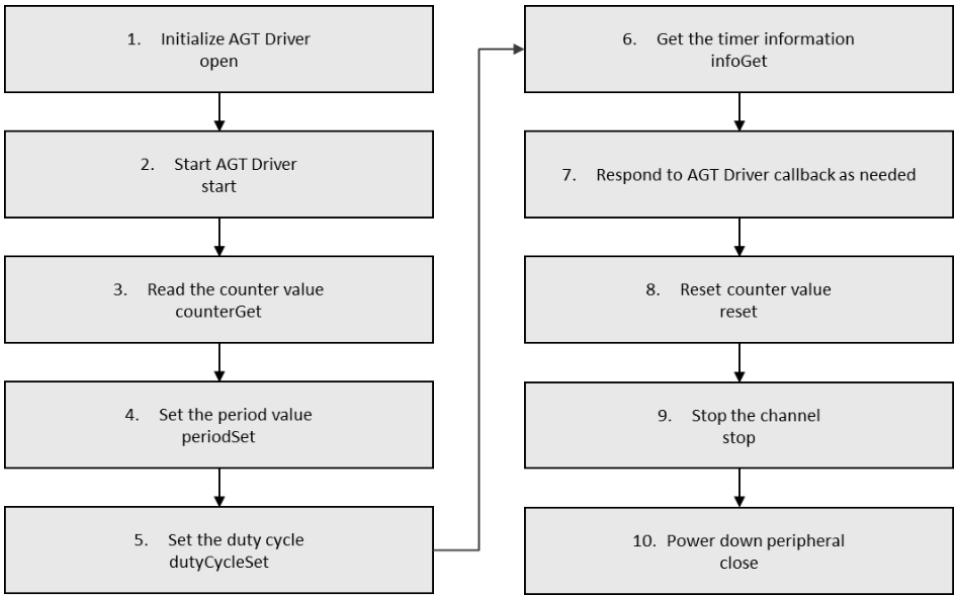


図 153:一般的な AGT HAL モジュールアプリケーションのフロー図

4.2.5 CAC ドライバー

CAC HAL モジュールは、クロック精度制御アプリケーション用のハイレベル API を提供し、Synergy MCU 上のクロック周波数精度測定回路（CAC）ペリフェラルを使用します。これは、信頼性を重視するアプリケーションのフェイルセーフメカニズムを実装する上で特に役立つ関数です。ユーザー定義のコールバックを作成し、さまざまなエラー表示に応答できます。

4.2.5.1 CAC HAL モジュールの特長

- 参照信号入力に基づいてクロック周波数測定および監視をサポートします。
- 参照は、外部から供給されるクロックソース、または内部クロックソースの可能性があります。
- 割り込み要求は、完了した測定、検出された周波数エラー、またはカウンタオーバーフローによってオプションで生成されることがあります。
- 外部から供給される参照クロックではデジタルフィルタが利用でき、除算値は内部提供の測定と参照クロックの両方で利用できます。
- 参照クロックのエッジ検出オプションは、上昇、下降、またはその両方で構成できます。

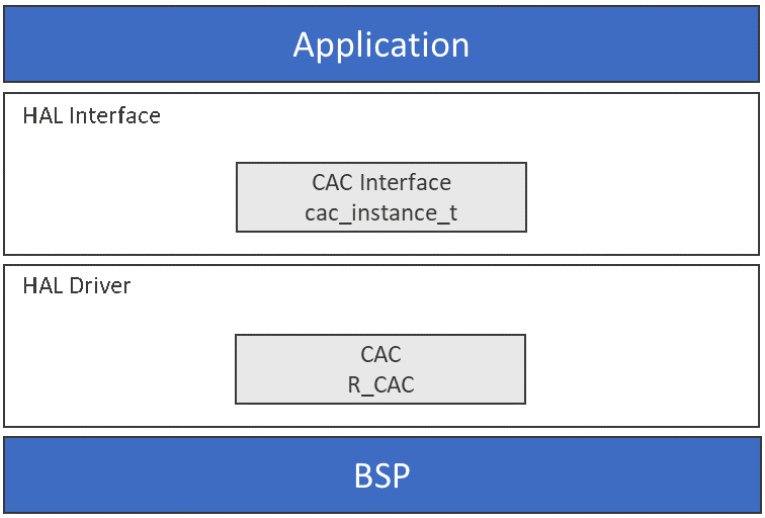


図 154:CAC HAL モジュールのブロック図

4.2.5.2 CAC HAL モジュールの API の概要

CAC HAL モジュールでは、CAC のオープン、クローズ、リード、開始、停止、リセットのための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

CAC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_cac0.p_api->open(g_cac0.p_ctrl, g_cac0.p_cfg) Open function for CAC device.
read	g_cac0.p_api->read(g_cac0.p_ctrl, &cac0_status, &cac0_counter) Read function for CAC peripheral.
close	g_cac0.p_api->close(g_cac0.p_ctrl) Close function for CAC device.
stopMeasurement	g_cac0.p_api->stopMeasurement(g_cac0.p_ctrl) Ends a measurement for the CAC peripheral.
startMeasurement	g_cac0.p_api->startMeasurement(g_cac0.p_ctrl) Begin a measurement for the CAC peripheral.
reset	g_cac0.p_api->reset(g_cac0.p_ctrl) Reset function for CAC device.
versionGet	g_cac0.p_api->versionGet(&cac0_version) Get the CAC API and code version information.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_ARGUMENT	One or more configuration options are invalid
SSP_ERR_NOT_OPEN	Open has not been successfully called
SSP_ERR_ASSERTION	Null provided for p_ctrl, p_cfg and others
SSP_ERR_INVALID_POINTER	Interrupt specified with NULL callback
SSP_ERR_HW_LOCKED	Hardware lock for CAC peripheral is already taken
SSP_ERR_INVALID_CAC_REF_CLOCK	Measured clock rate smaller than reference clock rate

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.5.3 CAC HAL モジュールの動作の概要

CAC HAL モジュールの API には、参照信号入力に基づいてクロック周波数を監視できるクロック周波数測定回路との間にインタフェースがあります。参照信号は、外部から供給されるクロックソース、またはいずれかの内部クロックソースの可能性があります。割り込み要求は、完了した測定、検出された周波数エラー、またはカウンタオーバーフローによってオプションで生成されることがあります。外部から供給される参照クロックではデジタルフィルタが利用でき、除算値は内部提供の測定と参照クロックの両方で利用できます。参照クロックのエッジ検出オプションは、上昇、下降、またはその両方で構成できます。

次のクロックの周波数を測定できます。

- メインクロック発振器からのクロック出力（メインクロック）
- サブクロック発振器からのクロック出力（サブクロック）
- 高速オンチップ発振器のクロック出力 (HOCO クロック)
- 中速オンチップ発振器のクロック出力 (MOCO クロック)
- 低速オンチップ発振器のクロック出力 (LOCO クロック)
- IWDT 専用オンチップ発振器のクロック出力 (IWDTCLK クロック)
- 周辺モジュール クロック (PCLKB)

測定クロックは参照クロックを使って監視されます。参照クロックとしては、外部クロック（CACREF 入力ピンで提供）または次のいずれかの内部クロックを使用できます。

- メインクロック発振器からのクロック出力（メインクロック）
- サブクロック発振器からのクロック出力（サブクロック）

- 高速オンチップ発振器のクロック出力 (HOCO クロック)
- 中速オンチップ発振器のクロック出力 (MOCO クロック)
- 低速オンチップ発振器のクロック出力 (LOCO クロック)
- IWDT 専用オンチップ発振器のクロック出力 (IWDTCLK クロック)
- 周辺モジュール クロック (PCLKB)

動作の説明

CAC HAL モジュールは、選択されたクロックの動作と精度を測定します。測定が要求されると、参照クロックで最初に検出された有効なエッジでカウントが始まり、次の有効なエッジで終わります。有効なエッジは、上昇、下降、またはその両方に設定できます。カウントは、クロックを 1、4、8、または 32 で分割可能な分周器回路を経由した後に、測定クロックの各サイクルでインクリメントされます。内部で提供される参照クロックも、クロックを 32、128、1024、または 8192 で分割可能な分周器回路を経由します。外部から提供される参照クロックは分周器回路を経由しませんが、設定されている場合はデジタルフィルタを経由することがあります。

たとえば、サブクロックが測定クロック（32kHz）として指定されており、除数が 1 に指定されている場合、カウンタは 32kHz のレートでインクリメントします。1kHz の参照クロックが提供されている場合、参照クロックの 1 サイクル後には、カウンタが 32 になることが予想されます。ここで、CAC の上限および下限の設定が確認されます。CAC 測定のセットアップの一部には、測定の上限と下限の仕様が含まれます。測定が完了すると、CAC はカウンターの内容と測定の制限を比較します。両方ともカウンタ？上限およびカウンタ？下限の場合、測定がエラーなしで完了し、測定された周波数は定義された制限内で運用されていたことになります。カウンタがこれらの条件を満たさなかった場合、周波数エラーが示されます。完了した測定は、API を呼び出してドライバーをポーリングするか、コールバック関数を設定することによって、識別できます。

CAC HAL モジュールの動作に関する重要な注意事項と制限事項

連続モード

CAC モジュールは、単一測定モードまたは連続測定モードで動作できます。連続モードでは、各測定の完了後に測定プロセスが再度開始されます。非連続モードつまり単一測定モードでは、最初の測定が完了した後で測定は停止します。この機能は、continuous_mode 構成メンバーによって制御されます。

割り込みとコールバック

ユーザーから（1 つ以上の割り込みが有効な）コールバックが提供されている場合、測定が完了すると、CAC HAL モジュールは、コールバック（p_callback）を呼び出して、イベント cac_event_t を示す引数 cac_callback_args_t を渡します。割り込みが有効になっていない場合は、API により測定が完了した（read）かどうかをポーリングして測定の状態を確認でき、測定の状態と CAC カウンタレジスタの現在値の両方が提供されます。

CAC ドライバーは 3 種類の割り込みをサポートしています。

- 周波数エラー割り込み。測定が完了し、CAC カウンタレジスタの値が open コールの一部として指定された範囲内ではない場合に発生します。この割り込みが生成されるためには、open コールで提供される構成の ferr_interrupt_enabled メンバーが有効になっている必要があります。
- オーバーフローエラー割り込み。CAC カウンタレジスタが最大値（0xFFFF）をオーバーフローすると発生します。この割り込みが生成されるためには、open コールで提供される構成の ovf_interrupt_enabled メンバーが有効になっている必要があります。
- 測定完了割り込み。測定が完了し、CAC カウンタレジスタの値が CAC ドライバー open の一部として指定された範囲内である場合に発生します。この割り込みが生成されるためには、open コールで提供される構成の mei_interrupt_enabled メンバーが有効になっている必要があります。

リセット

測定が停止された後、reset API を使用して、オーバーフロー、測定終了、周波数エラーの各割り込みフラグをリセットし、誤ったトリガを除去できます。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.5.4 アプリケーションへの CAC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに CAC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

クロック精度測定回路ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(クロック精度測定回路ドライバーのデフォルト名は g_cac0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

CAC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_comparator0 Comparator Driver on r_acmphs	Threads	New Stack> Driver> Analog> Comparator Driver on r_acmphs

次の図に示すように、r_cac のクロック精度測定回路ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

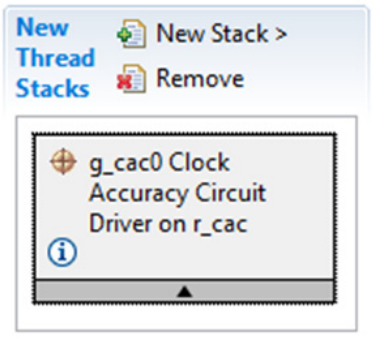


図 155:CAC HAL モジュールのスタック

4.2.5.5 CAC HAL モジュールの構成

ユーザーは必要な動作に合わせて CAC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_cac での CAC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Controls whether to include code for API parameter checking.
Name	g_cac0	Identifies this instance
Continuous Measurement Operation	Enabled, Disabled Default: Enabled	If Enabled, measurement will continuously restart after completing

ISDE Property	Value	Description
Measurement Complete Interrupt	Enabled, Disabled Default: Enabled	Enabling allows the CAC driver to generate an interrupt when a measurement is complete, providing the CAC MEASUREMENT END interrupt is enabled in the ICU.
Overflow Interrupt	Enabled, Disabled Default: Enabled	Enabling allows the CAC driver to generate an interrupt when a CAC overflow occurs, providing the CAC OVERFLOW interrupt is enabled in the ICU.
Frequency Error Interrupt	Enabled, Disabled Default: Enabled	Enabling allows the CAC driver to generate an interrupt when a frequency error occurs, providing the CAC FREQUENCY ERROR interrupt is enabled in the ICU.
Upper Limit Threshold	0	Top end of allowable range for measurement complete
Lower Limit Threshold	0	Bottom end of allowable range for measurement complete
Reference Clock Source	Main Oscillator, Sub-clock, HOCO, MOCO, LOCO, PCLKB, IWDT Default: Main Oscillator	Reference clock source
Reference Clock Divider	32,128,1024,8192 Default:32	Reference clock divider
Reference Clock Edge Detect	Rising, Falling, Both Default: Rising	Reference clock edge detection

ISDE Property	Value	Description
Reference Clock Digital Filter	Disabled, Sampling clock = measuring frequency, Sampling clock = measuring/4, Sampling clock = measuring/16 Default: Disabled	Reference clock Digital filter
Measurement Clock Source	Main Oscillator, Sub-clock, HOCO, MOCO, LOCO, PCLKB, IWDT Default: HOCO	Measurement clock source
Measurement Clock Divider	1,4,8,32 Default: 1	Measurement clock divider
Callback	NULL	Function name for callback
Frequency Error Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	CAC Frequency Error interrupt priority selection
Measurement End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	CAC Measurement End interrupt priority selection
Overflow Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	CAC Overflow interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

CAC HAL モジュールのクロック構成

アプリケーションの必要に応じて、[Clock] タブでクロックを選択します。

CAC HAL モジュールのピン構成

CAC HAL モジュールのピンは、次の表に示すように選択します。ピンの設定は後の表で示します。参照クロックの入力ピンに CACREF が使用されている場合は、このピンを構成する必要があります。

r_cac の CAC HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
CAC HAL Module	Pins	Peripherals > Monitoring: CAC > CAC0

注 : この選択シーケンスでは、CAC0 がドライバーに必要なハードウェアターゲットであることを想定しています。

CAC HAL モジュールのピン構成設定

Pin Configuration Property	Value	Description
Operation Mode	Disabled, External Reference Default: Disabled	Select Enable as the Operation Mode for CAC
CACREF	None, P204 Default: None	CACREF Pin

注 : 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトを想定したものです。他の Synergy キットと Synergy MCU では、使用可能なピン構成設定が異なる場合があります。

4.2.5.6 アプリケーションでの CAC HAL モジュールの使用

アプリケーションで CAC HAL モジュールを使用する一般的な手順は次のとおりです。

1) 必要に応じて、CGC clockStart API を使用して、参照クロックと測定クロックを開始します。
   a) 開始したクロックに対しては、CGC clockCheck API を使用して発振の安定性またはアクティブな状態を確認します。
2) 必要に応じて、versionGet を使用して、API とコードバージョンに関する情報を取得します。

3) open API を使用して、CAC HAL モジュールを初期化します。
4) startMeasurement API を使用して、測定を開始します。
5) read 関数を使用してポーリングを行い測定結果を調べるか、ISR でコールされるコールバック関数を使用して測定の状態と結果を取得します。
6) stopMeasurement API を使用して、測定を停止します。
7) reset API を使用して、オーバーフロー、測定終了、周波数エラーの各割り込みフラグをリセットします。
8) それ以上測定が必要ない場合は、close API を使用して、CAC HAL モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

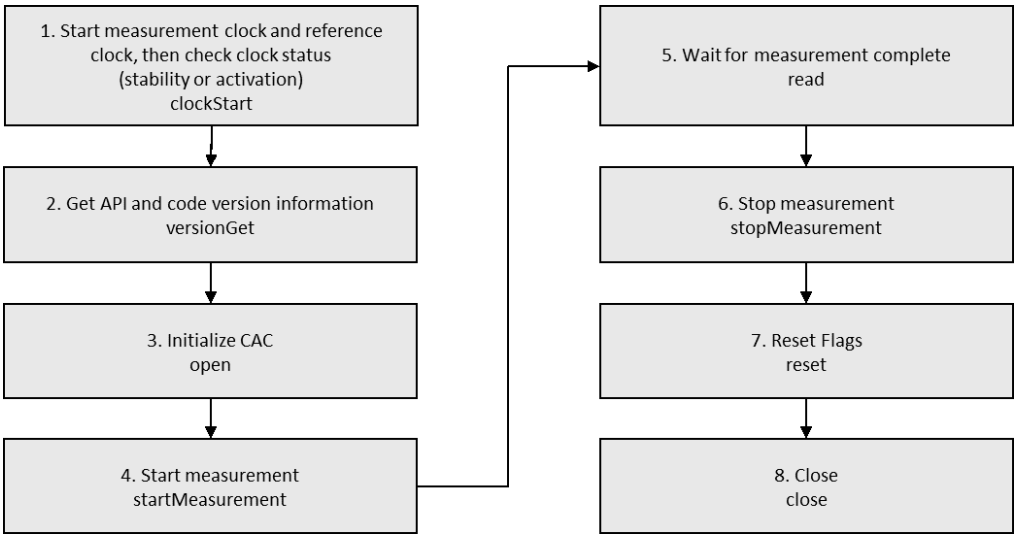


図 156:一般的な CAC HAL モジュールアプリケーションのフロー図

4.2.6 CAN ドライバー

CAN HAL モジュールは、CAN ネットワークアプリケーション用のハイレベル API で、Synergy マイクロコントローラハードウェア上で使用できる CAN ペリフェラルをサポートしています。ユーザーコールバック関数は、送信、受信、またはエラーの割り込みを受け取ったときにドライバーによって起動されるよう定義する必要があります。このコールバックは、チャネル、メールボックス、イベントを示すパラメータとともに返されます。

4.2.6.1 CAN HAL モジュールの特長

- 標準（11 ビット）と拡張（29 ビット）両方のメッセージ形式をサポートします
- CAN の仕様で定義されているビットタイミングの構成をサポートします
- 標準または拡張の ID フレームで最大 32 個の送信または受信メールボックスをサポートします
- 受信メールボックスは、データまたはリモート CAN フレームをキャプチャするように構成できます
- メールボックスマスクを使用して、ID の範囲を受信するよう受信メールボックスを設定できます

- 送信、受信、またはエラーの割り込みを受け取ったときのユーザーコールバック関数をサポートします

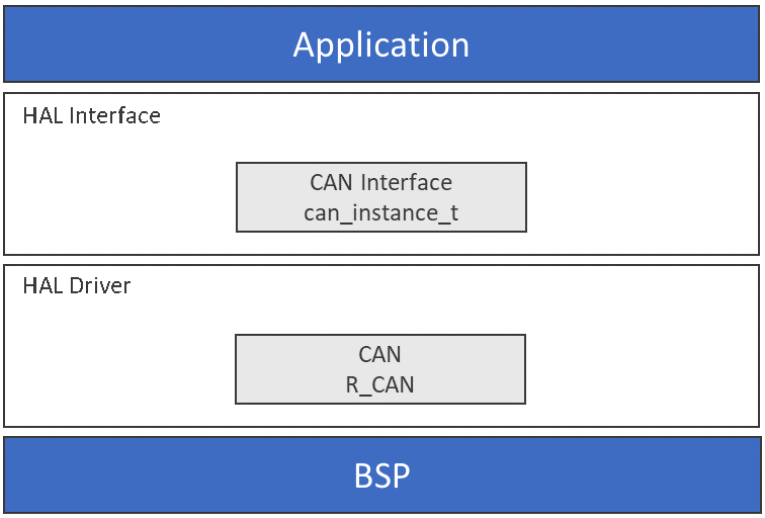


図 157:CAN HAL モジュールのブロック図

4.2.6.2 CAN HAL モジュールの API の概要

CAN HAL では、CAN データのオープン、クローズ、ライト（送信）、リード（受信）のための API が定義されています。また、さらに複雑なコマンドの処理を補助するための追加関数も提供されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

CAN HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_can0.p_api->open(g_can0.p_ctrl, g_can0.p_cfg) The open API configures CAN Channel 0. This function must be called before any other CAN functions. Note: This call is made automatically during system initialization, prior to entering the users thread. Unless the user closes the module, open will not need to be called.

Function Name	Example API Call and Description
close	g_can0.p_api->close(g_can0.p_ctrl) The close API handles the clean-up of internal driver data.
read	g_can0.p_api->read (g_can0.p_ctrl, p_args->mailbox, &receiveFrame) The read API reads received CAN data.
write	g_can0.p_api->write (g_can0.p_ctrl, 0, &transmitFrame) The write API write data into the CAN transmit frame buffer and send it out.

control	g_can0.p_api->control(g_can0.p_ctrl, CAN_COMMAND_MODE_SWITCH, &mode);
	
	withcan_mode_t mode = CAN_MODE_LOOPBACK_INTERNAL;
	
	The control API will be able to change the CAN mode of operation.

infoGet	g_can0.p_api->infoGet(g_can0.p_ctrl, p_info) The infoGet API retrieves the CAN mode of operation.
versionGet	g_can0.p_api->versionGet(version) The versionGet API retrieves the module version information.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_HW_LOCKED	Lock already owned by another user.
SSP_ERR_CAN_MODE_SWITCH_FAILED	Channel failed to switch modes.
SSP_ERR_CAN_INIT_FAILED	Channel failed to initialize.
SSP_ERR_ASSERTION	Null pointer presented.
SSP_ERR_NOT_OPEN	Port is not open.
SSP_ERR_CAN_DATA_UNAVAILABLE	No data available.
SSP_ERR_CAN_TRANSMIT_MAILBOX	Mailbox is not setup for receive.
SSP_ERR_CAN_TRANSMIT_NOT_READY	Transmit in progress, cannot write data at this time.
SSP_ERR_CAN_RECEIVE_MAILBOX	Mailbox is setup for receive and cannot send.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.6.3 CAN HAL モジュールの動作の概要

CAN HAL モジュールは、ユーザーの構成に従って Synergy マイクロコントローラ上の CAN 周辺機器を制御します。API は open、close、read、write、control、information の各関数を提供します。このドライバーでは、CAN の仕様で定義されているとおり、ビットタイミングの構成が可能です。これは、標準または拡張 ID フレームで最大 32 の送信または受信メールボックスで構成できます。受信メールボックスは、データまたはリモート CAN フレームをキャプチャするように構成できます。ユーザーコールバックは、送信、受信、またはエラーの割り込みが発生したときに、チャネル、メールボックス、イベントの情報と共に呼び出されます。

CAN の ID とマスクの使用

メッセージを受信するように構成された各 CAN メールボックスは、ID とマスクセットを持っています。受信メッセージは、以下の条件を満たす最も低いメールボックスに格納されます。

受信 ID & メールボックスマスク == メールボックス ID & メールボックスマスク

例 1: ID が 0x25 でマスクが 0x1FFFFFFF のメールボックスは、ID が 0x25 のメッセージを受信できます。

例 2: ID が 0x25 でマスクが 0x1FFFFFF0 のメールボックスは、ID が 0x20 ～ 0x2F のメッセージを受信できます。

CAN HAL モジュールのテストモードの使用

CAN モジュールには、受信待ちのみ、外部ループバック、内部ループバックの 3 つのテストモードがあります。

- 受信待ちのみモードでは、有効なデータフレームとリモートフレームを受信できます。ただし、CAN バスではレセシブビットのみを送信できます。ACK ビット、オーバーロードフラグ、アクティブエラーフラグは送信できません。受信待ちのみモードは、ボーレートの検出に使用できます。
- 外部ループバックモードでは、プロトコルモジュールは自分が送信したメッセージを CAN トランシーバによって受信されたメッセージ同じように扱い、受信メールボックスに格納します。外部シミュレーションと区別するため、プロトコルモジュールは ACK ビットを生成します。CTX ピンと CRX ピンをトランシーバに接続します。
- 内部ループバックモードは外部ループバックモードと似ていますが、プロトコルコントローラが内部 CTX ピンから内部 CRX ピンへの内部ループバックモードを実行することが異なります。外部 CRX ピンの入力値は無視されます。外部 CTX ピンはレセシブビットのみを出力します。CTX ピンと CRX ピンを、CAN バスまたは何らかの外部デバイスに接続する必要はありません。

CAN HAL モジュールの動作モードの変更

CAN モジュールのモードは、control API を使用して切り替えることができます。CAN_COMMAND_MODE_SWITCH と、変更後のモードを設定された can_mode_t 変数へのポインタを、control API に渡します。

Mode	can_mode_t value	Reason for use
Normal	CAN_MODE_NORMAL	Normal operation mode
Internal Loopback	CAN_MODE_LOOPBACK_INTERNAL	Internal loopback testing
External Loopback	CAN_MODE_LOOPBACK_EXTERNAL	External loopback testing
Listen Only	CAN_MODE_LISTEN	Baud rate detection
Halt	CAN_MODE_HALT	Mailbox configuration and test mode setting
Sleep	CAN_MODE_SLEEP	Stops the clock supply to the CAN module reducing current consumption
Exit Sleep	CAN_MODE_EXIT_SLEEP	Internal use only
Reset	CAN_MODE_RESET	Communication configuration

使用例：

```
/** Switch the device into internal loopback mode for easy testing. */

can_mode_t mode = CAN_MODE_LOOPBACK_INTERNAL;
```

```
ssp_err_t error = g_can0.p_api->control(g_can0.p_ctrl, CAN_COMMAND_MODE_SWITCH,
&mode);
```

CAN HAL モジュールの動作に関する重要な注意事項と制限事項

- 実行時にメインのクロック発振器（CANMCLK または XTAL）がまだ開始されていない場合は（たとえば、MCU クロックソースとして使用されていない場合）、ユーザーアプリケーションで CGC インタフェースを使用して開始する必要があります。
- S7、S5、S3、S1 の各 MCU では、クロックソースがメインのクロック発振器（CANMCLK）である場合、CAN HAL モジュールに対して次のクロック制限事項が満たされている必要があります。fPCLKB >= fCANCLK (XTAL/ ボーレートプリスケーラー)
- S7、S5、S3 の各 MCU では、クロックソースが PCLKB の場合、周辺モジュールクロックのソースは CAN HAL モジュールの PLL である必要があります。
- S3 MCU では、CAN HAL モジュールを使用する場合、PCLKA と PCLKB のクロック周波数の比が 2:1 である必要があります。他の設定では操作は保証されません。
- S1 MCU では、CAN HAL モジュールを使用する場合、ICLK と PCLKB のクロック周波数の比が 2:1 である必要があります。他の設定では操作は保証されません。
- SJW (同期ジャンプ幅) は多くの場合、バス管理者から供給されます。1 <= SJW <= 4 を選択します。
- タイムセグメントと SJW の設定は、次の制限に従う必要があります。TS1 > TS2 >= SJW。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.6.4 アプリケーションへの CAN HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに CAN HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

CAN ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(CAN ドライバーのデフォルト名は g_can0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

CAN HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_can0 CAN Driver on r_can	Threads	New Stack> Driver> Connectivity> CAN Driver on r_can

次の図に示すように、r_can の CAN ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

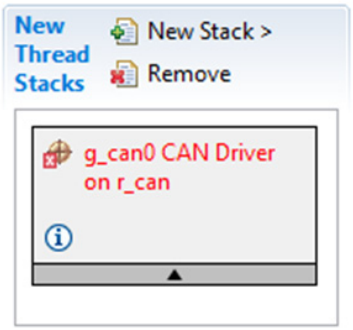


図 158:CAN HAL モジュールのスタック

4.2.6.5 CAN HAL モジュールの構成

ユーザーは必要な動作に合わせて CAN HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_can での CAN HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking

ISDE Property	Value	Description
Name	g_can0	Specify CAN Module instance name.
Channel	0	Specify if the CAN channel to use. 0 or 1 (S7G2 only).
Baud Rate Prescaler	5	Specify the baud rate prescaler(0-1023). See CAN Bit Rate Calculation.
Time Segment 1	4 Time Quanta thru 16 Time Quanta Default: 15 Time Quanta	Specify the time segment 1 value. (4-16) See CAN Bit Rate Calculation.
Time Segment 2	2 Time Quanta thru 8 Time Quanta Default: 8 Time Quanta	Specify the time segment 2 value (2-8). See CAN Bit Rate Calculation.
Synchronization Jump Width	1 Time Quanta thru 4 Time Quanta Default: 2 Time Quanta	Specify the Synchronization Jump Width value (1-4). See CAN Bit Rate Calculation.
Clock Source	PCLKB (S7G2 and S3A7 only), CAN MCLK Default: CAN MCLK	CAN clock source, CANMCLK or PCLKB (S7G2 and S3A7 only)

ISDE Property	Value	Description
Callback	NULL	A user callback function can be registered in open. If this callback function is provided, it is called from the interrupt service routine (ISR) each time any interrupt occurs. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Overwrite/Overrun Mode	Overwrite Mode, Overrun Mode Default: Overwrite Mode	Select whether receive mailbox will be overwritten or overrun if data is not read in time.
Standard or Extended ID Mode	Standard ID Mode, Extended ID Mode Default: Standard ID Mode	Select whether the driver will be using CAN standard or extended IDs.
Number of Mailboxes	4, 8, 16, 32 Mailboxes Default: 32 Mailboxes	Select 4, 8, 16 or 32 mailboxes.
Mailbox 0-31 ID	0-31	Select the receive ID for mailbox 0, between 0 and 0x7ff when using standard IDs, between 0 and 0x1FFFFFFF when using extended IDs. Value is not used when the mailbox is set as transmit type.
Mailbox 0 Type	Receive Mailbox, Transmit Mailbox Default: Transmit Mailbox	Select whether the mailbox is used for receive or transmit.

ISDE Property	Value	Description
Mailbox 1-31 Type	Receive Mailbox, Transmit Mailbox Default: Receive Mailbox	Select whether the mailbox is used for receive or transmit.
Mailbox 0 Frame Type	Data Mailbox, Remote Mailbox Default: Remote Mailbox	Select whether the mailbox is used to capture data frames or remote frames (receive only)
Mailbox 1-31 Frame Type	Data Mailbox, Remote Mailbox Default: Data Mailbox	Select whether the mailbox is used to capture data frames or remote frames (receive only)
Mailbox 0-3 Group Mask	0x1FFFFFFF	Select the Mask for mailboxes 0-3. See Setting the Mailbox Group Masks
Mailbox 4-7 Group Mask	0x1FFFFFFF	Select the Mask for mailboxes 4-7. See Setting the Mailbox Group Masks
Mailbox 8-11 Group Mask	0x1FFFFFFF	Select the Mask for mailboxes 8-11. See Setting the Mailbox Group Masks
Mailbox 12-15 Group Mask	0x1FFFFFFF	Select the Mask for mailboxes 12-15. See Setting the Mailbox Group Masks
Mailbox 16-19 Group Mask	0x1FFFFFFF	Select the Mask for mailboxes 16-19. See Setting the Mailbox Group Masks
Mailbox 20-23 Group Mask	0x1FFFFFFF	Select the Mask for mailboxes 20-23. See Setting the Mailbox Group Masks
Mailbox 24-27 Group Mask	0x1FFFFFFF	Select the Mask for mailboxes 24-27. See Setting the Mailbox Group Masks

ISDE Property	Value	Description
Mailbox 28-31 Group Mask	0x1FFFFFFF	Select the Mask for mailboxes 28-31. See Setting the Mailbox Group Masks
Error Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Specify the error interrupt priority 0-15 (required).
Receive Mailbox Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Specify the receive interrupt priority 0-15 (required).
Transmit Mailbox Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Specify the transmit interrupt priority 0-15 (required).

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

CAN HAL モジュールのクロック構成

CAN ペリフェラルは、CANMCLK（メインクロック発振器）または PCLKB（S7G2、S5D9、S5D5、S3A7、S3A7 のみ）をクロックソース（fCAN、CAN システムクロック）として使用します。PCLKB をデフォルトの 60MHz で、Synergy をデフォルト（S7G2 DK）の CAN 構成で使用すると、CAN ビットレートは 500Kbit になります。

PCLKB の周波数を設定するには、e^2 studio のクロックコンフィギュレータを使用します。実行時にクロック周波数を変更するには、CGC インタフェースを使用します。クロックの構成の詳細については、CGC モジュールのガイドを参照してください。

CAN HAL モジュールのピン構成

CAN 周辺モジュールは、MCU のピンを使用して、CAN バスに接続されている外部デバイスと通信します。[Peripherals] で [CAN] を選択し、続いて [CAN0] でチャネル 0、または [CAN1]（S7G2 と S3A7 のみ）でチャネル 1 を選択します。チャネルの動作モードを有効にし、PC ボードのレイアウトと一致するように CRXn ピンと CTXn ピンを選択する必要があります。ピン コンフィギュレーターは、pin_cfg フィールドで関連するピンの CAN ピン設定を適切に構成します。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では CAN ピンの選択例を示します。

注 : 動作モードの選択によって、使用可能なペリフェラル信号および必要な MCU ピンが決定されます。

r_can の CAN HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
CAN	Pins	Select Peripherals > CAN>CAN0

注 : この選択シーケンスでは、CAN0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_can での CAN HAL モジュールのピン構成設定

Property	Value	Description
Operation Mode	Disabled, Enabled	Enable the mode to use CAN0
CRX	None, P202, P402 Default: P402	CAN0_CRX0
CTX	None, P203 P401 Default: P401	CAN0_CTX0

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

CAN ビットレートの計算

タイムクォンタム（Tq）は、CAN 通信クロック fCANCLK の 1 ビットタイムです。これは CAN ビットタイムではなく、CAN 周辺機器の内部クロック周期です。周波数はボーレートプリスケーラの値と CAN ソースクロック fCAN（CANMCLK または PCLKB）によって決定されます。1 ビットタイムは複数のタイムクォンタム Tqtot に分割されます。1 タイムクォンタムは fCANCLK の周期と同じです。各ビットレートレジスタには、1 CAN ビット周期を構成する合計 Tq のうち、一定の数の Tq が付与されます。デフォルトの ISDE ビットレート設定（S7G2 DK テンプレート）は、fCAN が 60MHz の場合、500Kbps です（PCLKB を使用）。

ビットレートレジスタ設定を計算するための式は次のとおりです。

fCAN = (fPCLKB または fCANMCLK)

ボーレートプリスケーラは CAN 周辺クロックをスケールダウンします。

fCANCLK = fCAN/ ボー レート プリスケーラー = 60 MHz (デフォルト)/ 5 (デフォルト) = 12 MHz

1 タイムクォンタムは CAN クロックの 1 クロック周期です。

Tqtot =1/fCANCLK

Tqtot は 1 CAN ビット タイムあたりの CAN 周辺クロック サイクルの合計数で、「タイム セグメント」と「SS」(常に 1) の合計です。

Tqtot = TSEG1 + TSEG2 + SS (TSEG1 > TSEG2) = 15 + 8 + 1 = 24 (デフォルト)

この場合、ビットレートは次のようにして計算します。

ビットレート = fCANCLK/Tqtot = 12 MHz / 24 = 500 Kbps

重要な注意 :

- 実行時にメインのクロック発振器（CANMCLK または XTAL）がまだ開始されていない場合は（たとえば、MCU クロックソースとして使用されていない場合）、ユーザーアプリケーションで CGC インタフェースを使用して開始する必要があります。
- S7G2、S3A7、S124 の各 MCU では、クロックソースがメインのクロック発振器（CANMCLK）である場合、CAN モジュールに対して次のクロック制限事項が満たされている必要があります。fPCLKB >= fCANCLK (fCAN/ ボーレートプリスケーラー)
- S7G2 および S3A7 MCU では、クロックソースが PCLKB の場合、周辺モジュールクロックのソースは CAN HAL モジュールの PLL である必要があります。
- S3A7 MCU では、CAN HAL モジュールを使用する場合、PCLKA と PCLKB のクロック周波数の比が 2:1 である必要があります。他の設定では操作は保証されません。
- S124 MCU では、CAN HAL モジュールを使用する場合、ICLK と PCLKB のクロック周波数の比が 2:1 である必要があります。他の設定では操作は保証されません。
- SJW (同期ジャンプ幅) は多くの場合、バス管理者から供給されます。1 <= SJW <= 4 を選択します。

異なる CAN ビットレートに対するコンフィギュレータのサンプル値

fCAN (either PCLKB or CAN MCLK)	Baud Rate Prescalar	fCANCLK = fCAN/Baude Rate Prescalar	Time Segment 1 (TSEG1)	Time Segment 2 (TSEG2)	Synchronization Jump Width (SS)	Tqtot = TSEG1 + TSEG2 +SS	**Bitrate = fCAN
240	10	24	15	8	1	24	1 Mbps
60	5	12	15	8	1	24	500 kbps
240	48	5	16	2	2	20	250 kbps
240	96	2.5	16	2	2	20	125 kbps

メールボックス グループ マスクの設定

4 つのメールボックスのグループごとに、合計 8 つのメールボックスグループマスクがあります。これらのマスクでは、複数の ID を受信できるようにメールボックスを設定できます。マスクがすべて 1 に設定されている場合（標準

ID では 0x7ff、拡張 ID では 0x1FFFFFFF)、そのグループ内のメールボックスは ID のビットをマスクせず、メッセージがキャプチャされる前にメールボックス ID のすべてのビットがそのメールボックス ID に一致しなければなりません。マスクのビットの中に 0 に設定されているものがある場合、これらのビットはメールボックス ID の同じビットと一致する必要はありません。たとえば、メールボックス ID 1 が 0x7ff に設定され、メールボックス 0 ～ 3 のグループ マスクが 0x7ff に設定されている場合、メールボックス 1 は ID が 0x7ff であるメッセージのみをキャプチャします。メールボックス 0 ～ 3 のグループマスクが 0x7fe に設定されている場合、メールボックス 1 は ID が 0x7f のメッセージに加え、ID が 0x7fe のメッセージもキャプチャします。

4.2.6.6 アプリケーションでの CAN HAL モジュールの使用

CAN アプリケーションで CAN 通信を行うには、少なくとも 2 つのノードが必要です。一方のノードはトランスミッタで、他のノードはレシーバです（または、両方がトランスミッタかつレシーバとして動作することもできます)。

アプリケーションで CAN HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、CAN HAL モジュールを初期化します。
2) （オプション）control API を使用して、内部ループバックまたは外部ループバックのテストモードに入ります。
3) （オプション）モジュールの状態に関する情報（ビットレートなど）を、infoGet API を使用して取得できます。
4) メッセージを送信するには：
   a) CAN フレームを作成し、正しい ID とフレームタイプに構成します。
   b) write API を使用して、送信モードに構成されているメールボックスに CAN フレームを書き込みます。
5) メッセージを受信するには：
   a) read API を使用して、フレームを受信したメールボックスから読み取ります。
6) close API を使用して、CAN HAL モジュールを閉じます（必要な場合)。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

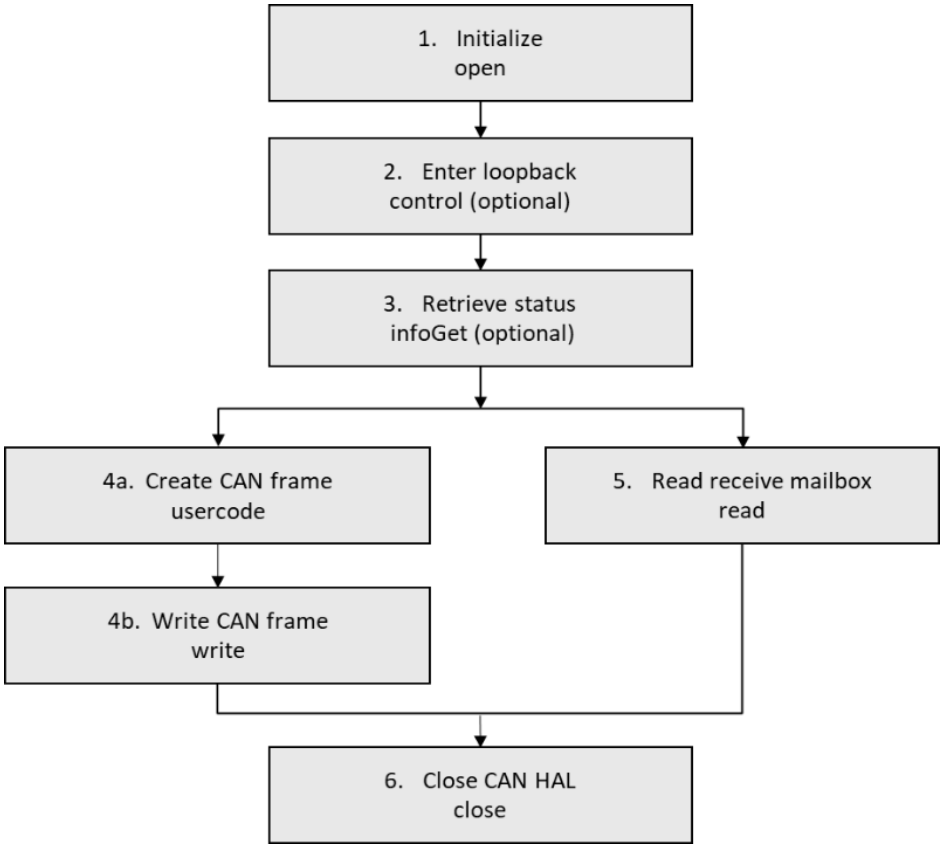


図 159:一般的な CAN HAL モジュールアプリケーションのフロー図

4.2.7 CGC ドライバー

CGC HAL モジュールは、クロック制御アプリケーション用のハイレベル API を提供し、クロック制御ペリフェラルを使用して Synergy MCU のクロック制御機能を構成および制御します。すべてのプロジェクトでクロック機能が必要となるため、CGC HAL モジュールはデフォルトでプロジェクトに追加されます（モジュールは、統合ソリューション開発環境内で構成されます）。ユーザー定義のコールバックを作成し、メイン発振器が停止したときに信号を送信できます。

4.2.7.1 CGC HAL モジュールの特長

CGC HAL モジュールは、Synergy MCU のさまざまなクロック機能の構成と制御をサポートします。主な特長を以下に示します。

- システムクロックソースを選択します
  - HOCO（高速オンチップ発振器）、MOCO（中速オンチップ発振器）、LOCO（低速オンチップ発振器）、メインクロック、PLL、またはサブ発振器
- 内部クロックを構成し、オンまたはオフにします

- 出力クロックを構成します
- 発振停止検出機能を設定します
- 最大 6 つの各クロックドメインにクロック除数を設定します
- 一部の Synergy MCU は、独立した分周が可能な外部クロック出力もサポートしています

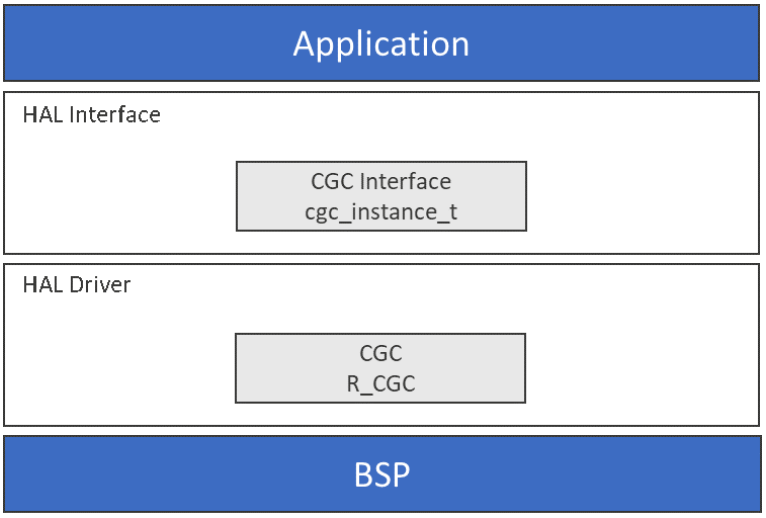


図 160:CGC HAL モジュールのブロック図

4.2.7.2 CGC HAL モジュールの API の概要

CGC HAL モジュールでは、MCU クロックの初期化、開始、制御、停止のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の API の要約の表に含まれます。ステータス戻り値の表は API の要約の後にあります。

CGC HAL モジュールの API の要約

Function Name	Example API Call and Description
init	g_cgc.p_api->init(); Initial clock configuration called by BSP automatically.
clocksCfg	g_cgc.p_api->clocksCfg(&p_clock_cfg); The BSP calls this function at startup, but it can also be called from the application to change clocks at runtime.

Function Name	Example API Call and Description
clockStart	g_cgc.p_api->clockStart(clock_source, &p_clock_cfg); Start a clock.
clockStop	g_cgc.p_api->clockStop(clock_source); Stop a clock.
systemClockSet	g_cgc.p_api->systemClockSet(clock_source, &p_clock_cfg); Set the system clock.
systemClockGet	g_cgc.p_api->systemClockGet(&clock_source, &clock_config); Get the system clock information.
systemClockFreqGet	g_cgc.p_api->systemClockFreqGet(&clock_source, &frequency_hz); Return the frequency of the selected clock.
clockCheck	g_cgc.p_api->clockCheck(clock_source); Check the stability of the selected clock.
oscStopDetect	g_cgc.p_api->oscStopDetect(callback, enable); Configure the Main Oscillator stop detection.
oscStopStatusClear	g_cgc.p_api->oscStopStatusClear(); Clear the oscillator stop detection flag.

Function Name	Example API Call and Description
busClockOutCfg	g_cgc.p_api->busClockOutCfg (divider); Configure the bus clock output secondary divider. The primary divider is set using the BSP clock configuration and the systemClockSet function (S7G2 and S3A7 MCU only).
busClockOutEnable	g_cgc.p_api->busClockOutEnable (); Enable the bus clock output (S7G2 and S3A7 MCU only).
busClockOutDisable	g_cgc.p_api->busClockOutDisable (); Disable the bus clock output (S7G2 and S3A7 MCU only).
clockOutCfg	g_cgc.p_api->clockOutCfg(clock_source, clock_dividers); Configure clockOut.
clockOutEnable	g_cgc.p_api->clockOutEnable(); Enable clock output on the CLKOUT pin. The source of the clock is controlled by clockOutCfg.
clockOutDisable	g_cgc.p_api->clockOutDisable(); Disable clock output on the CLKOUT pin. The source of the clock is controlled by clockOutCfg.
lcdClockCfg	g_cgc.p_api->lcdClockCfg(clock); Configure the segment LCD Clock (S3A7 and S124 MCUs only).

Function Name	Example API Call and Description
lcdClockEnable	g_cgc.p_api->lcdClockEnable(); Enable the LCD clock (S3A7 and S124 MCUs only).
lcdClockDisable	g_cgc.p_api->lcdClockDisable(); Disables the LCD clock (S3A7 and S124 MCUs only).
sdramClockOutEnable	g_cgc.p_api->sdramClockOutEnable(); Enables the SDRAM clock output (S7G2 MCU only).
sdramClockOutDisable	g_cgc.p_api->sdramClockOutDisable(); Disables the SDRAM clock (S7G2 only).
usbClockCfg	g_cgc.p_api->usbClockCfg(divider); Configures the USB clock (S7G2 only).
systickUpdate	g_cgc.p_api->systickUpdate(period_count, units); Update the Systick timer.
versionGet	g_cgc.p_api->versionGet(&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_ABORTED	Attempt to update systick timer failed.
SSP_ERR_HARDWARE_TIMEOUT	Hardware timed out.
SSP_ERR_STABILIZED	Clock stabilized.
SSP_ERR_CLOCK_INACTIVE	Clock not turned on.
SSP_ERR_MAIN_OCO_INACTIVE	Main OCO off/unstable.
SSP_ERR_CLOCK_ACTIVE	Clock active.
SSP_ERR_NOT_STABILIZED	Clock source un-stabilized.
SSP_ERR_CLKOUT_EXCEEDED	Clock out exceeded.
SSP_ERR_NULL_PTR	Pointer null.
SSP_ERR_OSC_DET_ENABLED	Oscillation stop detection enabled.
SSP_ERR_OSC_STOP_DETECTED	The Oscillation stop detect status flag is set. Under this condition it is not possible to disable the Oscillation stop detection function.
SSP_ERR_OSC_STOP_CLOCK_ACTIVE	The Oscillation Detect Status flag cannot be cleared if the Main Osc or PLL is set as the system clock. Change the system clock before attempting to clear this bit.
SSP_ERR_INVALID_ARGUMENT	Invalid argument.
SSP_ERR_INVALID_MODE	Attempt to start a clock in a restricted operating power control mode.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.7.3 CGC HAL モジュールの動作の概要

CGC HAL モジュールインタフェースでは、CGC HAL モジュールのすべての機能を構成し、使用することができます。たとえば、複数のクロックソースをシステムクロックソースとして選択できます。また、システムクロックを分周することで、システムや周辺機能のニーズに適した多様な周波数を提供できます。

クロックの安定性を確認し、クロックが不要なときは停止して電力を節減することができます。API には、実行時にシステムおよびシステム周辺クロックの周波数を返す関数があります。メイン発振器の停止を検出する機能があり、それにはユーザー定義のコールバック関数を呼び出すオプションも用意されています。

CGC HAL モジュールは以下の目的で使用できます。

- 使用可能な任意のクロック（HOCO、MOCO、LOCO、メインクロック、PLL、サブ発振器）をシステムクロックソースとして設定します
- 内部クロックを構成します（ICLK、PCLK など）。
- クロックのオンとオフを切り替えます
- 出力クロックを構成します
- 発振停止検出機能を設定します

クロック生成回路周辺機器の特長は、次の発振器とクロック発振器です。

- メイン発振器入力（最大 24 MHz）
- 32.768kHz サブクロック発振器
- 64MHz で動作する HOCO（デバイスのバージョンに依存）
- 8MHz で動作する MOCO
- 32.768kHz で動作する LOCO
- 24MHz ～ 240MHz で動作する PLL 回路出力（デバイスに依存）

Synergy マイクロコントローラーには、6 つの内部クロック ドメインがあります。各ドメインは独立した分周ができますが、システム クロック制御レジスタで選択したクロック入力に依存します。以下の除数があります。

- ICLK – コアクロック。CPU、DMAC、ROM、RAM 用（最大 32/48/240MHz）
- PCLKA – 周辺クロック。EtherC、EDMAC、USB2.0 HS、QSPI、および SCIF を含むモジュール用（最大 32/48/120MHz）
- PCLKB – 周辺クロック。IIC、CAN、DAC12、RTC、USBFS、I/O ポート、WDT、IWDT などのモジュール用（最大 32/60MHz）
- PCLKC – 周辺クロック。ADC12 変換クロック用（最大 64/60MHz）
- PCLKD – 周辺クロック。GPT カウントクロック用（最大 64/120MHz）
- FCLK – フラッシュメモリ用クロックソース（最大 32/60MHz）

また、一部の Synergy マイクロコントローラーでは、制御可能な外部クロック出力がサポートされており、そのいくつかには独立した除数があります。以下の除数があります。

- CLKOUT – CLOCKOUT/BUZZER クロック (最大 24 MHz) (独立したクロック セレクターと除数)
- BCLK – 外部バスコントローラへの外部バスクロック（最大 16/120MHz）
- SDCLK – SDRAM クロック (最大 120 MHz)
- UCLK – USB クロック (最大 120 MHz) (Synergy バージョン 2 用の独立した除数 / セレクター)
- LCD_CLK – LCD クロック（独立したクロックセレクタ。ただし除数なし）

CGC HAL モジュールによる実行時のシステムクロックペリフェラルクロック除数の変更

CGC HAL モジュールには、実行時に clocksCfg API 関数を使用して、システムクロックおよびクロックツリーの設定を変更するオプションもあります。clocksCfg API 関数で使用するための構成構造体を作成するには、[New Stack] > [System] > [CGC Configuration Instance] の順に選択します。

clocksCfg 関数により、システムクロック、ペリフェラルクロック分周器、PLL 乗算値および分周器、システムクロックの状態（停止 / 開始）（HOCO、主発振器、サブクロック発振器など）を変更できます。次の図のオプションは、システムクロックを HOCO から MOCO に変更し、周辺クロック分周器を更新している例です。各クロックに対するオプションは、開始、停止、なし（変更なしという意味）です。すべての MCU で、すべてのクロックが利用可能とは限りません。すべての MCU で、すべての周辺クロックが利用可能とは限りません。

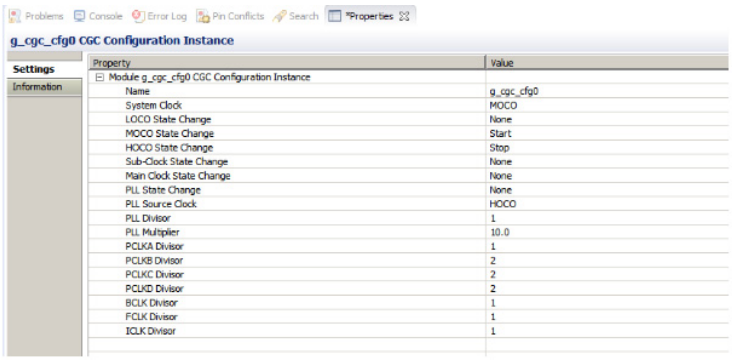


図 161:CGC の構成プロパティ

上の例の関数呼び出しは次のようになります。

```
g_cgc.p_api->clocksCfg(&g_cgc_cfg0);
```

CGC HAL モジュールオプション設定メモリ

すべての Synergy マイクロコントローラにはオプション設定メモリが含まれます。このメモリを使用して、リセット後の周辺機能の動作状態を設定できます。OFS は、IWDT、WDT、LVD、CGC HOCO の状態の設定に使用できます。次の表では、OFS レジスタで構成できる CGC HOCO パラメータの一覧を示します。

OFS レジスタで設定できる項目

Control	Description
HOCO oscillation enable	Automatically starts the HOCO after a Reset, if enabled
HOCO Frequency	S7 and S5 Series 16MHz 18MHz 20MHz S3 and S1 Series 24 MHz 32 MHz 48 MHz 64 MHz

OFS レジスタの値は、プロパティダイアログで設定できます。プロパティダイアログは、Synergy 構成エディタで [BSP] タブを選択して使用できます。

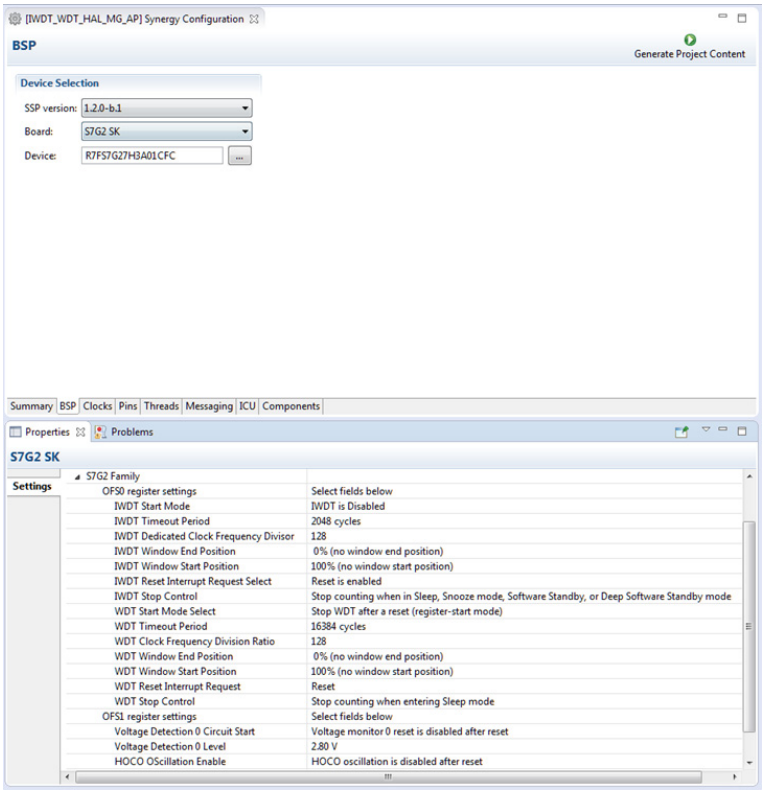


図 162:OFS のレジスタ設定

ローパワー動作

低消費電力モードバージョン 2 の HAL モジュールは MCU の動作電力制御モードを処理しなくなるので、CGC HAL モジュールが MCU の動作電力制御モードも処理します。

動作電力制御モードがサブ発振器速度モードに設定されている場合は、サブ発振器と LOCO クロックを除くすべてのシステムクロックはオフになります。さらに消費電力を低減するために、アプリケーションプログラムは clockStop API 関数をコールして、LOCO クロックをオフにすることができます。

CGC HAL モジュールの動作に関する重要な注意事項と制限事項

- CGC HAL モジュールは、ThreadX RTOS に依存しません。
- CGC HAL モジュールは MCU のコア機能であり、BSP 初期化プロセスの後で設定されます。多くの場合、CGC を変更する必要はありません。ただし、CGC HAL モジュールにはクロック構成変更関数が用意されており、アプリケーションの要件に応じて動作速度の要件と電力消費のバランスを取ることができます。
- Synergy マイクロコントローラの CGC 周辺機能は、発振器の停止検出をサポートします。アプリケーションで有効にすると、発振器停止検出機能は、主 /PLL クロックが停止したかどうかを自動的に検出して、MOCO に動作を切り替えます。SSP でこの機能を有効にするときは、ユーザーが手動でコールバック関数を作成する必要があります。この手順について以下で詳しく説明します。

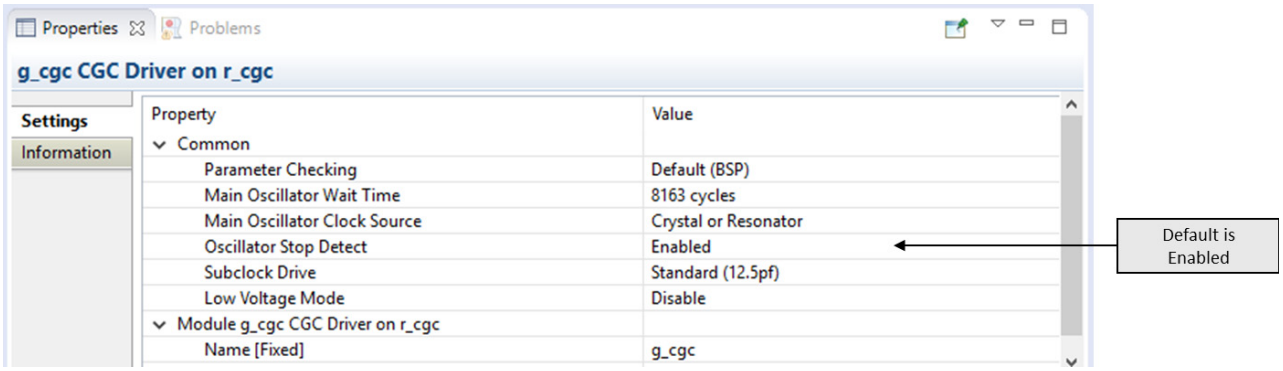


図 163:発振器停止検出の有効化 / 無効化

注 :SSP 1.4.0 以降では、[Oscillator Stop Detect] プロパティはデフォルトで [Enabled] に設定されます。以前のバージョンの SSP では、デフォルトは [Disabled] に設定されていました。

- [Configure Subclock Drive on Reset] が [Disabled] に設定されている場合は、開始時にサブクロックは構成されません。サブクロックを構成するには、ユーザーは弱くリンクされた関数 R_BSP_WarmStart(). の定義を作成する必要があります。この関数では、引数が "BSP_WARM_START_POST_C", である場合、ユーザーは API clockStart clockStart をコールしてサブクロックを開始できます。

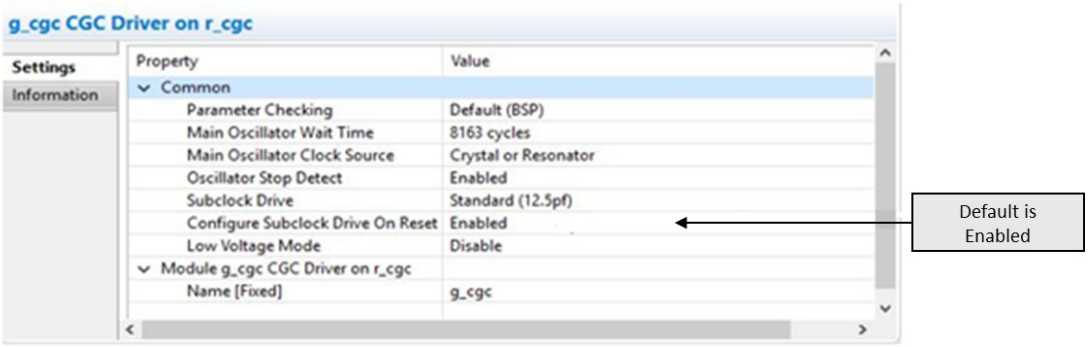


図 164:Configure Sub-Clock Drive on Reset

アプリケーションのコードで、コールバック関数を作成します。この例では、osc_stop_callback. という名前にします。

```
void osc_stop_callback(cgc_callback_args_t * p_args)
{
    /* perform Oscillator Stop Detection processing */
}
```

前に宣言したコールバックで API を呼び出すことにより、発振器停止検出を有効にします。

```
/* Enable the Osc Stop Detect functionality */

g_cgc.p_api->oscStopDetect( osc_stop_callback, true );
```

ICU 内で割り込みを有効にします。

```
/* Osc Stop Detect is an NMI interrupt.  Enable the NMI in ICU */

R_ICU->NMIER_b.OSTEN = 1;
```

このモジュールの使用に関する制限事項については、SSP の最新のリリースノートを参照してください。

4.2.7.4 アプリケーションへの CGC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに CGC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

CGC ドライバーは HAL/ 共通スレッドに自動的に追加されるので、削除された場合に新しいスレッドに追加することだけが必要です。(CGC のデフォルト名は g_cgc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

CGC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_cgc CGC HAL on r_cgc	Threads	New Stack> Driver> System> CGC Driver on r_cgc

次の図に示すように、r_cgc の CGC ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

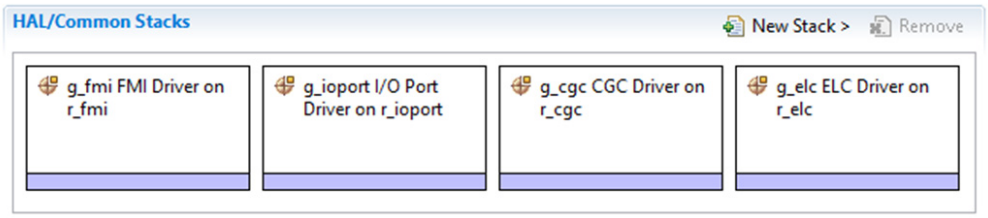


図 165:CGC HAL モジュールのスタック

4.2.7.5 CGC HAL モジュールの構成

ユーザーは必要な動作に合わせて CGC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_cgc での CGC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Main Oscillator Wait Time	3, 35, 67, 131, 259, 547, 1059, 2147, 4291, 8163 cycles Default: 8163 cycles	Set to 0 if a resonator, or crystal, is used. Set to 1 if an external oscillator input is used.
Main Oscillator Clock Source	External Oscillator, Crystal or Resonator Default: Crystal or Resonator	Set to one of these values. It should be at least as long as the main clock stabilization time. This delay will be configured only if #define CGC_CFG_MAIN_OSC_CLOCK_SOURCE is set to 0, indicating that a resonator/ crystal is used. Set the main clock oscillation stabilization time to longer than or equal to the stabilization time recommended by the oscillator manufacturer.

ISDE Property	Value	Description
Oscillator Stop Detect	Enabled, Disabled Default: Enabled	This allows the R_CGC_OscStopDetect function code to be generated if enabled. The user must call this function with a callback pointer to use this feature.
Subclock Drive	Middle (4.4 pf), Standard (12.5 pf) Default: Standard (12.5 pf)	This setting is for matching the subclock oscillator drive capacitance based on the crystal parameters #define CGC_CFG_SUBCLOCK_ DRIVE.
Low Voltage Mode	Enable, Disable Default: Disable	Low voltage mode selection.
Name	g_cgc	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

CGC HAL モジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、クロックを選択的にオンまたはオフにしたり、電力と性能の特性を最適化するために周波数を変更したりすることが役に立つ場合があります。

注 : モジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

CGC HAL モジュールのクロック構成

BSP 初期化プロセスによって設定されるデフォルトの CGC HAL モジュールクロック周波数は、ISDE でコンフィギュレータの [Clocks] タブを使用して構成できます。無効な選択肢を選択すると、赤く表示されます。

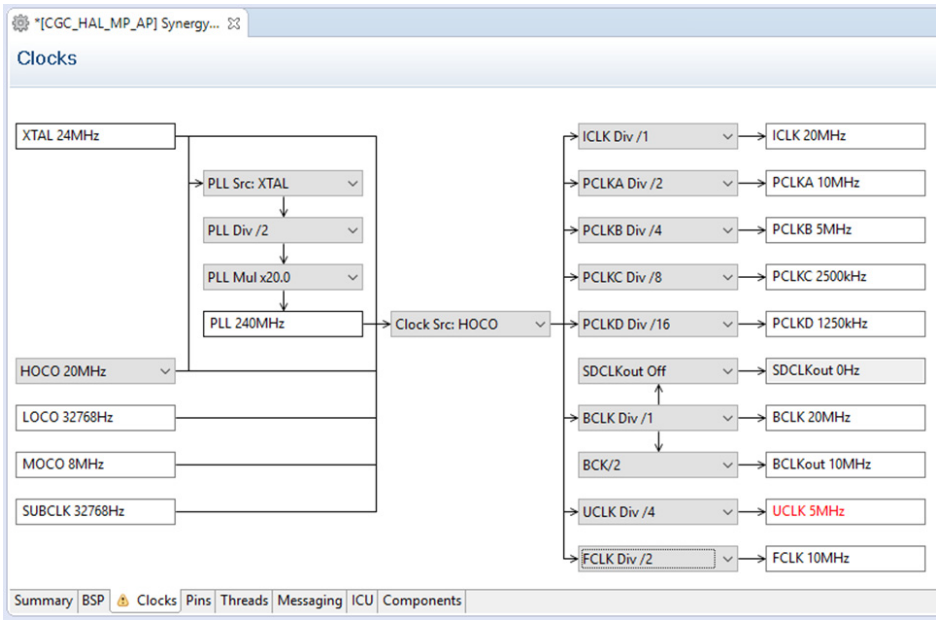


図 166:[Clocks] タブでのデフォルトのクロック設定

この例では、クロックソースは HOCO であり、周辺クロックに対してさまざまなクロック分周器が選択されています。有効な USB クロック（UCLK）を実現できない場合は、赤で強調表示されます。これはあくまでもアドバイスであり、このようなクロック周波数が要求されても、プロジェクトはビルドされます。

CGC HAL モジュールのピン構成

CGC 周辺モジュールは、BCLK および SDCLK 信号の出力を制御します。この機能を有効または無効にするには、[Clocks] タブを使用します。必要に応じて、[Pins] タブで BCLK_SDCLK I/O ピンを選択して構成する必要があります。

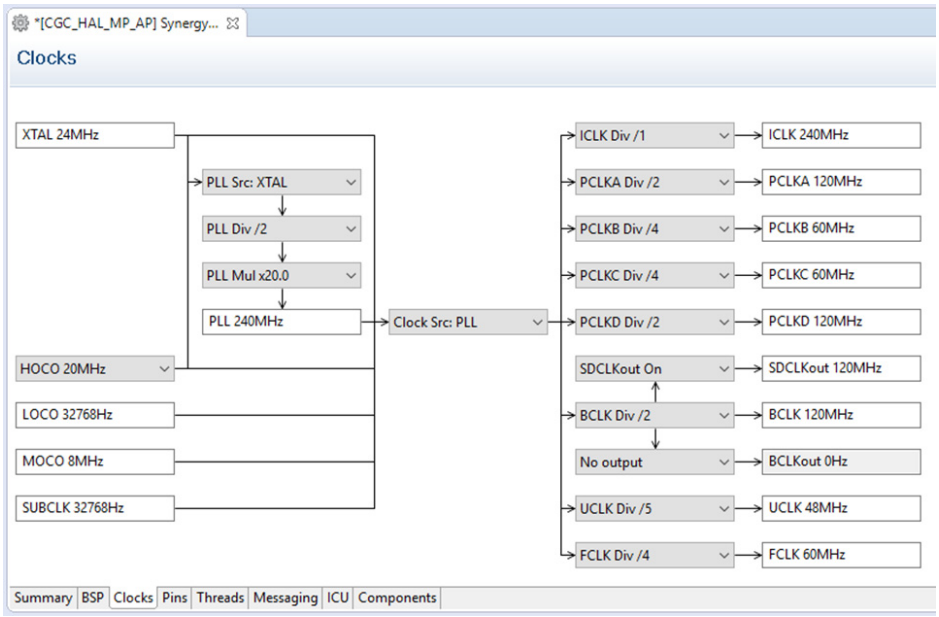


図 167:[Clock] タブでの SDCLK および BCLK の有効化 / 無効化

この例では、SDRAM クロックが有効にされ、BUS クロックが無効にされています。

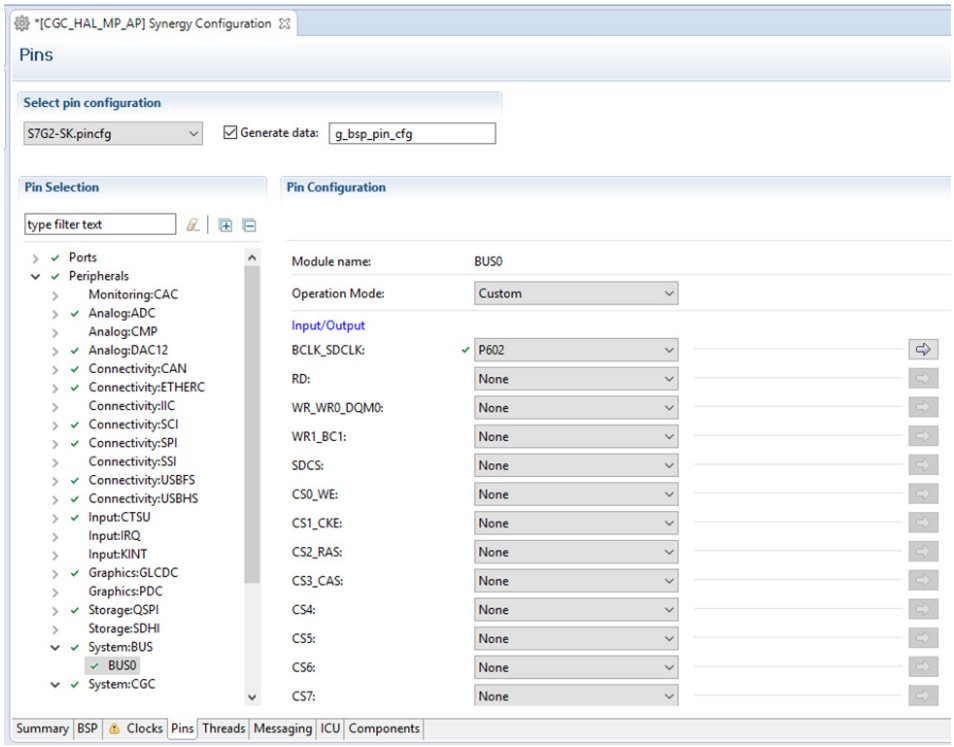


図 168:[Pins] タブでの SDCLK および BCLK の有効化 / 無効化

この例では、SDRAM クロックと BUS クロックが P602 で有効にされています。

CGC に関連付けられている他のピン設定により、外部発振器のピンを有効または無効にしたり、システムクロックの出力ピンを設定したりできます。

Synergy デバイスはオンチップ発振器を使用して動作することができ、その場合は、メインクロックの外部発振器のピン XTAL および EXTAL に対する要件はありません。アプリケーションでは、これらを入力ピンとして使用できます。サブクロック外部発振器のピン XCIN および XCOUT の機能は固定です。

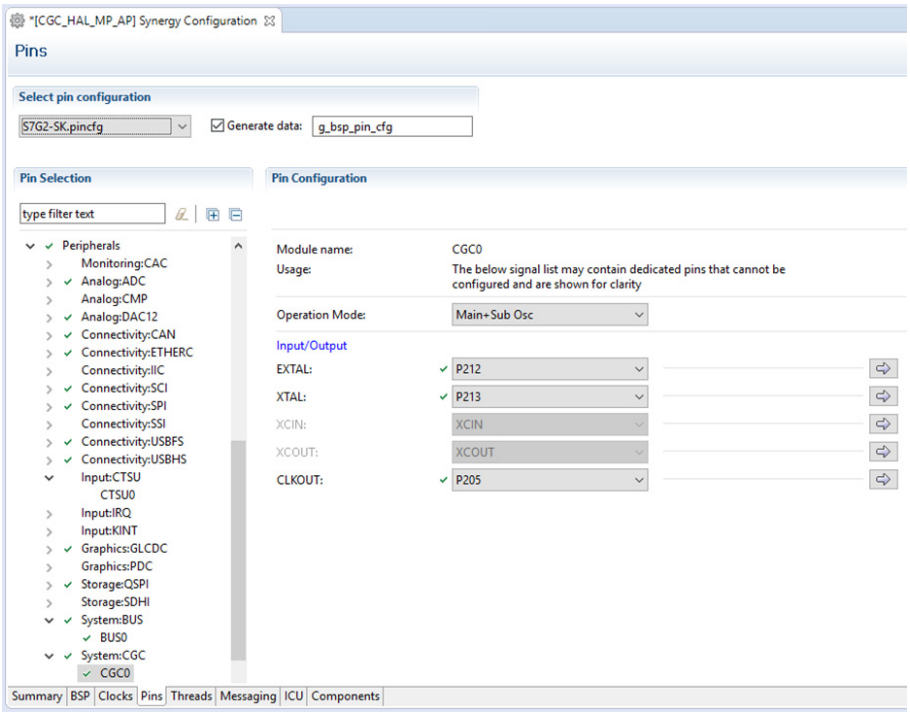


図 169:[Pins] タブでの EXTAL、XTAL、CLKOUT の有効化 / 無効化

この例では、外部主発振器がピン P212 と P213 を介して使用され、CLKOUT が P205 で有効にされています。

注 : 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと Synergy MCU では、使用可能なピン構成設定が異なる場合があります。

4.2.7.6 アプリケーションでの CGC モジュールの使用

アプリケーションで CGC HAL モジュールを使用する一般的な手順は次のとおりです。

1) CGC はシステムのリセット後に自動的に設定されます。
2) 必要に応じて、clocksCfg API を使用してクロックを構成します。
3) 必要に応じて、clockStart API を使用してクロックを開始します。
4) 必要に応じて、clockStop API を使用してクロックを停止します。
5) 必要に応じて、他の CGC 関数を呼び出します。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

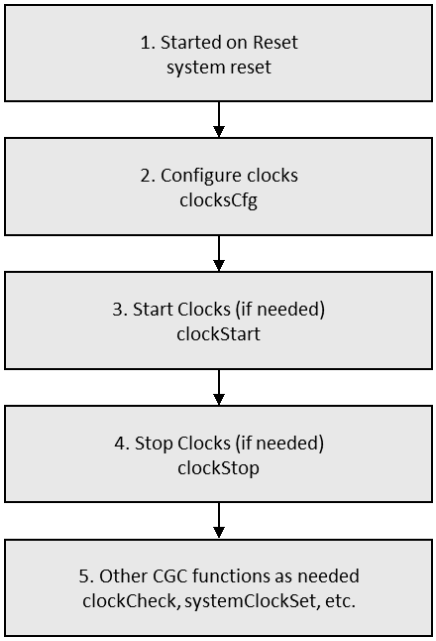


図 170:一般的な CGC HAL モジュールアプリケーションのフロー図

4.2.8 CRC ドライバー

CRC HAL モジュールは、メモリ内のデータブロック上またはシリアル通信インタフェース（SCI）チャネルを介したデータストリーム上の 8、16、および 32 ビットの CRC 値を、業界標準の多項式を利用して計算するためのハイレベルの API を提供します。

4.2.8.1 CRC HAL モジュールの特長

- CRC HAL モジュールは、メモリ内のデータブロックに対して CRC 計算を実行できます。
- CRC HAL モジュールは、シリアル通信インタフェース（SCI）チャネル上で送受信されるデータストリーム上の CRC を計算できます（スヌープモード）。
- CRC HAL モジュールは、以下の 8 および 16 ビット CRC 多項式をサポートします。これらの多項式は並行して 8 ビットデータに使用できます
  - $X^8 + *X^{*2} + *X^* + 1$ (CRC-8)
  - $X^{16} + *X^{*15} + *X^{*2} + 1$ (CRC-16)
  - $X^{16} + *X^{*12} + *X^{*5} + 1$ (CRC-CCITT)
- CRC HAL モジュールは、以下の 32 ビット CRC 多項式をサポートします。これらの多項式は並行して 32 ビットデータに使用できます
  - $X^{32} + *X^{*26} + *X^{*23} + *X^{*22} + *X^{*16} + *X^{*12} + *X^{*11} + *X^{*10} + *X^{*8} + *X^{*7} + *X^{*5} + *X^{*4} + *X^{*2} + *X^* + 1$ (CRC-32)

– $X^{32} + X^{28} + X^{27} + X^{26} + X^{25} + X^{23} + X^{22} + X^{20} + X^{19} + X^{18} + X^{14} + X^{13} + X^{11} + X^{10} + X^{9} + X^{8} + X^{6} + 1$ (CRC-32C)

- CRC HAL モジュールは、LSB ファーストまたは MSB ファーストのビット順で CRC を計算できます。

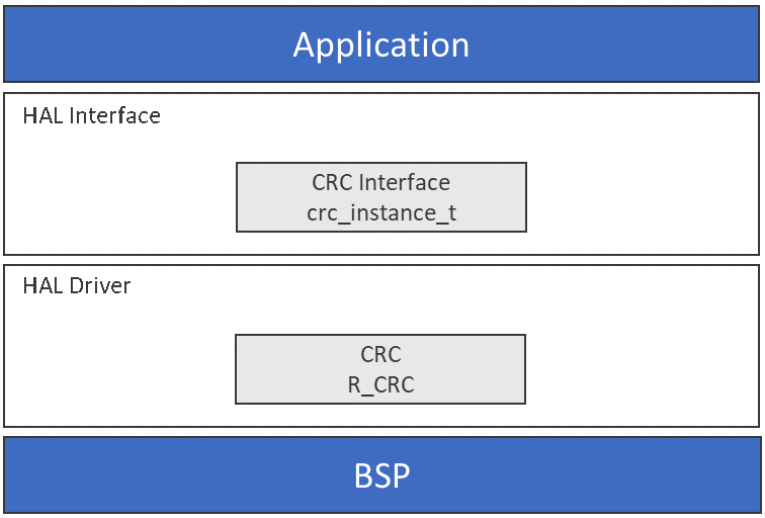


図 171:CRC HAL モジュールのブロック図

4.2.8.2 CRC HAL モジュールの API の概要

CRC HAL モジュールでは、オープン、クローズ、有効化、計算のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

CRC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_crc.p_api->open(g_crc.p_ctrl, g_crc.p_cfg); Open the CRC driver module.
close	g_crc.p_api->close(g_crc.p_ctrl); Close the CRC module driver.

Function Name	Example API Call and Description
crcResultGet	g_crc.p_api->crcResultGet(g_crc.p_ctrl, &result); Return the current calculated value.
snoopEnable	g_crc.p_api->snoopEnable(g_crc.p_ctrl, seed); Enable snooping.
snoopDisable	g_crc.p_api->snoopDisable(g_crc.p_ctrl); Disable snooping.
snoopCfg	g_crc.p_api->snoopCfg(g_crc.p_ctrl, g_crc.p_cfg); Configure the snoop channel and direction.
calculate	g_crc.p_api->calculate(g_crc.p_ctrl, &input_buffer, num_bytes, crc_seed, &crc_result); Perform a CRC calculation on a block of data.
versionGet	g_crc.p_api->versionGet(&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Configuration was successful.
SSP_ERR_ASSERTION	Assertion error.
SSP_ERR_INVALID_ARGUMENT	Invalid argument error.
SSP_ERR_NOT_OPEN	The driver is not opened.
SSP_ERR_IN_USE	If driver is already open.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.8.3 CRC HAL モジュールの動作の概要

メモリ内のデータブロックの CRC 値の計算に CRC HAL モジュールを使用するときは、calculate API を使用できます。この API は、入力バッファポインタ、長さ、CRC シード値を入力として受け取り、計算された CRC 値を出力します。

シリアル通信インタフェース (SCI) チャネルで送受信されるデータストリームの CRC を計算するために CRC HAL モジュールを使用するときは (スヌープモード)、最初に、snoopCfg API をコールしてから snoopEnable API を呼び出して、モジュールをスヌープモードで構成する必要があります。要求した数のデータが SCI チャネルで送受信された後、crcResultGet API を使用して、モジュールから計算された CRC 値を取得できます。

CRC HAL モジュールの動作に関する重要な注意事項と制限事項

- CRC ブロックは割り込みを使用しません。
- CRC モジュールにはクロック構成はありません。
- CRC モジュールにはコールバックはありません。
- メモリ内のデータブロックの CRC 値の計算に 32 ビットの CRC 多項式を使用すると、データブロックはリトルエンディアンのバイト順序を使用して解釈されます。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.8.4 アプリケーションへの CRC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに CRC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

CRC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(CRC ドライバーのデフォルト名は g_crc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

CRC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
r_crc0 CRC Driver on r_crc	Threads	New Stack> Driver> Monitoring> CRC Driver on r_crc

次の図に示すように、r_crc の CRC ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

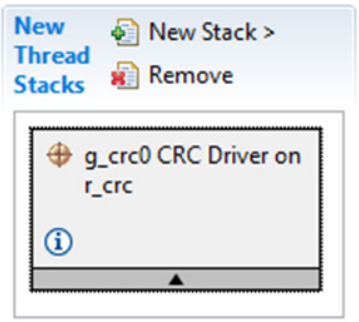


図 172:CRC HAL モジュールのスタック

4.2.8.5 CRC HAL モジュールの構成

ユーザーは必要な動作に合わせて CRC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_crc 上の CRC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g-crc0	Module name.
CRC Polynomial	CRC-8, CRC-16, CRC-CCITT, CRC-32, CRC-32C Default: CRC-32C	Specify the polynomial to use for calculation.
Bit Order	LSB, MSB Default: MSB	Specify the bit order of the calculation.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

CRC HAL モジュールのクロック構成

CRC HAL モジュールは周辺クロック A（PCLKA）からクロックを供給されます。

CRC HAL モジュールでは、API を使用して動作周波数を設定することはできません。

CRC HAL モジュールのピン構成

CRC HAL モジュールには構成可能なピンはありません。

4.2.8.6 アプリケーションでの CRC HAL モジュールの使用

CRC の実装には、大きく分けてノーマルモードとスヌープモードの 2 つのタイプがあります。各モードの一般的な手順を以下に示します。

アプリケーションで CRC HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、CRC HAL モジュールを初期化します。
2) calculate API を使用して、CRC HAL モジュールを計算します。
3) close API を使用して、CRC HAL モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

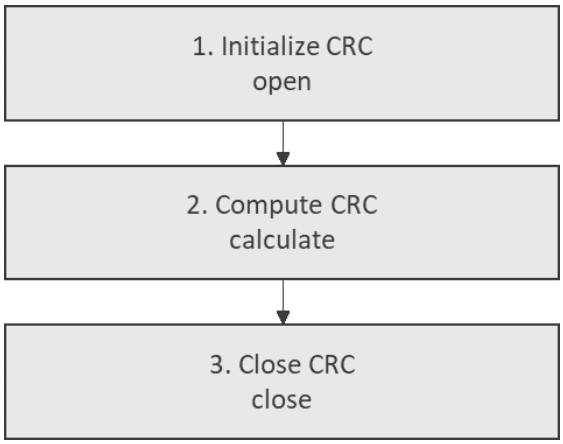


図 173:一般的な CRC HAL モジュールアプリケーションのフロー図

スヌープモードで計算に CRC HAL モジュールを使用する場合の一般的な手順は次のとおりです。

1) open API を使用して、CRC HAL モジュールを初期化します。
2) snoopCfg API を使用して、SCI チャネル（およびその方向）をスヌープするように CRC モジュールを構成します。
3) snoopEnable API を使用して、SCI チャネルのスヌーピングを有効にします。
4) SCI チャネルで必要なバイト数が送信または受信された後、crcResultGet API を使用して計算された CRC を取得します。
5) snoopDisable API を使用して、スヌーピング動作を無効化します。
6) close API を使用して、CRC HAL モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

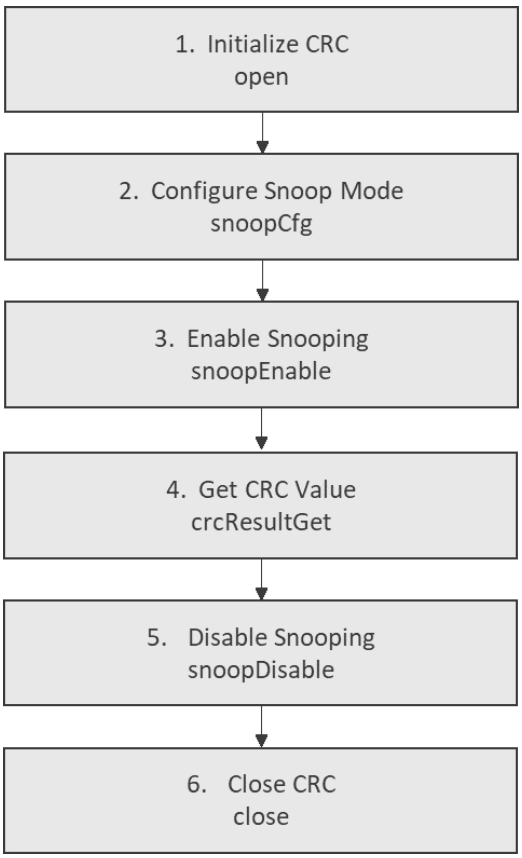


図 174:一般的な CRC HAL モジュールアプリケーションのフロー図

4.2.9 CTSU ドライバー

CTSU は、静電容量式タッチセンシングアプリケーション用のハイレベル API を提供し、Capacitive Touch Workbench ツールと連携して、ドライバーが初期化と動作のために使用する必要がある構造体を生成します。CTSU HAL モジュールは、Synergy MCU 上の CTSU ペリフェラルを使用します。コールバック関数が用意されており、スキャンが完了したときおよび新しい処理データが使用可能になったときに呼び出されます。

4.2.9.1 CTSU HAL モジュールの特長

CTSU HAL モジュールは、CTSU ペリフェラルデバイスを初期化して任意の設定済み（かつ有効化済み）チャネルのキャパシタンスの変化を検出し、必要なフィルタリングを実行し、ボタン、ホイール、スライダなどハイレベルのウィジェットレイヤーで使用できるさまざまなデータを生成するために使用されます。主な特長は次のとおりです。

- 相互容量とセルフキャパシタンスをサポートします。
- DTC 操作をサポートします。

- フィルタリング、ドリフト補正、自動調整（事前プログラムまたはユーザー定義）をサポートします。
- CTW for Synergy（Capacitive Touch Workbench for Renesas Synergy）ツールと連携して初期化データおよび動作データを作成します。

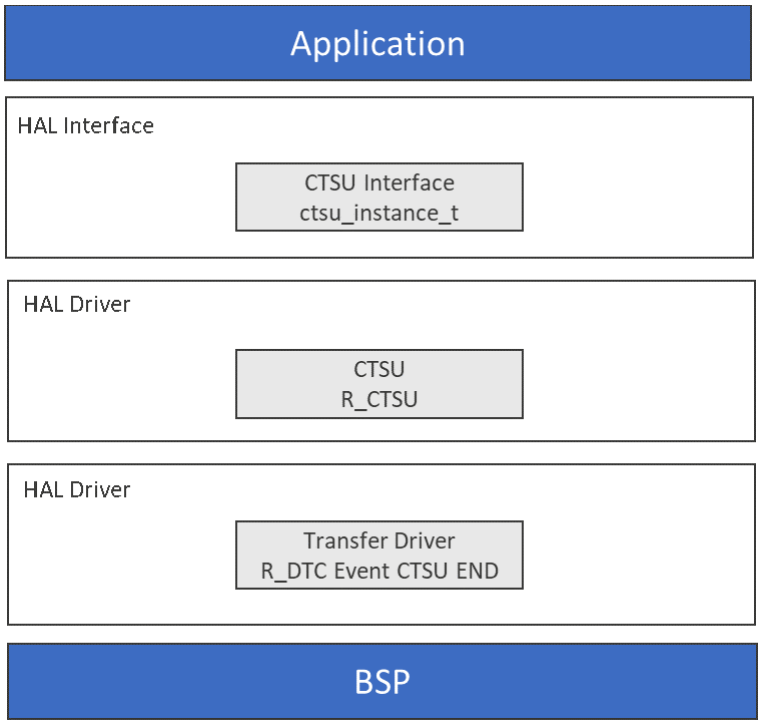


図 175:CTSU HAL モジュールのブロック図

4.2.9.2 CTSU HAL モジュールの API の概要

CTSU では、オープン、クローズ、開始、リード、スキャンのための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。戻りステータス値の表は API 要約表の後にあります。

CTSU HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_ctsu0.p_api->open(g_ctsu0.p_ctrl, g_ctsu0.p_cfg); Initialize the CTSU; enable power and clock and set the register configuration.

Function Name	Example API Call and Description
close	g_ctsu0.p_api->close(g_ctsu0.p_ctrl, options); Close the CTSU by ending any scan in progress, disabling interrupts, and removing power to the peripheral and saving configurations according to options.
scan	g_ctsu0.p_api->scan(g_ctsu0.p_ctrl); Start off a single CTSU scan.
update	g_ctsu0.p_api->update(g_ctsu0.p_ctrl); Update the CTSU internal data including the touch decision and other derived data according to the results of the scan.
read	g_ctsu0.p_api->read(g_ctsu0.p_ctrl, &destination, options, &channels, count); Read the results from the CTSU including raw data, binary data and other derived data according to the selected options.
versionGet	g_ctsu0.p_api->versionGet(&version); Retrieve the API version with the pointer version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_ASSERTION	The parameter p_ctrl or p_sample is NULL.

Name	Description
SSP_ERR_INVALID_POINTER	Invalid pointer.
SSP_ERR_INVALID_CMD	Invalid command.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_CTSU_OFFSET_ADJUSTMENT_FAILED	Adjustment failed.
SSP_ERR_IN_USE	Module in use.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.9.3 CTSU HAL モジュールの動作の概要

CTSU を使用しているアプリケーションで要求される種類のデータをサポートするため、この実装では上位レベルのレイヤーで多様な種類の処理済みデータを必要性に基づいて読み取れるよう、read 関数が提供されています。read API コールに引数として渡される ctsu_read_t 列挙には、さまざまなデータ型が列記されています。またドライバーには、各スキャンが完了したときおよび新しい処理データが使用可能になったときに呼び出されるコールバックも用意されています。このコールバックを使用して、上位レイヤーでデータを読み取り、処理することができます。

CTSU HAL モジュールを利用すると、相互容量やセルフキャパシタンスを含む、サポートされているすべての動作モードで CTSU チャネルを設定できます。ドライバーは設定済みチャネルのスキャン、DTC を使用したデータの移動、フィルタリングの実行、ドリフト補正、自動調整などを実行し、各繰り返し処理が完了するたびにコールバック関数によってユーザーに通知を行います。またこのドライバーにより、ユーザーは独自のフィルタリングや自動調整アルゴリズムを設定し、それをプロセスに統合することも可能になります。ドライバーでは一度に 1 つの構成しかサポートされませんが、アプリケーションの必要に応じて複数のチャネル設定でドライバーを再開することができます。

ドライバーを使用してさまざまなチャネルの「タッチされた」状態および「解放された」状態を検出するには、各チャネルの総静電容量に直接影響する操作環境に合わせてハードウェアを最初に「調整」する必要があります。この「調整用データ」は、ドライバーが実行時に既知の参照値として、また、「タッチされた」および「解放された」状態を決定するために使用します。

調整用データを生成するには、CTW for Synergy を使用します。このツールの使用と入手の詳細については、Renesas Synergy ウェブサイトにあるドキュメントを参照してください。

初期化中に open 関数がコールされると、ドライバーは自動調整を実行します。ドライバーに提供された調整データが「タッチされた / 解放された」状態の変化を検出するのに十分でない場合、この自動調整は失敗します。この原因としては、調整用データにエラーがあるか、チャネルの静電容量を含む現在の動作環境が、調整用データが生成された環境と著しく異なることが考えられます。

CTSU HAL モジュールの動作に関する重要な注意事項と制限事項

- CTSU ドライバーは、CTSU ペリフェラルを使用するすべての Synergy デバイスをサポートするように設計されています。
- ドライバーは [相互容量] モードおよび [セルフ キャパシタンス マルチスキャン] モードのみをサポートします。ドライバーでは動的メモリ割り当てモードがサポートされないため、使用されるチャネルの数が動的に変化するアプリケーションで使用されるメモリの量は引き続き、システムで初期化されたチャネルの総数によって決まります。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.9.4 アプリケーションへの CTSU HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに CTSU HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

CTSU ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(CTSU ドライバーのデフォルト名は g_ctsu0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

CTSU HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ctsu0 CTSU driver on r_ctsu	Threads	New Stack> Driver> Input> CTSU Driver on r_ctsu

次の図に示すように、r_ctsu の CTSU ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

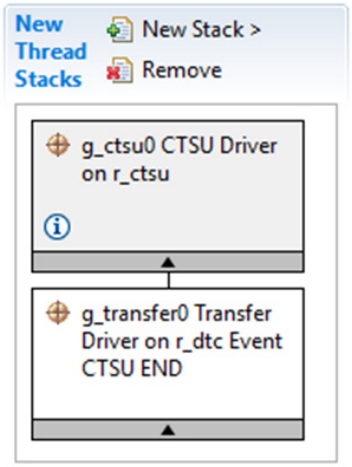


図 176:CTSU HAL モジュールのスタック

4.2.9.5 CTSU HAL モジュールの構成

ユーザーは必要な動作に合わせて CTSU HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_ctsu 上の CTSU HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking Enable	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.

ISDE Property	Value	Description
Offset Adjustment	Enabled, Disabled Default: Enabled	Used to enable or disable offset adjustment. Leave as enabled unless you are tuning the board. (Rate is determined by *Runtime rate of tuning of sensor values*.)
Drift Compensation	Enabled, Disabled Default: Enabled	Used to enable or disable drift compensation. Leave as enabled unless you are tuning the board. Drift compensation allows the baseline values used to determine the presence or absence of a touch to change over time as the board or surface ages or with changes in temperature and environment. Drift compensation rate is determined by *Steady state drift compensation rate*, *Startup drift compensation rate*, and *Channel release compensation rate*.
Drift Compensation Method (valid only if Drift Compensation is enabled above)	Alternate Method 1	Drift Compensation algorithms provided. Only Alternate 1 is currently supported. Other options may be supported in future releases. Drift compensation method Alternate 1 allows for different rates of drift compensation for the following events: steady state, startup, and channel release.
Steady state drift compensation rate, drift compensation will be applied per n scans	500	Determines the rate of drift compensation (applied every n scans) when the channel is in steady state. (Dependent on the scan rate.)

ISDE Property	Value	Description
Startup drift compensation rate (Should be less than the steady state drift compensation)	5	Determines the rate of drift compensation (applied every n scans) when the driver is performing its initialization. (Should be less than the *Steady state drift compensation rate*.)
Channel release compensation rate (Should be less than the steady state drift compensation rate)	500	Determines the rate of drift compensation (applied every n scans) when a channel transitions from pressed to released. (Should be less than or equal to the *Steady state drift compensation rate*.)
Default filter depth (used in sensor count filter provided by driver)	1	Used in the software sensor count filter provided by driver. This default filter is a modification of the relatively common "leaky integrator" software filter. A depth of 4 equates to an approximate filter constant of 16. When the input is a constant value of 16 or greater, the filter output will reach 68-69% of the constant input value after 16 cycles/iterations of the filter.
Runtime rate of tuning of sensor values (if drift compensation is used this value is overrriden to be twice the rate of the steady state drift compensation)	800	Rate of offset adjustment. (If drift compensation is used this value is set to twice the rate of the steady state drift compensation.)
Perform auto-tune and drift compensation only when all channels are untouched	True, False Default: True	Perform auto-tune and drift compensation only when all channels are untouched.

ISDE Property	Value	Description
Max. active channels	1	Specify the maximum number of channels that are to be used by the application. In Self Capacitance Mode, this is the number of channels used. In Mutual Capicitace Mode, this is the multiplication result of the Rx and Tx channels. For example: 4 Rx and 3 Tx channels = 12 Maximum Active Channels.
Name	g_ctsu0	Module name.
CTSU configuration used	g_ctsu0_config	Name of the configuration structure generated by CTW.
Callback	NULL	The user callback function that will be invoked each time the scan is complete as well as when newly processed data is available. Can be set to NULL if not used.
Data Processing Option	Default Processing (recommended), No Processing (tuning only) Default: Default Processing	Can be used to specify if scans should start after the data from the previous one is completed, if auto-calibration should be run between scans etc. Use the Default Processing unless you are tuning.
Write Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Write interrupt priority selection.
Read Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Read interrupt priority selection.

ISDE Property	Value	Description
End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	End interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

CTSU HAL モジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_dtc Event CTSU END 時の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Enabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Module name
Mode	Block	Mode selection
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection

ISDE Property	Value	Description
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	1	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	1	Number of blocks selection
Activation Source (Must enable IRQ)	Event CTSU END	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

CTSU HAL モジュールのクロック構成

クロック速度を、静電容量式タッチ調整ツール Workbench 6 で選択したクロック速度と同じ値に設定します。

CTSU HAL モジュールのピン構成

CTSU ペリフェラルモジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は CTSU ピンの選択例を示しています。

r_ctsu 上の CTSU HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
SCI	Pins	Select Peripherals > CTSU> Cap_Touch_Sensing_Unit

注 : この選択シーケンスでは、SCI1 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_ctsu 上の CTSU HAL モジュールのピン構成設定

Property	Value	Description
Operation Mode	Disabled, Enabled Default: Disabled	Select Enabled.
TS0:17	None, Pn Default: None	TSD Pin selection.
TSCAP	None, P203, P205 Default: None	TS Capacitance Pin selection.

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

4.2.9.6 アプリケーションでの CTSU HAL モジュールの使用

CTSU は、Workbench プログラムを ISDE と組み合わせて使用し、モジュールの初期化に必要なコードを生成します。確認する必要がある重要な手順は次のとおりです。

1) ISDE で、CSTU HAL モジュール、構成ピン、使用する総チャネル数を追加し、割り込み（同じ優先順位に対する）を選択し、アプリケーションが使用するコールバック関数を作成します。
2) CTW for Synergy によって生成された CTSU 構成を含むフォルダが ${PROJECT_ROOT}\src フォルダに存在し、ビルドに含まれていることを確認します。
3) [CTSU Configuration Used] フィールドの名前が、CTW for Synergy によって生成された構成の名前に設定されていることを確認します。
4) [Generate Project Content] をクリックして、必要な構造体とドライバーコードを生成します。

CTSU コードが正常に生成された後、アプリケーションで CTSU を使用するための一般的な手順は次のとおりです。

1) open API を使用して、モジュールを初期化します。
2) メインのループから周期的に scan API をコールし続け、新しいスキャンを開始します。このタスクはタイマを使用して定期的に実行すると便利です。それにより、スキャン レートが決定されます。
3) 各スキャンの終了時には、ユーザー コールバックが呼び出されます。コールバックでは、コールバックのイベントを確認し、update API をコールしてデータに対する内部処理を実行し、いずれかのチャネルでタッチ / 解放イベントが発生したかどうかを確認します。処理が完了すると、ユーザーコールバックが再度、データが処理されたことを示す異なる引数で呼び出されます。
4) read API を使用して、最新のスキャンから未加工または処理済みのデータを読み取ります。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

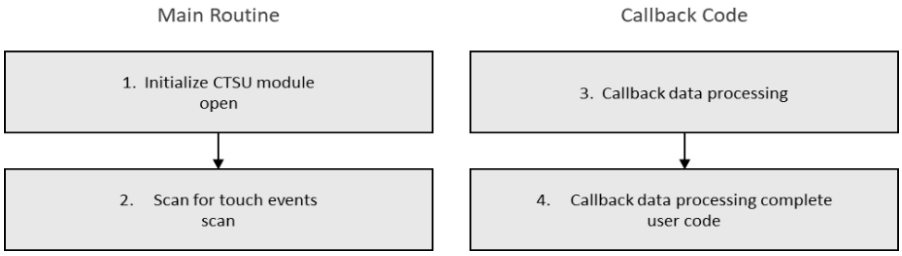


図 177:一般的な CTSU HAL モジュールアプリケーションのフロー図

4.2.10 DAC ドライバー

DAC HAL モジュールは、デジタル / アナログ変換アプリケーション用のハイレベル API を提供し、Synergy MCU 上のデュアルチャネル 12 ビット D/A コンバータ（DAC12）ペリフェラルをサポートします。

4.2.10.1 DAC HAL モジュールの特長

このモジュールは、デュアルチャネル 12 ビット D/A コンバータ（DAC12）を、正および負の基準電圧の間の 4096 段階の電圧レベルのいずれかを出力するように構成します。このモジュールには以下の構成設定が含まれます。

- 16 ビット入力データレジスタに対する左詰めまたは右詰めの 12 ビット値フォーマットを設定します
- 出力アンプを有効または無効にします
- アナログ / デジタルコンバータ（ADC）モジュールを使用して、同期干渉防止モードで動作します。

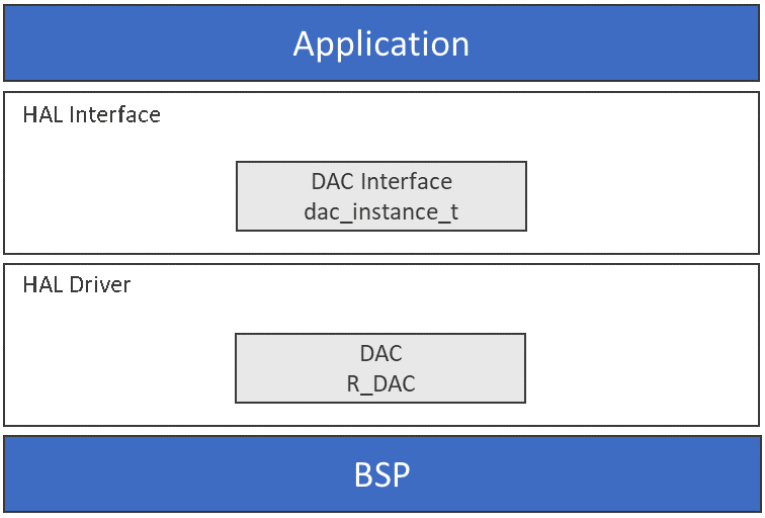


図 178:DAC HAL モジュールのブロック図

4.2.10.2 DAC HAL モジュールの API の概要

DAC HAL モジュールでは、オープン、クローズ、開始、停止、および DAC へのライトを行うための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

DAC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_dac.p_api->open(g_dac.p_ctrl, g_dac.p_cfg) Initial Configuration.
close	g_dac.p_api->close(g_dac.p_ctrl) Close the D/A Converter.
write	g_dac.p_api->write(g_dac.p_ctrl, val) Write Sample value to the D/A Converter.

Function Name	Example API Call and Description
start	g_dac.p_api->start(g_dac.p_ctrl) Start the D/A Converter if it has not been started yet.
stop	g_dac.p_api->stop(g_dac.p_ctrl) Stop the D/A Converter if the converter is running.
versionGet	g_dac.p_api->versionGet(&version) Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_NOT_OPEN	Unit is not open.
SSP_ERR_ASSERTION	Wrong parameter.
SSP_ERR_IN_USE	DAC resource is locked.
SSP_ERR_NOT_OPEN	The peripheral is not opened.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.10.3 DAC HAL モジュールの動作の概要

DAC HAL モジュールは、デュアルチャネル 12 ビット D/A コンバータ（DAC12）を、正および負の基準電圧の間の 4096 段階の電圧レベルのいずれかを出力するように構成します。このモジュールを使用して、16 ビット入力データレジスタに対する 12 ビット出力の左詰めフォーマットまたは右詰めフォーマットを構成できます。DAC HAL モジュールは、出力アンプを有効または無効化したり、ADC HAL モジュールを使用して同期干渉防止モードで動作したりすることもできます。

DAC HAL モジュールの動作に関する重要な注意事項と制限事項

DAC HAL モジュールでは、以下の初期化手順が必要です。

- DAC モジュールストップビットをゼロにクリアします。
- DAC チャネル出力有効化を 1 に設定します。

DAC モジュールストップビットは、ドライバーのインスタンスカウンタがゼロの場合、open を呼び出した時点でゼロにクリアされます。ドライバーのインスタンスカウンタは、ゼロに初期化された後、チャネルの open が正常に戻るとインクリメントされ、チャネルの close が呼び出されるとデクリメントされます。DAC モジュールストップビットが1に設定されるのは、ドライバーのインスタンスカウンタがcloseの呼び出しで0にデクリメントされたときです。

DAC チャネル出力有効は、open が正常に呼び出された後で初めてチャネルの write が呼び出された時点で、1 に設定されます。open の呼び出しでは、dac_ctrl_t 構造体の要素 channel_started. に 0 が書き込まれます。channel_started が 0 にクリアされた状態で write が呼び出されると、そのチャネルの DAC 出力有効ビットは 1 に設定されます。チャネルの DAC 出力有効は、close および stop が呼び出されると 0 にクリアされます。

- DAC HAL モジュールには、ピン構成は実装されていません。現在、DA0 と DA1 の出力は、ピン構成コントロールレジスタの ASEL フィールドのリセット値によって有効になります。
- DAC HAL モジュールの電圧基準の選択は実装されていません。現在、D/A VREF コントロールレジスタ（DAVREFCR）のリセット値によって基準は選択されませんが、これは正しい状態です。
- 変換をトリガする DAC 入力イベントの構成は、現在実装されていません。コントロールレジスタの DAE ビットのデフォルトのリセット値（0）では、各チャネルを個別にトリガできます。
- DAC HAL モジュール変換の同期のためのイベント信号入力は、現在実装されていません。
- このモジュールの動作に関する制限事項については、最新の SSP リリースノートを参照してください。

4.2.10.4 アプリケーションへの DAC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに DAC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

DAC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（DAC ドライバーのデフォルト名は g_dac0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

DAC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_dac0 DAC Driver on r_dac	Threads	New Stack > Driver > Analog > DAC Driver on r_dac

次の図に示すように、r_dac の DAC ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調

表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

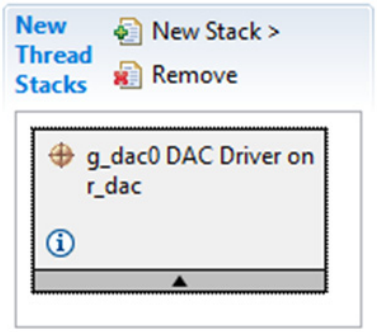


図 179:DAC HAL モジュールのスタック

4.2.10.5 DAC HAL モジュールの構成

ユーザーは必要な動作に合わせて DAC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_dac 上の DAC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_dac0	Module name.

ISDE Property	Value	Description
Channel	0	Set to 0 for output DA0 or 1 for output DA1.
Synchronize with ADC	Enabled, Disabled Default: Disabled	Set to true for anti-interference synchronization with the Analog-to-Digital Converter (ADC) Module. Set to false false if power supply interference between the analog modules is not a problem, or if asynchronous conversion by the DAC Module is desired.
Data Format	Right Justified, Left Justified Default: Right Justified	Set to zero, if 12-bit data values are loaded in bits 11 through 0, or right justified. Set to one, if 12-bit data values are loaded in bits 15 through 4, or left justified.
Output Amplifier	Enable, Disable Default: Disable	Set to true, if output amplifier hardware function is desired. Set to false to bypass output amplifier hardware function.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

DAC HAL モジュールのクロック構成

DAC HAL モジュールには、固有のクロック構成は必要ありません。

DAC HAL モジュールのピン構成

DAC HAL モジュールを使用するには、アナログ入力を受信するチャネルのポートピンを、ピンコンフィギュレータでアナログピンとして設定する必要があります。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では DAC ピンの構成設定を示します。

r_dac 上の DAC HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
DAC	Pins	Select Peripherals > Analog: DAC12 > DAC120

r_dac 上の DAC HAL モジュールのピン構成設定

Property	Value	Description
Module Name	DAC120	DAC Peripheral Module.
Operation Mode	Enabled, Disabled Default: Enabled	DAC Peripheral operation mode.
DA0	None, DA0 Default: None	DAC Output Pin.
DA1	None, DA1 Default: None	DAC Output Pin.

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

4.2.10.6 アプリケーションでの DAC HAL モジュールの使用

アプリケーションでの DAC HAL モジュールの一般的な使用手順は次のとおりです。

1) open API を使用して、DAC HAL モジュールを初期化します。
2) write API を使用して、データ値を書き込みます。
3) start API を使用して、データのライトを開始します。
4) 必要に応じて、write API を使用してデータ値のライトを続けます。
5) stop API を使用して、データのライトを停止します。
6) close API を使用して、DAC HAL モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

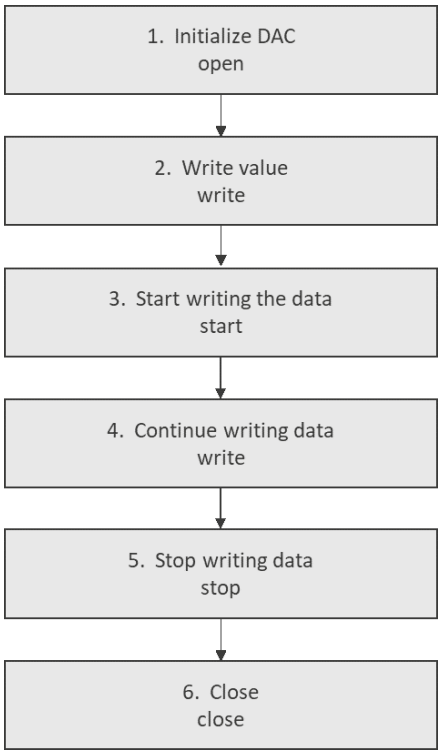


図 180:一般的な DAC HAL モジュールアプリケーションのフロー図

4.2.11 DAC8 ドライバー

DAC8 HAL モジュールは、デジタル / アナログ変換アプリケーション用のハイレベル API を提供し、Synergy MCU 上の 8 ビット D/A コンバータ（DAC8）ペリフェラルをサポートします。

4.2.11.1 DAC8 HAL モジュールの特長

- 8 ビット D/A コンバータ
- 左詰めまたは右詰めの入力データフォーマット
- アナログ / デジタルコンバータ（ADC）モジュールとの同期
- 複数の動作モード
  - ノーマル
  - リアルタイム（イベントリンク）
- チャージポンプコントロール

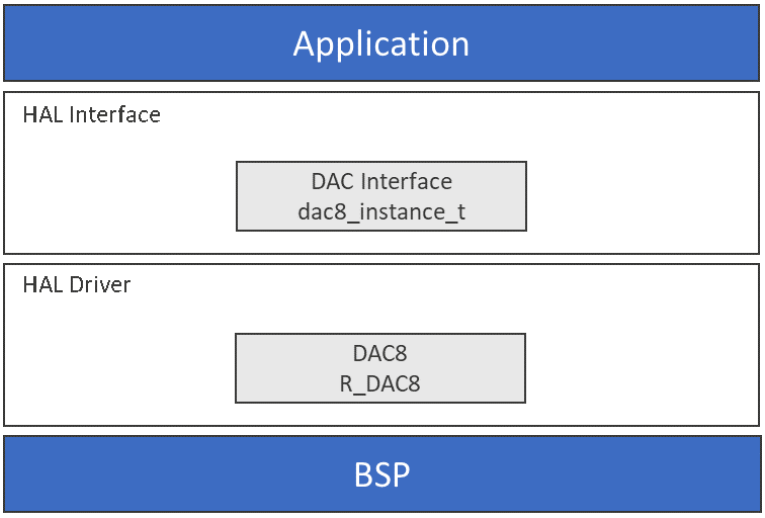


図 181:DAC8 HAL モジュールのブロック図

4.2.11.2 DAC8 HAL モジュールの API の概要

DAC8 HAL モジュールでは、オープン、クローズ、開始、停止、DAC へのライトのための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

DAC8 HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_dac8.p_api->open(g_dac8.p_ctrl, g_dac8.p_cfg Initial Configuration.
close	g_dac8.p_api->close(g_dac8.p_ctrl) Close the D/A Converter.
write	g_dac8.p_api->write(g_dac8.p_ctrl, val) Write Sample value to the D/A Converter.

Function Name	Example API Call and Description
start	g_dac8.p_api->start(g_dac8.p_ctrl) Start the D/A Converter if it has not been started yet.
stop	g_dac8.p_api->stop(g_dac8.p_ctrl) Stop the D/A Converter if the converter is running.
versionGet	g_dac8.p_api->versionGet(&version) Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_HW_LOCKED	DAC resource is locked.
SSP_ERR_NOT_OPEN	Unit is not open.
SSP_ERR_ASSERTION	Wrong parameter.
SSP_ERR_IP_CHANNEL_NOT_PRESENT	Wrong channel selected.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.11.3 DAC8 HAL モジュールの動作の概要

DAC8 HAL モジュールは、8 ビット D/A コンバータ（DAC8）を、正および負の基準電圧の間の 256 段階の電圧レベルのいずれかを出力するように構成します。16 ビット入力データの左詰めまたは右詰めフォーマットで 8 ビット出力データを受け付けるように、ドライバーを構成できます。このドライバーは、2 モードの DAC をサポートします。

- 通常モード – D/A 出力は、データレジスタへのライト時に更新されます。

- リアルタイム（イベントリンク） - D/A 出力は、イベントリンクのイベント時に更新されます。このモードの間は、いつでもデータレジスタに書き込むことができます。イベントリンクイベントは変換の開始をトリガします。詳細については、『SSP ユーザーズ マニュアル』の「ELC インタフェース」を参照してください。

ADC のリードでのノイズを減らすには、ドライバーで ADC モジュールを使用して同期干渉防止モードを構成できます。このモードは、ADC のサンプリング中に DAC チャージを無効にすることによって変換ノイズを減らします。この機能がサポートされているかどうかは、ハードウェアのマニュアルで確認してください。

低 AVCC 電圧での動作では、ドライバーはハードウェアチャージポンプを有効または無効にできます。

リアルタイムモード

リアルタイムモードでは、出力電圧は ELC ペリフェラルからの信号によってのみ変化します。リアルタイムモードを使用するときは、最初の write コールで初期出力電圧が設定されることに注意してください。

たとえば、次のコードは、チャネル 0 での DAC8 出力を ELC ソフトウェアイベントにリンクし、出力の変更をトリガする方法を示しています。

```
/* Open DAC8 driver. */
g_dac8_0.p_api->open(g_dac8_0.p_ctrl, &g_dac8_0.p_cfg);

/* Link DAC8 output 0 to software event 0.

   In a real application, this feature would be used to link DAC8 output

   to a hardware event. A software event is used here for simplicity. */

g_elc_on_elc.linkSet(ELC_PERIPHERAL_DA80, ELC_EVENT_ELC_SOFTWARE_EVENT_0);

/* Set initial output voltage, the output is immediately updated
   because this is the first write since open. */

g_dac8_0.p_api->write(g_dac8_0.p_ctrl, initial_dac_value);

while (true) {

    /* Set the next output value, the output voltage is not updated until

       an ELC event occurs. */

    g_dac8_0.p_api->write(g_dac8_0.p_ctrl, next_dac_value);

    /* Generate software ELC event to update DAC output. */

    g_elc_on_elc.softwareEventGenerate(ELC_SOFTWARE_EVENT_0);

    R_BSP_SoftwareDelay(10, BSP_DELAY_UNITS_MILLISECONDS);

}
```

DAC8 HAL モジュールの動作に関する重要な注意事項と制限事項

- DAC8 チャネルの出力は、start から write の間は有効になり、stop から close の間は無効になります。

- DAC8 ドライバーでは、リアルタイムモード用の ELC ペリフェラルは構成されません。ユーザーは、DAC8 モジュールでリアルタイムモードを有効にするだけでなく、イベントリンクコントローラを構成する必要があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.11.4 アプリケーションへの DAC8 HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに DAC8 HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

DAC8 ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(DAC8 ドライバーのデフォルト名は g_dac8_0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

DAC8 HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_dac8_0 DAC Driver on r_dac8	Threads	New Stack > Driver > Analog > DAC Driver on r_dac8

次の図に示すように、r_dac8 の DAC8 ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

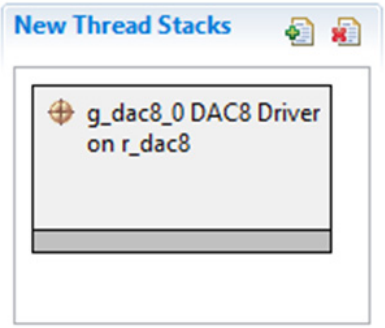


図 182:**DAC8 HAL モジュールのスタック**

4.2.11.5 DAC8 HAL モジュールの構成

ユーザーは必要な動作に合わせて DAC8 HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_dac8 上の DAC8 HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_dac8_0	Module name
Channel	0	Channel selection
Synchronize with ADC	Enabled, Disabled Default: Disabled	Choose whether to sync with the ADC module

ISDE Property	Value	Description
Data Format	Right Justified, Left Justified Default: Right Justified	Data format selection
DAC Mode	Normal Mode, Real-time (Event Link) Mode Default: Normal Mode	DAC mode selection
Charge Pump Enabled (Requires MOCO active)	Enabled, Disabled Default: Enabled	Enable or disable the charge pump

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

DAC8 HAL モジュールのクロック構成

DAC8 HAL モジュールには、特定のクロック構成は必要ありません。

DAC8 HAL モジュールのピン構成

DAC8 HAL モジュールを使用するには、アナログ入力を受信するチャネルのポートピンを、ピンコンフィギュレータでアナログピンとして設定する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は DAC ピンの選択例を示しています。

r_dac8 上の DAC8 HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
DAC8	Pins	Select Peripherals > Analog:DAC8 > DAC80

注 : この選択シーケンスでは、DAC80 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_dac8 上の DAC8 HAL モジュールのピン構成設定

Property	Value	Description
Module Name	DAC80	DAC Peripheral Module.

Property	Value	Description
Operation Mode	Enabled, Disabled Default: Enabled	DAC Peripheral operation mode.
DA0	None, DA0 Default: None	DAC Output Pin.
DA1	None, DA1 Default: None	DAC Output Pin.

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

4.2.11.6 アプリケーションでの DAC8 HAL モジュールの使用

アプリケーションで DAC8 HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、DAC8 HAL モジュールを初期化します。
2) write API を使用して、値を書き込みます。
3) start API を使用して、変換を開始します。
4) 必要に応じて、write API を使用して値のライトを続けます。
5) stop API を使用して、変換を停止します。
6) close へのコールを使用して、ペリフェラルの電源をオフにします。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

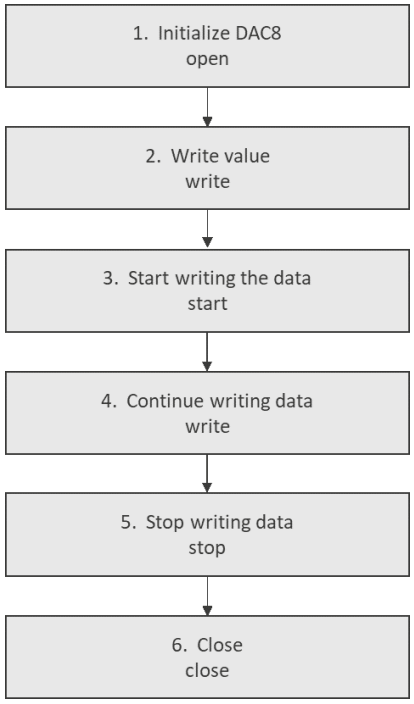


図 183:一般的な DAC8 HAL モジュールアプリケーションのフロー図

4.2.12 ディスプレイドライバー

4.2.12.1 GLCDC HAL モジュール

GLCDC HAL モジュールは、Synergy MCU の GLCDC 周辺機能を使用し、グラフィック表示アプリケーション向けのハイレベル API をサポートします。フレームバッファの切り替えやアンダーフロー検出処理のためにユーザー定義したコールバックを生成できます。

GLCDC HAL モジュールの特長

GLCDC HAL は次の機能をサポートします。

- RGB インタフェース（最大 24 ビット）と同期信号（HSYNC、VSYNC、およびデータ有効（オプション））で LCD パネルをサポートします
- 入力グラフィックプレーン用の各種の色フォーマットをサポートします（RGB888、ARGB888、RGB565、ARGB1555、ARGB4444、CLUT8、CLUT4、CLUT1）
- 512 ワード（32 ビット / ワード）の入力グラフィックプレーンのためのカラールックアップテーブル（CLUT）の使用をサポートします
- 出力用の各種の色フォーマットをサポートします（RGB888、RGB666、RGB565、シリアル RGB）
- 背景プレーン上に 2 つのグラフィックプレーンを入力し、画面上でブレンドできます
- パネルにドットクロックを生成します。クロックソースは内部または外部 (LCD_EXTCLK) から選択できます

- 明るさ調整、コントラスト調整、ガンマ補正をサポートします
- フレームバッファの切り替えまたはアンダーフロー検出を処理するための、GLCDC 割り込みをサポートします

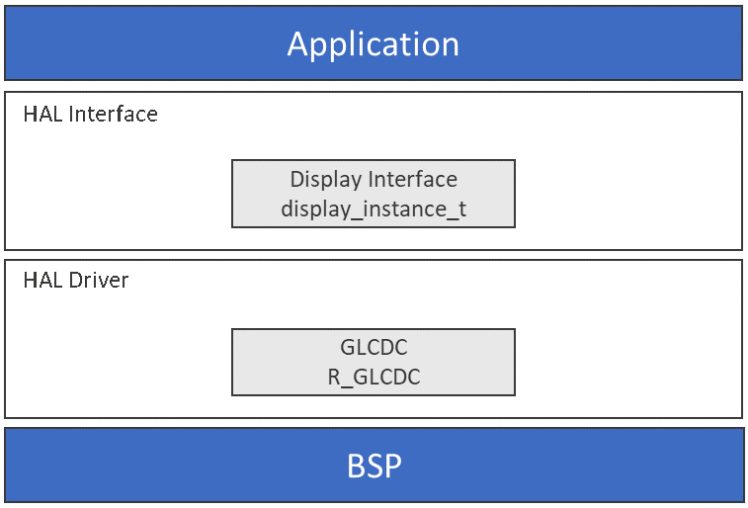


図 184:GLCDC HAL モジュールのブロック図

4.2.12.2 GLCDC HAL モジュールの API の概要

GLCDC HAL モジュールでは、オープン、クローズ、開始、停止、および LCD パネルへの情報表示の制御のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

GLCDC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_display.p_api->open (g_display.p_ctrl, g_display.p_cfg); Open display device.
close	g_display.p_api->close (g_display.p_ctrl); Close display device.

Function Name	Example API Call and Description
start	g_display.p_api->start (g_display.p_ctrl); Display start.
stop	g_display.p_api->stop (g_display.p_ctrl); Display stop.
layerChange	g_display.p_api->layerChange(g_display.p_ctrl, &layercng) Change layer parameters at runtime.
correction	g_display.p_api->correction(g_display.p_ctrl, &display_correction) Color correction.
clut	g_display.p_api->clut(g_display.p_ctrl, &clut) Set CLUT for display device.
statusGet	g_display.p_api->statusGet(g_display.p_ctrl, &status) Get status for display device.
versionGet	g_display.p_api->versionGet(&version) Retrieve the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API call successful.
SSP_ERR_ASSERTION	Parameter has invalid value.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter in the argument.
SSP_ERR_HW_LOCKED	GLCDCC resource is locked.
SSP_ERR_CLOCK_GENERATION	Dot clock cannot be generated from clock source.
SSP_ERR_INVALID_TIMING_SETTING	Invalid panel timing parameter.
SSP_ERR_INVALID_LAYER_SETTING	Invalid layer setting found.
SSP_ERR_INVALID_LAYER_FORMAT	Invalid format is specified.
SSP_ERR_INVALID_GAMMA_SETTING	Invalid gamma correction setting found.
SSP_ERR_NOT_OPEN	The function call is performed when the driver state is not equal to DISPLAY_STATE_CLOSED.
SSP_ERR_INVALID_UPDATE_TIMING	A function call is performed when the GLCDC is updating register values internally.
SSP_ERR_INVALID_MODE	Function call is performed when the driver state is not DISPLAY_STATE_OPENED.
SSP_ERR_INVALID_CLUT_ACCESS	Illegal CLUT entry or size is specified.
p_version	The version number.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.12.3 GLCDC HAL モジュールの動作の概要

GLCDC HAL モジュールは LCD パネルを制御します。下図に、GLCDC HAL モジュールを使用したときのグラフィックデータフローの概要を示します。このモジュールは、メモリからのグラフィックフレーム画像データのリード（最大 2 フレーム）と、これら画像のモノクロ背景画面上でのブレンドをサポートしています。このドライバーは CLUT メモリをサポートしており、CLUT 用のグラフィックフレーム形式を指定します。

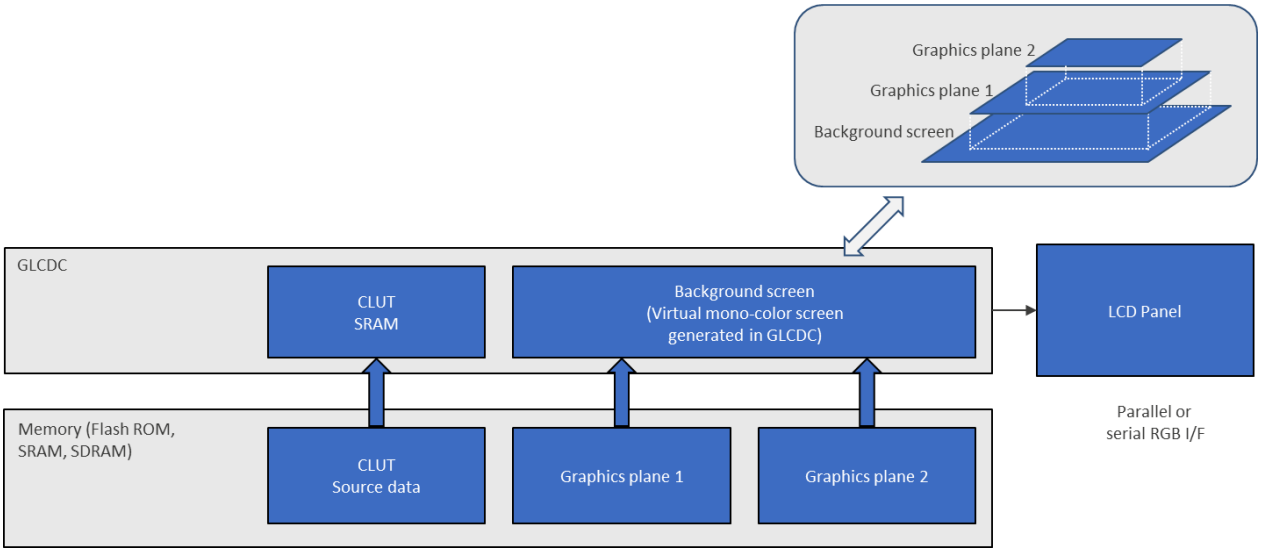


図 185:GLCDC HAL モジュールデータフロー

下の図は、ピンポンフレームバッファを備えた表示システムを示しています。単一フレームバッファ表示システムで発生する画像の切れ目を回避するため、表示システムでは 3 つ以上のフレームバッファを使用することをお勧めします。そのような設計では、GLCDC ハードウェアがグラフィックフレーム画像をいずれかのフレームバッファから読み込んでいる間に、画像描画エンジン（DRW や JPEG など）、CPU、または DMAC/DTC が同時にグラフィックフレーム画像を別のフレームバッファに転送することができます。このモジュールは、display_api_t::layerChange API による実行時のフレームバッファ切り替えをサポートしています。

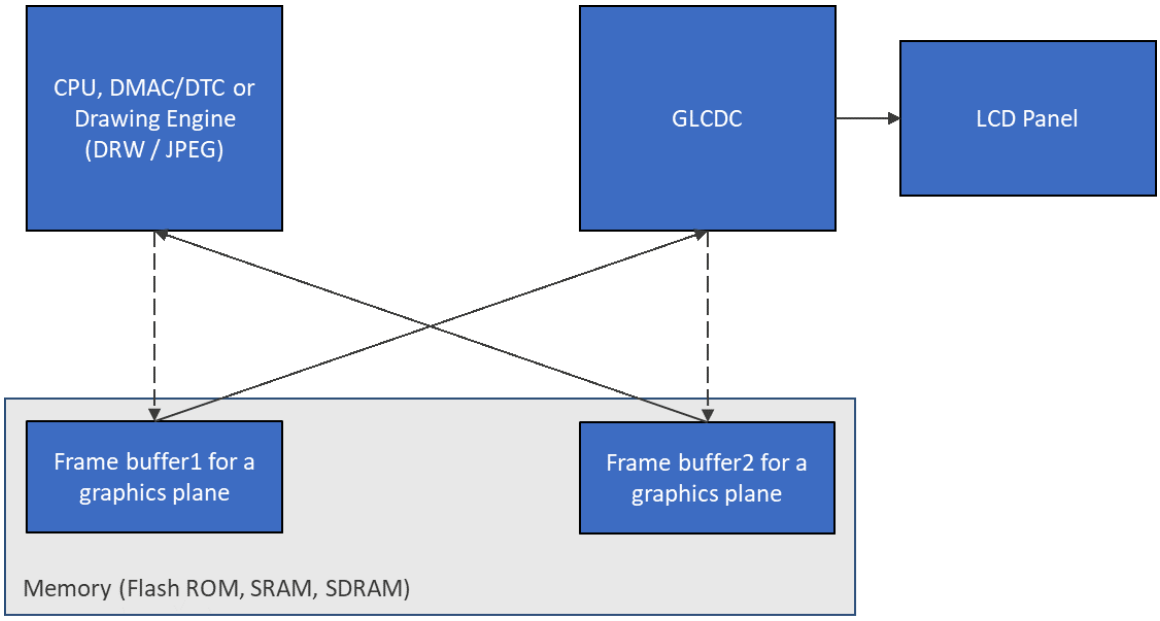


図 186:GLCDC HAL モジュールのピンポンバッファシステム

画面のフォーマット

次の図は、GLCDC モジュールの LCD 画面フォーマットと LCD タイミングパラメータの関係を示したものです。このモジュールは、さまざまなLCDパネルをサポートするLCDパネル設定用の汎用タイミングパラメータを備えています。

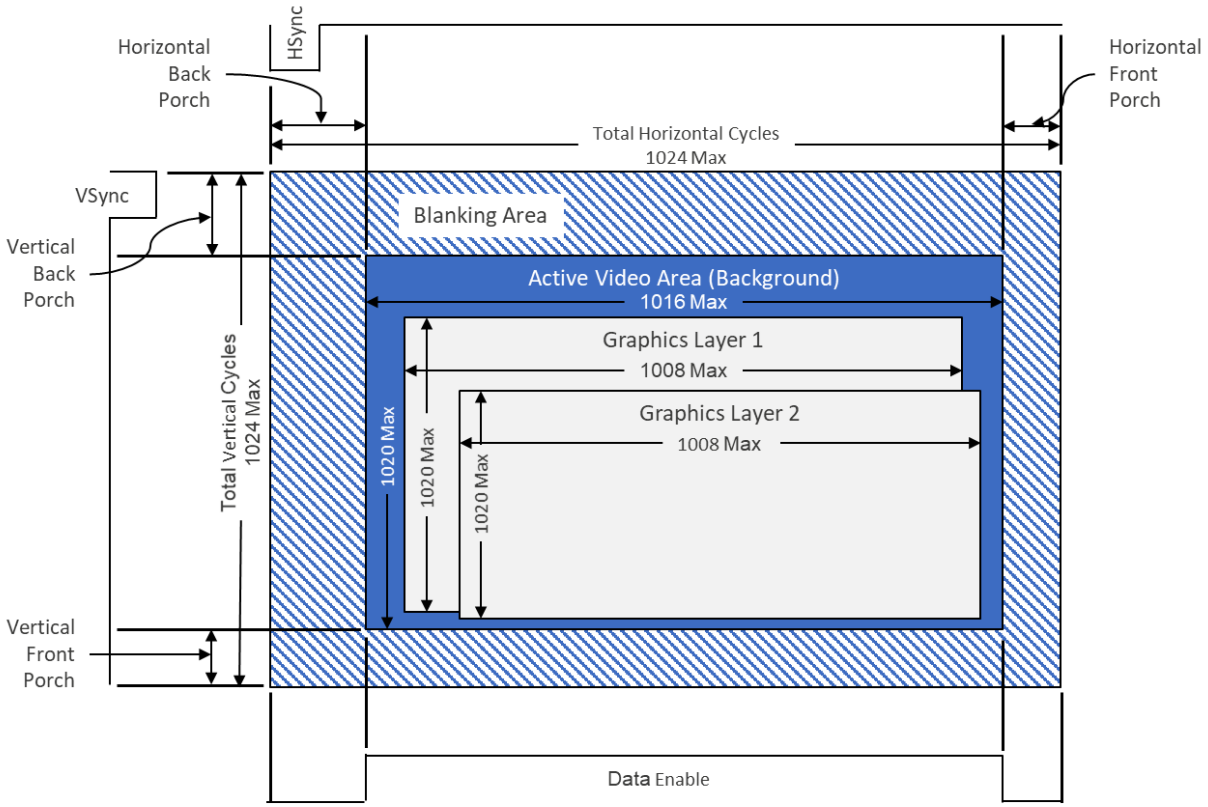


図 187:GLCDC HAL モジュール画面フォーマット

フロントポーチ期間

GLCDC モジュールは、垂直 / 水平フロントポーチサイクル / ラインに対する設定を備えていません。これらのサイクル / ラインは、合計水平サイクル / 垂直ライン設定に含まれている必要があります。

注 : このモジュールでは、GLCDC のハードウェア仕様に基づいて、バックポーチサイクル / ラインを設定する必要があります。一般的な LCD パネルのバックポーチサイクル / ラインは説明する数より多いため、これはモジュールの真の制限ではありません。

- 水平バックポーチサイクル数 >= 3
- 垂直バックポーチライン数 >= 2

パラメータ設定例

PE-HMI1 v2.0 ボード (LXD Research & Display, LLC, M7504A):

以下の例では、60Hz の LCD パネルリフレッシュレートを生成するため、水平合計サイクル、垂直合計ライン、パネルクロック分周比を調整しています。LCD パネルの記号については、M7504A データシートを参照してください。

LCD パネルのパラメータの設定 - PE-HMI1 v2.0 ボード

ISDE Property	Setting
Panel clock source select	Internal clock
Graphics screen 1/2 input horizontal size	800(thd)
Graphics screen 1/2 input vertical size	480(tvd)
Graphics screen 1/2 input horizontal size	800(thd)
Horizontal total cycles	976(th)
Horizontal active video cycles	800(thd)
Horizontal back porch cycles	46(thp+thb)
Horizontal sync signal cycles	20(thp)
Horizontal sync signal polarity	Low active
Vertical total lines	512(tv)
Vertical active video lines	480(tvd)
Vertical back porch cycles	23(tvp+tvb)
Vertical sync signal lines	10(tvp)
Vertical sync signal polarity	Low active
Output format	24bits RGB888
Data enable signal polarity	High active
Sync edge	Rising edge
TCON - Hsync pin select	LCD_TCON0
TCON - Vsync pin select	LCD_TCON1
TCON - DataEnable pin select	LCD_TCON2
TCON - Panel clock division ratio	1/8

DK-S7G2 v3.0 ボード (LXD Research & Display, LLC, M7190A):

以下の例では、60Hz の LCD パネルリフレッシュレートを生成するため、水平合計サイクル、垂直合計ライン、パネルクロック分周比を調整しています。LCD パネルの記号については、M7190A データシートを参照してください。

LCD パネルのパラメータの設定 - DK-S7G2 v3.0 ボード

ISDE Property	Setting
Panel clock source select	Internal clock
Graphics screen N input horizontal size	480(thd)
Graphics screen N input vertical size	272(tvd)
Graphics screen N input horizontal stride	480(thd)
Horizontal total cycles	582(th)
Horizontal active video cycles	480(thd)
Horizontal back porch cycles	43(thp+thb)
Horizontal sync signal cycles	41(thp)
Horizontal sync signal polarity	Low active
Vertical total lines	286(tv)
Vertical active video lines	272(tvd)
Vertical back porch cycles	12(tvp+tvb)
Vertical sync signal lines	10(tvp)
Vertical sync signal polarity	Low active
Output format	16bits RGB565
Data enable signal polarity	High active
Sync edge	Rising edge
TCON - Hsync pin select	LCD_TCON1
TCON - Vsync pin select	LCD_TCON2
TCON - DataEnable pin select	LCD_TCON0
TCON - Panel clock division ratio	1/24

SK-S7G2 v2.0 ボード (ILI Technology Corp., IL9341C):

以下の例では、水平合計サイクルと垂直合計ラインを、約 76.8Hz の LCD パネルリフレッシュレートのパネルに許容される最大値に設定しています。LCD パネルの記号については、LIL9314V データシートを参照してください。

LCD パネルのパラメータの設定 - SK-S7G2 v2.0 ボード

ISDE Property	Setting
Panel clock source select	Internal clock
Graphics screen N input horizontal size	256(see note)
Graphics screen N input vertical size	320(VAdr)
Graphics screen N input horizontal stride	256(see note)
Horizontal total cycles	290(HAdr+Hsync(16)+HBP(24)+HFP(16))
Horizontal active video cycles	240(HAdr)
Horizontal back porch cycles	40(Hsync(16)+HBP(24))
Horizontal sync signal cycles	16(Hsync)
Horizontal sync signal polarity	Low active
Vertical total lines	330(VAdr+Vsync(4)+VBP(2)+VFP(4))
Vertical active video lines	320(VAdr)
Vertical back porch cycles	6(Vsync+VBP)
Vertical sync signal lines	4(Vsync)
Vertical sync signal polarity	Low active
Output format	16bits RGB565
Data enable signal polarity	High active
Sync edge	Rising edge
TCON - Hsync pin select	LCD_TCON2
TCON - Vsync pin select	LCD_TCON1
TCON - DataEnable pin select	LCD_TCON0
TCON - Panel clock division ratio	1/32

注：パネルのパラメータは 240 ピクセルに設定する必要がありますが、入力幅および水平ストライドは、意図的に 256 ピクセルに設定されています。これは、GLCDC ハードウェアに合わせ、水平ラインを 64 バイトにする必要があるためです。ライン開始点での 240 ピクセルが有効であれば、ラインの残りのピクセル（16 ピクセル）は考慮されません。

CLUT

GLCDC モジュールは、色フォーマットが ARGB1555、CLUT8、CLUT4、または CLUT1 の場合に使用されるカラールックアップテーブルをサポートします。CLUT API は、CLUT0/CLUT1 SRAM（GLCDC ハードウェア内部で実装）を、グラフィックの前景画面または背景画面それぞれに対して更新できます。

注：CLUT を使用する色フォーマットを選択する場合は、display_api_t::start API を使用する前に、必ず CLUT API を呼び出してください。そうしないと、CLUT0 と CLUT1 が不明な状態となり、グラフィックが正しく表示されません。

また、実行時に CLUT API を呼び出して、CLUT SRAM を更新することもできます。

注：API は、CLUT データのソースを、現在使用されていない CLUT SRAM にコピーします（各 CLUT SRAM はピンポンバッファで構成されます）。CLUT データ更新の完了後、API は、画像が切れるのを避けるため、GLCDC ハードウェアが読み取る CLUT SRAM を次のフレームから自動的に切り替えます。

ライン繰り返しモード

ライン繰り返しは、メモリが十分にないシステムで特に重要なモードです。このモードでは、GLCDC モジュールは LCD パネルの画面サイズより少ないピクセルのラスタ画像を読み込み、ラスタを繰り返し画面に表示します。下の図は、背景グラフィックプレーンに小さいラスタ画像を繰り返し読み込むことで構築された画面の例です。

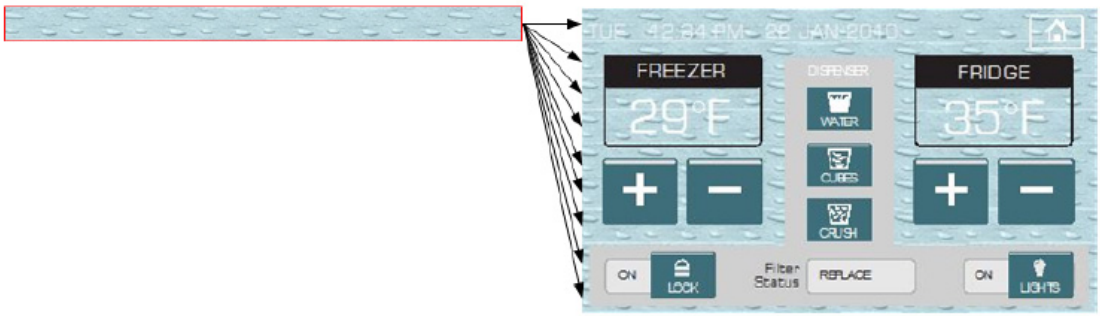


図 188:GLCDC HAL モジュールライン繰り返しモード

注：このモードを有効にするには、Synergy コンフィギュレータで GLCDC モジュールのプロパティ [Input - Graphics screen N input lines repeat]（N = 1 または 2）を [ON] に設定します。また、ラスタ画像を読み込む繰り返し回数を [Input - Graphics screen N input lines repeat times] と指定します。ラスタ画像の水平ピクセルサイズを [Input - Graphics screen N input horizontal size] と [Input - Graphics screen N input horizontal stride] に指定し、ラスタ画像の垂直ピクセルサイズを [Input - Graphics screen N input vertical size] に指定します。

ガンマ補正

ガンマ補正は、LCD パネルの色特性をフラットな特性に変更するために使用します。下の図は、GLCDC モジュールで設定可能なガンマ補正カーブです。このモジュールは、（R、G、B）色のそれぞれについて、入力色レベルに対して 16 個のしきい値をサポートしており、16 のエリアそれぞれに対してしきい値で割ったゲインレベルを定義します。

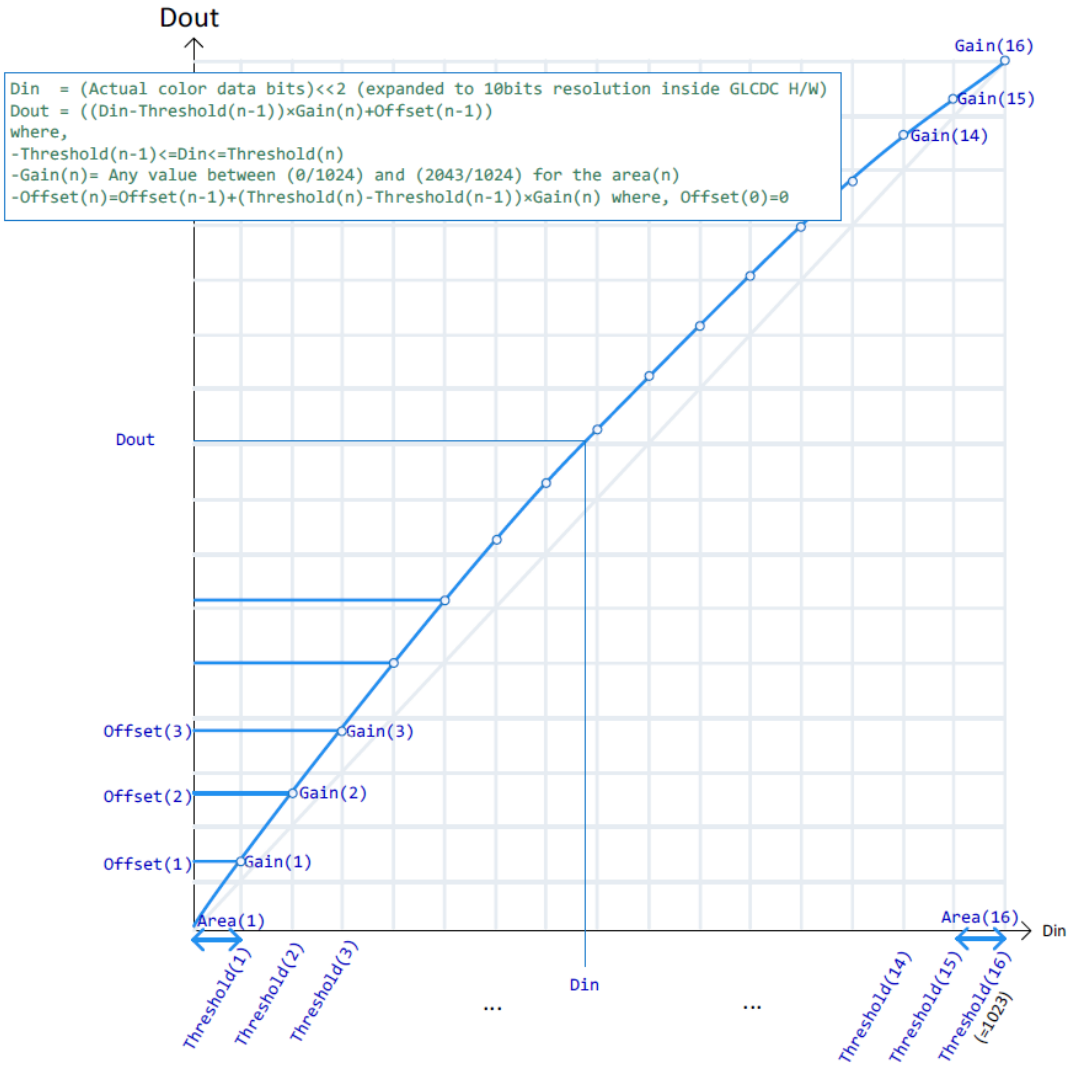


図 189:GLCDC HAL モジュールガンマ補正カーブ

注：(R、G、B) チャネルのそれぞれについてガンマ補正を有効にするには、Synergy コンフィギュレータを使用して GLCDC モジュールのプロパティ [Color correction – Gamma correction (R, G, B)] を [ON] に設定します。しきい値（合計 16 個）は、[Color correction – Gamma correction threshold (R, G, B) \[n\]]（n=[0..15]）に設定します。各エリアのゲイン値は、[Color correction – Gamma correction gain (R, G, B) \[n\]]（n=[0..15]）に設定します。

GLCDC HAL モジュールの動作に関する重要な注意事項と制限事項

GLCDC HAL モジュールの動作に関する重要な注意事項

以降のセクションで説明するように、複数の GLCDC 割り込みを構成することもできます。

ライン検出割り込み

ライン検出割り込みは、GLCDC がすべてのラインの LCD パネルへの出力を完了し、ブランク期間になったことを示すために使用されます。この割り込みを使用してグラフィックシステムでのフレームバッファの切り替えを処理し、3 つ以上のフレームでフレームバッファを使用します。

layer1 または layer2 バッファアンダーフローの割り込み

GLCDC layer1 または layer2 のバッファアンダーフロー割り込みを使用すると、システムのメモリ帯域幅不足を検出できます。バッファアンダーフローが発生するのは、メモリ（SDRAM や SRAM など）から GLCDC の内部ラインバッファへのグラフィックデータ転送が、他のデータ転送によってブロックされ、GLCDC ラインバッファから LCD パネルインタフェースへのデータ転送に対して十分でない場合です。この割り込みが発生しないように、グラフィックシステムを設計する必要があります。

GLCDC コールバック

ユーザーコールバック関数を open で登録できます。ユーザーコールバック関数が指定されている場合、割り込みが発生するたびに割り込みサービスルーチン（ISR）からコールバック関数が呼び出されます。コールバック関数イベントの引数では、グラフィックシステムで発生したイベントをユーザーが識別できるように、表に示す列挙値を受け取ることができます。DISPLAY_EVENT_LINE_DETECTION イベントは、画面を更新するためにフレームバッファを切り替えるために使用でき、DISPLAY_EVENT_GRn_UNDERFLOW イベントは、アンダーフローが発生した場合のエラー処理に使用できます。

イベントと割り込みの要約

Name of Event	Name of Interrupt	Condition for the Event
DISPLAY_EVENT_LINE_DETECTION	Line detection	When GLCDC is done outputting the last line in the active video region.
DISPLAY_EVENT_GR1_UNDERFLOW	Graphics 1 underflow	When GLCDC underflows during reading the data for graphics1 plane.
DISPLAY_EVENT_GR1_UNDERFLOW	Graphics 2 underflow	When GLCDC underflows during reading the data for graphics2 plane.

注 : コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。

GLCDC HAL モジュールの制限事項

- r_glcd 上のディスプレイドライバーは、RGB インデックス彩度キーをサポートしていません。
- r_glcd 上のディスプレイドライバーは、イベントリンク機能をサポートしていません。

4.2.12.4 アプリケーションへの GLCDC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに GLCDC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

ディスプレイドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに単純に追加します。(ディスプレイドライバーのデフォルト名は g_display0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

GLCDC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_display0 Display Driver on r_glcdc	Threads	New Stack> Driver> Graphics> Display Driver on r_glcdc

次の図に示すように、r_glcdc のディスプレイドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

青色の帯のモジュールは共有または共通モジュールです。それらは一度だけ追加する必要があり、複数のスタックで使用できます。ピンク色の帯のモジュールでは、ローレベルモジュールを選択する必要があります。これらはオプションまたは推奨のモジュールです (テキストを含めてブロックに示されます)。ローレベルモジュールを追加する必要がある場合は、モジュールの説明でテキストに Add を含めます。ピンク色の帯のモジュールをクリックすると、新規アイコンが表示され、選択可能な項目が表示されます。

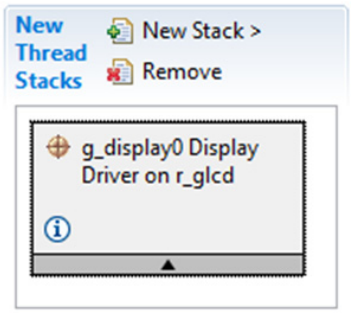


図 190:GLCDC HAL モジュールのスタック

4.2.12.5 GLCDC HAL モジュールの構成

ユーザーは必要な動作に合わせて GLCDC HAL モジュールを構成する必要があります。ユーザーがアクセス可能なすべてのプロパティで使用可能な構成設定項目とデフォルト値は、SSP コンフィギュレータのプロパティタブに表示され、参照しやすいように表に示されています。

また、変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらは、ISDE の [Properties] ウィンドウではロックされたプロパティに対するロックアイコンで示されます。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウ内にあります。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。ISDE の [Properties] ウィンドウに一覧表示される割り込み優先順位に、MCU ターゲット（CM4 または CM0+）に基づく設定の有効性に応じた表示が含まれることにも注意してください。このレベルの詳細情報は以下の構成プロパティ表には含まれませんが、割り込み優先順位を設定する際に ISDE ですぐに確認できます。

注 : ISDE を開き、次に示す構成テーブルの設定を見ながら、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発の表と裏を学習するための有用な「実践的」アプローチになる可能性があります。

r_glcdc での GLCDC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_display0	The name to be used for a GLCDC module control block instance. This name is also used as the prefix of the other variable instances.
Name of display callback function to be defined by user	NULL	Name must be a valid C symbol.
Input – Panel clock source select	Internal clock(GLCDCLK), External clock(LCD_EXTCLK) Default: Internal clock (GLCDCLK)	Choose the panel clock source depends on your system.

ISDE Property	Value	Description
Input – Graphics screen1	Used, Not used Default: Used	Specify "Used" if the graphics screen N is used. Then the frame buffer named "display_fb_background" for graphics screen1 and "display_fb_foreground" for graphics screen2 is autogenerated by ISDE. If not using either of the graphics screens, specify "Not used". Then the frame buffer is not created. Note that there is no memory read access to the frame buffer when you specify "Not used", which reduces the consumption of bus bandwidth.
Input – Graphics screen1 frame buffer name	fb_background	Custom name for frame buffer.
Input – Number of Graphics screen1 frame buffer	2	Number of graphics selection.
Input – section where Graphics screen1 frame buffer allocated	sdram	Specify the section name to allocate the frame buffer. This is valid if "Input - Graphics screen1" is set as "Used."
Input – Graphics screen1 input horizontal size	800	Specify the number of horizontal pixels. Default value is the size for an image with 800x480 pixels.
Input – Graphics screen1 vertical size	480	Specify the number of vertical pixels. Default value is the size for an image with 800x480 pixels.

ISDE Property	Value	Description
Input – Graphics screen1 input horizontal stride (not bytes but pixels)	800	Specify the memory stride for a horizontal line. This value must be specified with the number of pixels, not actual bytes. Typically, this parameter is set to same number as parameter 'input horizontal size'. Default value is the size for an image with 800x480 pixels.
Input – Graphics screen1 input format	32 bits ARGB888, 32 bits RGB888, 16 bits RGB565, 16 bits ARGB1555, 16 bits ARGB4444, CLUT 8, CLUT 4, CLUT 1 Default: 16 bits RGB565	Specify the graphics screen Input format. If selecting CLUT formats, you must write CLUT data using clut before performing start. Default setting supports a RGB565 formatted image.
Input – Graphics screen1 input line descending	Used, Not used Default: Not used	Specify "On" if image data descends from the bottom line to the top line in the frame buffer. Usually "Off".
Input – Graphics screen1 input line repeat	On, Off Default: Off	Specify "On" if expecting to repeatedly read a raster image which is smaller than the LCD panel size. Usually "Off". For details, see the description of Line Repeating function.
Input – Graphics screen1 input line repeat times	0	Specify the number of repeating times for a raster image which is read repeatedly in a frame.
Input – Graphics screen1 layer coordinate X	0	Specify the horizontal offset in pixels of the graphics screen from the background screen.
Input – Graphics screen1 layer coordinate Y	0	Specify the vertical offset in pixels of the graphics screen from the background screen.

ISDE Property	Value	Description
Input – Graphics screen1 layer background color alpha	255	Based on the alpha value, either the graphics screen2 (foreground graphics screen) is blended into the graphics screen1 (background graphics screen) or the graphics screen1 is blended into the monochrome background screen.
Input – Graphics screen1 layer background color Red	255	Specify the background color in the graphics screen N.
Input – Graphics screen1 layer background color Green	255	Specify the background color in the graphics screen N.
Input – Graphics screen1 layer background color Blue	255	Specify the background color in the graphics screen N.
Input – Graphics screen1 layer fading control	None, Fade-in, Fade-out Default: None	Specify "On" when performing a fade-in for the graphics screen. The transparent screen changes gradually to opaque. Specify "Off" when performing the fade-out for the graphics screen. The opaque screen changes gradually to transparent. Note that this processing is accelerated by the GLCDC hardware and cannot stop once started. The transition status can be monitored by statusGet.
Input – Graphics screen1 layer fade speed	0	Specify the number of frames for the fading transition to complete.

ISDE Property	Value	Description
Input – Graphics screen2	Used, Not used Default: Not used	Specify "Used" if the graphics screen N is used. Then the frame buffer named "display_fb_background" for graphics screen1 and "display_fb_foreground" for graphics screen2 is autogenerated by ISDE. If not using either of the graphics screens, specify "Not used". Then the frame buffer is not created. Note that there is no memory read access to the frame buffer when you specify "Not used", which reduces the consumption of bus bandwidth.
Input – Graphics screen2 frame buffer name	fb_foreground	Custom name for frame buffer.
Input – Number of Graphics screen2 frame buffer	2	Number of graphics selection.
Input – section where Graphics screen2 frame buffer allocated	sdram	Specify the section name to allocate the frame buffer. This is valid if "Input - Graphics screen1" is set as "Used."
Input – Graphics screen2 input horizontal size	800	Specify the number of horizontal pixels. Default value is the size for an image with 800x480 pixels.
Input – Graphics screen2 vertical size	480	Specify the number of vertical pixels. Default value is the size for an image with 800x480 pixels.

ISDE Property	Value	Description
Input – Graphics screen2 input horizontal stride (not bytes but pixels)	800	Specify the memory stride for a horizontal line. This value must be specified with the number of pixels, not actual bytes. Typically, this parameter is set to same number as parameter 'input horizontal size'. Default value is the size for an image with 800x480 pixels.
Input – Graphics screen2 input format	32 bits ARGB888, 32 bits RGB888, 16 bits RGB565, 16 bits ARGB1555, 16 bits ARGB4444, CLUT 8, CLUT 4, CLUT 1 Default: 16 bits RGB565	Specify the graphics screen Input format. If selecting CLUT formats, you must write CLUT data using clut before performing start. Default setting supports a RGB565 formatted image.
Input – Graphics screen2 input line descending	On, Off Default: Off	Specify "On" if image data descends from the bottom line to the top line in the frame buffer. Usually "Off".
Input – Graphics screen2 input line repeat	On, Off Default: Off	Specify "On" if expecting to repeatedly read a raster image which is smaller than the LCD panel size. Usually "Off". For details, see the description of Line Repeating function.
Input – Graphics screen2 input line repeat times	0	Specify the number of repeating times for a raster image which is read repeatedly in a frame.
Input – Graphics screen2 layer coordinate X	0	Specify the horizontal offset in pixels of the graphics screen from the background screen.
Input – Graphics screen2 layer coordinate Y	0	Specify the vertical offset in pixels of the graphics screen from the background screen.

ISDE Property	Value	Description
Input – Graphics screen2 layer background color alpha	255	Based on the alpha value, either the graphics screen2 (foreground graphics screen) is blended into the graphics screen1 (background graphics screen) or the graphics screen1 is blended into the monochrome background screen.
Input – Graphics screen2 layer background color Red	255	Specify the background color in the graphics screen N.
Input – Graphics screen2 layer background color Green	255	Specify the background color in the graphics screen N.
Input – Graphics screen2 layer background color Blue	255	Specify the background color in the graphics screen N.
Input – Graphics screen2 layer fading control	None, Fade-in, Fade-out Default: None	Specify "On" when performing a fade-in for the graphics screen. The transparent screen changes gradually to opaque. Specify "Off" when performing the fade-out for the graphics screen. The opaque screen changes gradually to transparent. Note that this processing is accelerated by the GLCDC hardware and cannot stop once started. The transition status can be monitored by statusGet.
Input – Graphics screen2 layer fade speed	0	Specify the number of frames for the fading transition to complete.

ISDE Property	Value	Description
Output – Horizontal total cycles	1024	Specify the total cycles in a horizontal line. Set to the number of cycles defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.
Output – Horizontal active video cycles	800	Specify the number of active ideo cycles in a horizontal line. Set to the number of cycles defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.
Output – Horizontal back porch cycles	46	Specify the number of back porch cycles in a horizontal line. Back porch starts from the beginning of Hsync cycles, which means back porch cycles contain Hsync cycles. Set to the number of cycles defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.
Output – Horizontal sync signal cycles	20	Specify the number of Hsync signal assertion cycles. Set to the number of cycles defined in the data sheet of LCD panel sheet in your system. Default value matches LCD panel on S7G2 PE-HMI1 board.
Output – Horizontal sync signal polarity	Low active, High active Default: Low active	Select the polarity of Hsync signal to match your system. Default setting matches the LCD panel on S7G2 PE-HMI1 board.

ISDE Property	Value	Description
Output – Vertical total lines	525	Specify number of total lines in a frame. Set to the number of lines defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.
Output – Vertical active video lines	480	Specify the number of active video lines in a frame. Set to the number of lines defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.
Output – Vertical back porch lines	23	Specify the number of back porch lines in a frame. Back porch starts from the beginning of Vsync lines, which means back porch lines contain Vsync lines. Set to the number of lines defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.
Output – Vertical sync signal lines	10	Specify the Vsync signal assertion lines in a frame. Set to the number of lines defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.
Output – Vertical sync signal polarity	Low active, High active Default: Low active	Select the polarity of Vsync signal to match to your system. Default setting matches LCD panel on S7G2 PE-HMI1 board.

ISDE Property	Value	Description
Output – Format	24bits RGB888, 18bits RGB666, 16 bits RGB565, 8bits serial Default: 24bits RGB888	Specify the graphics screen output format to match to your LCD panel. Default setting matches the LCD panel on S7G2 PE-HMI1 board.
Output – Endian	Little endian, Big endian Default: Little endian	Select data endian for output signal to LCD panel. Default setting matches the LCD panel on S7G2 PE-HMI1 board.
Output – Color order	RGB, BGR Default: RGB	Select data order for output signal to LCD panel. The order of blue and red can be swapped if needed. Default setting matches the LCD panel on S7G2 PE-HMI1 board.
Output – Data Enable Signal Polarity	Low active, High active Default: High active	Select the polarity of Data Enable signal to match to your system. Default setting matches the LCD panel on S7G2 PE-HMI1 board.
Output – Sync edge	Rising Edge, Falling Edge Default: Rising Edge	Select the polarity of Sync signals to match to your system. Default setting matches the LCD panel on S7G2 PE-HMI1 board.
Output – Background color alpha channel	255	Specify the background color of the background screens.
Output – Background color R channel	0	Specify the background color of the background screens.
Output – Background color G channel	0	Specify the background color of the background screens.
Output – Background color B channel	0	Specify the background color of the background screens.

ISDE Property	Value	Description
CLUT	Used, Not used Default: Not used	Specify "Used" if selecting CLUT formats for a graphics screen input format. Then, a buffer named "CLUT_buffer" for the CLUT source data is generated in the ISDE auto-generated source file.
CLUT - CLUT buffer size	256	Specify the number of entries for the CLUT source data buffer. Each entries consumes 4 bytes (1 word). Words of CLUT source data specified by this parameter are generated in the ISDE auto-generated source file.
TCON – Hsync pin select	Not used, LCD_TCON0, LCD_TCON1, LCD_TCON2, LCD_TCON3 Default: LCD_TCON0	Select the TCON pin used for the Hsync signal to match to your system. Default setting is for LCD panel on S7G2 PE-HMI1 board.
TCON – Vsync pin select	Not used, LCD_TCON0, LCD_TCON1, LCD_TCON2, LCD_TCON3 Default: LCD_TCON1	Select TCON pin used for Vsync signal to match to your system. Default setting is for LCD panel on S7G2 PE-HMI1 board.
TCON – DataEnable pin select	Not used, LCD_TCON0, LCD_TCON1, LCD_TCON2, LCD_TCON3 Default: LCD_TCON2	Select TCON pin used for DataEnable signal to match to your system. Default setting is for LCD panel on S7G2 PE-HMI1 board.
TCON – Panel clock division ratio	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, 1/9, 1/12, 1/16, 1/24, 1/32 Default: 1/8	Select the clock source divider value. See the note at bottom of this table about the source clock for the pixel clock.
Color correction – Brightness	Off, On Default: Off	Specify "On" when performing brightness control. If specifying "Off", the setting below does not affect the output color.

ISDE Property	Value	Description
Color correction – Brightness R channel	512	Output color level is calculated as follows: Output color level = Input color level +/ - 512. Set the value for each of R, G, B channels.
Color correction – Brightness G channel	512	Output color level is calculated as follows: Output color level = Input color level +/ - 512. Set the value for each of R, G, B channels.
Color correction – Brightness B channel	512	Output color level is calculated as follows: Output color level = Input color level +/ - 512. Set the value for each of R, G, B channels.
Color correction – Contrast	Off, On Default: Off	Specify "On" when performing contrast control. If specifying "Off", the setting below does not affect the output color.
Color correction – Contrast(gain) R channel	128	Output color level is calculated as follows: Output color level = Input color level x (/128). Set the value for each of R, G, B channels.
Color correction – Contrast(gain) G channel	128	Output color level is calculated as follows: Output color level = Input color level x (/128). Set the value for each of R, G, B channels.
Color correction – Contrast(gain) B channel	128	Output color level is calculated as follows: Output color level = Input color level x (/128). Set the value for each of R, G, B channels.
Color correction – Gamma correction(Red)	Off, On Default: Off	Control for each channel R/G/B. Specify "On" when performing gamma correction for the red channel. If specifying "Off", the settings for gain and threshold do not affect the output color.

ISDE Property	Value	Description
Color correction – Gamma gain R[0-15]	0	Set the gain value for the red channel in the area N on the gamma correction curve. The gain setting for area N is applied to the input data with a color level between ((Gamma threshold R[N-1])<<2) and ((Gamma threshold R[N])<<2). The output value is calculated as follows: Output color level = Input color level / 1024 (/128).
Color correction – Gamma threshold R[0-15]	0	Set the threshold value for the red channel in the area N on the gamma correction curve. The gain setting for area N is applied to the input data with a color level between Gamma threshold R[N-1] and Gamma threshold R[N]. The output value is calculated as follows: Output color level = Input color level / 1024 (/128).
Color correction – Gamma correction(Green)	Off, On Default: Off	Control for each channel R/G/B. Specify "On" when performing gamma correction for the red channel. If specifying "Off", the settings for gain and threshold do not affect the output color.
Color correction – Gamma gain G[0-15]	0	Set the gain value for the red channel in the area N on the gamma correction curve. The gain setting for area N is applied to the input data with a color level between ((Gamma threshold R[N-1])<<2) and ((Gamma threshold R[N])<<2). The output value is calculated as follows: Output color level = Input color level / 1024 (/128).

ISDE Property	Value	Description
Color correction – Gamma threshold G[0-15]	0	Set the threshold value for the red channel in the area N on the gamma correction curve. The gain setting for area N is applied to the input data with a color level between Gamma threshold R[N-1] and Gamma threshold R[N]. The output value is calculated as follows: Output color level = Input color level / 1024 (/128).
Color correction – Gamma correction(Blue)	Off, On Default: Off	Control for each channel R/G/B. Specify "On" when performing gamma correction for the red channel. If specifying "Off", the settings for gain and threshold do not affect the output color.
Color correction – Gamma gain B[0-15]	0	Set the gain value for the red channel in the area N on the gamma correction curve. The gain setting for area N is applied to the input data with a color level between ((Gamma threshold R[N-1])<<2) and ((Gamma threshold R[N])<<2). The output value is calculated as follows: Output color level = Input color level / 1024 (/128).
Color correction – Gamma threshold B[0-15]	0	Set the threshold value for the red channel in the area N on the gamma correction curve. The gain setting for area N is applied to the input data with a color level between Gamma threshold R[N-1] and Gamma threshold R[N]. The output value is calculated as follows: Output color level = Input color level / 1024 (/128).

ISDE Property	Value	Description
Dithering	Off, On Default: Off	Dithering enable. Specify "On" when applying the dither effect to reduce the banding in case of selecting RGB666 or RGB565 output formats. Dithering can be applied when converting. If specified "Off", the settings for dithering below do not affect the output. For details on the dither effect, see Output Control Block Panel Dither Correction Register (OUT_PDTHA) in the hardware manual.
Dithering – Mode	Truncate, Round off, 2x2 Pattern Default: Truncate	Specify the dither mode. For detail, see the Output Control Block Panel Dither Correction Register (OUT_PDTHA) in the hardware manual.
Dithering – Pattern A	Pattern 00, Pattern 01, Pattern 10, Pattern 11 Default: Pattern 11	Specify the dither pattern for 2X2 pattern mode. For details, see the Output Control Block Panel Dither Correction Register (OUT_PDTHA) in the hardware manual.
Dithering – Pattern B	Pattern 00, Pattern 01, Pattern 10, Pattern 11 Default: Pattern 11	Specify the dither pattern for 2X2 pattern mode. For details, see the Output Control Block Panel Dither Correction Register (OUT_PDTHA) in the hardware manual.
Dithering – Pattern C	Pattern 00, Pattern 01, Pattern 10, Pattern 11 Default: Pattern 11	Specify the dither pattern for 2X2 pattern mode. For details, see the Output Control Block Panel Dither Correction Register (OUT_PDTHA) in the hardware manual.

ISDE Property	Value	Description
Dithering – Pattern D	Pattern 00, Pattern 01, Pattern 10, Pattern 11 Default: Pattern 11	Specify the dither pattern for 2X2 pattern mode. For details, see the Output Control Block Panel Dither Correction Register (OUT_PDTHA) in the hardware manual.
Misc – Correction Process Order	Brightness and Contrast then Gamma, Gamma then Brightness and Contrast Default: Brightness and Contrast then Gamma	Specify the color correction processing order if needed.
Line Detect Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Default: Disabled	The driver needs valid interrupt priority setting and it won't work if disabled.
Underflow 1 Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Default: Disabled	The driver needs valid interrupt priority setting and it won't work if disabled.
Underflow 2 Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Default: Disabled	The driver needs valid interrupt priority setting and it won't work if disabled.

注:例の値とデフォルトは、Synergy S7G2 MCU ファミリを使用するプロジェクト向けです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

GLCDC HAL モジュールのクロック構成

GLCDC モジュールは、以下のいずれかのクロックソースからピクセルクロックを生成できます。ソースクロックの選択は、e^2 studio の [Synergy Configuration] で行うことができます。

- 内部クロックソース (PLLOUT; 240 MHz)
- LCD_EXTCLK ピンからの外部クロックソース

注 : S7G2 WS1（Working Sample1）チップと WS2（Working Sample2）チップ以降では、内部クロックが異なります。WS1 チップは PCLKB(最大 60MHz)を使用しますが、WS2 以降のチップは PLLOUT(最大 240MHz)を使用します。

GLCDC HAL モジュールのピン構成

GLCDC モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。ピン選択の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、構成設定の表では GLCDC ピンを選択する例を示します。

r_glcdc の GLCDC HAL モジュールのピン選択

Resource	ISDE Tab	Stacks Selection Sequence
GLCDC	Pins	Select Peripherals > Graphics: GLCDC> GLCDC0

注 : 選択シーケンスでは、GLCDC0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_glcdc の GLCDC HAL モジュールのピン構成設定

Property	Value	Description
Pin Group Selection	Mixed, _A Only, _B Only Default: Mixed	Pin group selection
Operation Mode	Disabled, Custom, RGB888, RGB666, RGB565 Default: Disabled	Select desired operation mode
LCD_CLK	None, P900, P101 Default: None	LCD_CLK Pin
LCD_DATA00:15	None, Pn, Pm Default: None	LCD_DATA Pins
LCD_TCON0:3	None, Pn, Pm Default: None	LCD_TCON Pins
LCD_EXTCLK	None, Pn, Pm Default: None	LCD_EXTCLK Pin

注 : 表で示されている例の値は、Synergy S7G2 および SK-S7G2 キットのプロジェクト向けです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

注 : S7G2 PE-HMI1 ボード上で GLCDC モジュールを使用するには、PORT10 ピン 3（PA03）とピン 5（PA05）を、出力レベル HIGH の IOPORT ピンとして設定してください。ピン PA03 は DISP 信号（ディスプレイオン / オフ）を制御し、ピン PA05 は LCD パネルのバックライトを制御します。詳細については、S7G2 PE-HMI1 ボードの回路図を参照してください。

4.2.12.6 アプリケーションでの GLCDC HAL モジュールの使用

アプリケーションで GLCDC HAL モジュールを使用するときの一般的な手順は次のとおりです。

1) display_api_t::open API を使用して、GLCDC HAL モジュールを初期化します。
2) アプリケーションのコードで、フレームバッファにプライマリ画像を描画します。
3) display_api_t::start API で画像の表示を開始します。
4) display_api_t::stop API で画像の表示を停止します。
5) display_api_t::clut API でソース CLUT データを CLUT SRAM にコピーします。
6) アプリケーションコードを使用して表示を更新するために、フレームバッファに新しい画像を描画してます。
    a) 通常、ユーザーのシステムはピンポンフレームバッファシステムで構成されているので、描画時に表示に使用されていない別のフレームバッファに画像を描画します。
7) display_api_t::layerChange API を使用して、フレームバッファの切り替えを GLCDC ハードウェアに要求します。
    a) GLCDC ハードウェアは、フレームバッファを切り替えて、次のフレームから新しい画像を表示します。
8) display_api_t::correction API で色補正をします。
9) display_api_t::statusGet API で画面のステータスを取得します。
10) display_api_t::versionGet API でドライバーのバージョン情報を取得します（オプション）
11) display_api_t::close API を呼び出して GLCDC モジュールを閉じます。

アプリケーションのコードを現在のフレームバッファの描画完了と同期させるには、ライン検出割り込みを使用し、GLCDC コールバックを通してアプリケーションのコードにタイミングを通知します。

上記で説明した手順は共通の動作フローとして次の図のように表現できます。

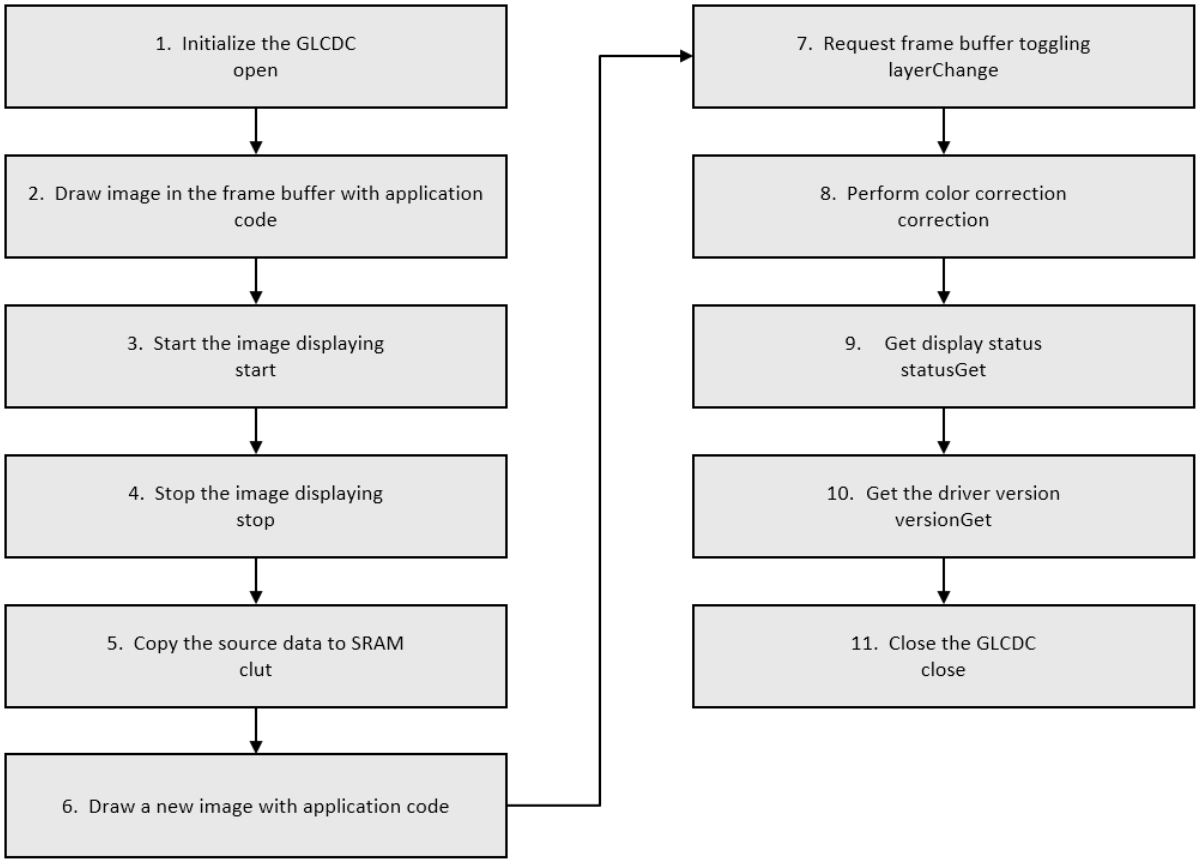


図 191:通常の GLCDC HAL モジュールアプリケーションのフロー図

4.2.13 データ操作回路ドライバー

データ演算回路（DOC）HAL モジュールは、DOC アプリケーション用のハイレベル API を提供し、Synergy MCU 上の DOC ペリフェラルを使用します。ユーザー定義のコールバックを作成し、イベント発生時に CPU に通知できます。

4.2.13.1 DOC HAL モジュールの特長

DOC HAL モジュールの周辺機能は 16 ビットデータを比較するために使用され、以下のイベントを検出できます。

- データ値の不一致または一致
- 加算演算のオーバーフロー
- 減算演算のアンダーフロー

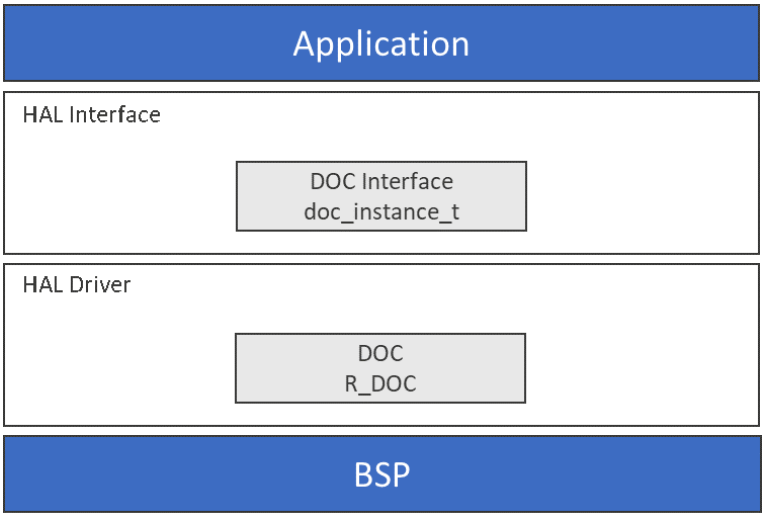


図 192:**DOC HAL モジュールのブロック図**

4.2.13.2 DOC HAL モジュールの API の概要

DOC HAL モジュールでは、データ演算回路のオープン、クローズ、状態の確認、データのライトのための API が定義されています。DOC HAL モジュールは、Synergy MCU 上の DOC 周辺機能を使用します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

DOC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_doc.p_api->open(g_doc.p_ctrl, g_doc.p_cfg) Initial configuration.
close	g_doc.p_api->close(g_doc.p_ctrl) Allow the driver to be reconfigured. Will reduce power consumption.

Function Name	Example API Call and Description
statusGet	g_doc.p_api->statusGet(g_doc.p_ctrl, &my_Status) Get the DOC status and stores it in the provided pointer p_status.
statusClear	g_doc.p_api->statusClear(g_doc.p_ctrl) Clear DOPCF status flag.
write	g_doc.p_api->write(g_doc.p_ctrl, value) Write to the DODIR and DODSR registers.
inputRegisterWrite	g_doc.p_api->inputRegisterWrite(g_doc.p_ctrl, &doc_values) Write to the DODIR register.
versionGet	g_doc.p_api->versionGet(g_doc.p_ctrl, &version) Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	DOC successfully configured.
SSP_ERR_IN_USE	Module already open.
SSP_ERR_ASSERTION	One or more pointers point to NULL.
SSP_ERR_INVALID_ARGUMENT	ISR is not enabled. Enable the ISR in bsp_irq_cfg.h.

Name	Description
SSP_ERR_HW_LOCKED	DOC resource is locked.
SSP_ERR_NOT_OPEN	Driver not open.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.13.3 DOC HAL モジュールの動作の概要

DOC HAL モジュールは、Synergy MCU 上の DOC 周辺機能を制御します。16 ビットデータの比較に使用され、データ値間の不一致、加算値のオーバーフロー、減算演算のアンダーフローを検出できます。コールバックを使用でき、関連する割り込みが有効になっている場合は、DOC イベントに対応してコールバック関数が呼び出されます。

DOC は、2 つのデータレジスタを使用して演算を実行します。DOC データ入力レジスタ（DOCDIR）は演算対象のデータを保持し、DOC データ設定レジスタ（DOCDSR）は入力データに対する演算に使用される値を保持します。加算モードと減算モードでは、このレジスタはデータ演算の結果を格納します。(これらのレジスタはどちらも 16 ビット幅です)。

DOC HAL モジュールの動作に関する重要な注意事項と制限事項

比較データの初期設定は、write API をコールすることで DOC に書き込まれます。write API は、DOC DODSR および DODIR レジスタに書き込みます。write API は、以下に示すように doc_data_t 型の変数を使用します。

```
doc_data_t g_doc_values;
   g_doc_values.dodir = 0x1000;
   g_doc_values.dodsr = 0x1000;
   g_doc.p_api->write(g_doc.p_ctrl, &g_doc_values);
```

比較対象のデータが変化しない場合は、比較が必要になるたびにそのデータを書き込み直す必要はありません。入力データの値は、inputRegisterWrite API を使用して DOC に書き込むことができます。inputRegisterWrite API は、DOC のデータ入力レジスタのみに書き込みます。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.13.4 アプリケーションへの DOC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに DOC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

DOC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(DOC ドライバーのデフォルト名は g_doc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

DOC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_doc0 Data Operation Circuit Driver on r_doc	Threads	New Stack > Driver > Monitoring > Data Operation Circuit Driver on r_doc

次の図に示すように、r_kint の Key Matrix ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

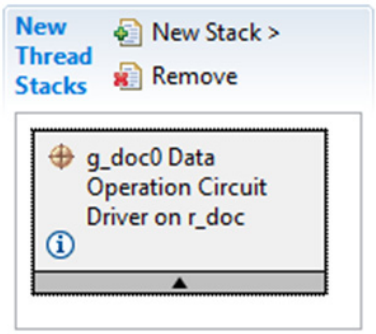


図 193:DOC HAL モジュールのスタック

4.2.13.5 DOC HAL モジュールの構成

ユーザーは必要な動作に合わせて DOC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_doc 上の DOC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_doc0	Module name.
Event	Comparison mismatch, Comparison match, Addition overflow, Subtraction underflow Default: Comparison mismatch	Specify the event which will trigger the DOC interrupt.
Callback	NULL	A user callback function can be registered in open. If this callback function is provided, it is called from the interrupt service routine (ISR) when the configured DOC event occurs. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
DOC Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Use the pull down to set the DOC interrupt priority.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

DOC HAL モジュールのクロック構成

DOC HAL モジュールには、特定のクロック構成は必要ありません。

DOC HAL モジュールのピン構成

DOC HAL モジュールには、特定のピン構成は必要ありません。

4.2.13.6 アプリケーションでの DOC HAL モジュールの使用

アプリケーションで DOC HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、DOC を初期化します。
2) write API を使用して、DODIR と DODSR にレジスタ値を設定します。
3) inputRegisterWrite API を使用して、DOC にデータをストリームします。
4) statusGet API またはコールバック（有効な場合）を使用して、比較の状態を読み取ります。
5) statusClear API を使用して、状態フラグをクリアします。
6) close API を使用して、モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

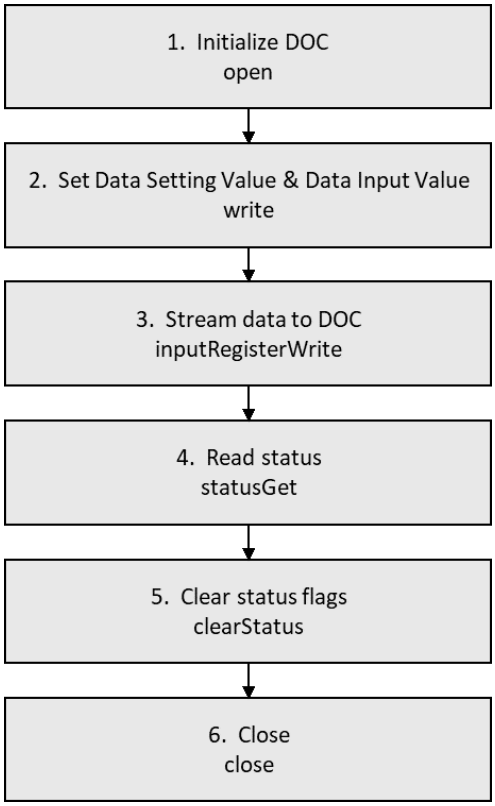


図 194:一般的な DOC HAL モジュールアプリケーションのフロー図

4.2.14 DMAC ドライバー

DMAC HAL モジュールは、データ転送アプリケーション用のハイレベル API を提供し、Synergy MCU 上の DMAC ペリフェラルを使用します。ユーザー定義のコールバックを作成し、転送イベント発生時に CPU に通知できます。

4.2.14.1 DMAC HAL モジュールの機能

DMAC HAL モジュールは、割り込みまたはイベントの発生時に、ユーザー指定の移動元からユーザー指定の移動先にデータを移動します。DMAC HAL モジュールは、以下の機能をサポートしています。

- Synergy MCU での DMAC モジュール
- 割り込み（必要な場合）
- 複数の転送モード
  - 単一転送
  - リピート転送
  - ブロック転送
  - アドレスインクリメントモードまたは固定モード
- 複数チャネル（数は使用されている MCU に依存）

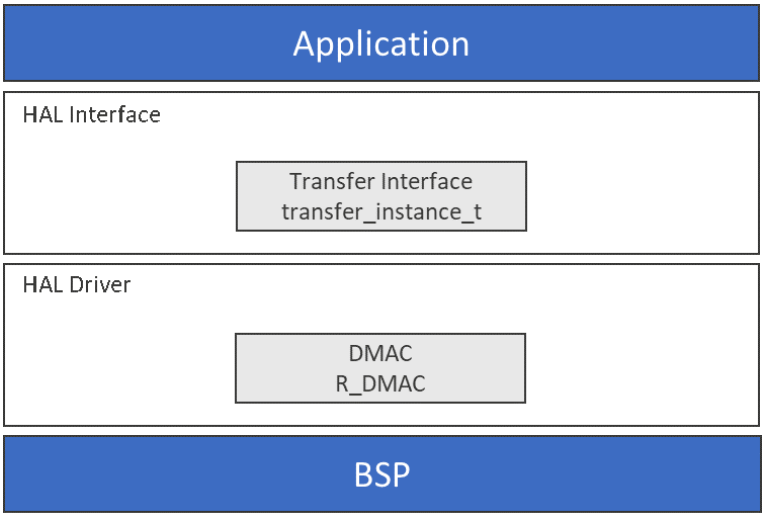


図 195:DMAC HAL モジュールのブロック図

4.2.14.2 DMAC HAL モジュールの API の概要

DMAC HAL モジュールでは、オープン、クローズ、タイマの開始、タイマの停止のための API が定義されています。データトランスファコントローラ（DTC）と DMAC は同じ転送インタフェースを使用することに注意してください。インタフェースを共有することで、DTC と DMA の実装の変更が容易になります。API の呼び出しは同じであり、ローレベルの実装から独立しています。次の表には、使用可能なすべての API のリスト、API コールの例、各関数の簡単な説明が記載されています。ステータス戻り値の表は API 要約表の後にあります。

DMAC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_transfer0.api->open(g_transfer0.p_ctrl, g_transfer0.p_cfg) Open device channel. Initialize driver and hardware on first call.
close	g_transfer0.api->close(g_transfer0.p_ctrl) Close device channel. Turns off hardware if last channel open.
reset	g_transfer0.api->reset(g_transfer0.p_ctrl, &source, &destination, number_of_transfers) Reset channel settings.
start	g_transfer0.api->start(g_transfer0.p_ctrl, mode) Start data transfer.
stop	g_transfer0.api->stop(g_transfer0.p_ctrl) Stop data transfer.
enable	g_transfer0.api->enable(g_transfer0.p_ctrl) Enable channel.
disable	g_transfer0.api->disable(g_transfer0.p_ctrl) Disable channel.
versionGet	g_transfer0.api->versionGet(&version) Retrieve the API version with the version pointer.

Function Name	Example API Call and Description
infoGet	g_transfer0.api->infoGet(g_transfer0.p_ctrl, &info) Get transfer channel info.
blockReset	g_transfer0.api->blockReset(g_transfer0.p_ctrl, &source, &destination, length, size, number_of_transfers) Reset Block Transfer parameters.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_ASSERTION	Parameter has invalid value.
SSP_ERR_NOT_OPEN	The channel is not opened.
SSP_ERR_UNSUPPORTED	Operation not configured correctly.
SSP_ERR_IN_USE	The channel specified has already been opened. No configurations were changed. Call the associated Close function or use associated Control commands to reconfigure the channel.
SSP_ERR_IRQ_BSP_DISABLED	IRQ not enabled in BSP.
SSP_ERR_NOT_ENABLED	Operation failed.
SSP_ERR_NOT_OPEN	The channel is not opened.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.14.3 DMAC HAL モジュールの動作の概要

DMAC と DTC を使用して、Synergy MCU 内でデータを移動することができます。これらのモジュールのいずれかを選択するときは、いくつかの考慮事項があり、各自のアプリケーションにどの実装が最適化かを判断する必要があります。DTC モジュールは大部分の一般的な転送アプリケーションに推奨され、データ転送操作でも使用できます。以下のユースケースで、各転送モジュール内での動作を説明します。

DTC 転送モジュールの選択

DTC HAL モジュールでは、システム内のすべての割り込みに対するスロットを持つ、RAM ベースのベクタテーブルが使用されます。DTC 転送が完了すると、アクティベーション ソース割り込みが呼び出されます。DTC を使用するには、アクティベーション ソース割り込みを有効にする必要があります。一般に、アクティベーションソース割り込みは、TRANSFER_IRQ_EACH が構成で指定されていない限り、転送が完了するまで DTC によってミュートされます。たとえば、長さが 16 である通常モード転送がタイマーによってトリガーされた場合、転送が有効である間、最初の 15 回のタイマー割り込みは発生しません。タイマー割り込みは 16 番目の転送の後で発生します。DTC ではチェーンされた転送も許可されます。つまり、単一のアクティベーション ソース割り込みの後で複数の転送が発生できます。この機能はドライバーによってサポートされていますが、ISDE の外部で構成する必要があります。

DMAC 転送モジュールの選択

DMAC HAL モジュールは、割り込みまたはイベントの発生時に、ユーザー指定の移動元からユーザー指定の移動先にデータを移動します。DMAC HAL モジュールは DMAC 周辺レジスタを使用するため、システムでの転送回数はデバイス上の DMAC チャネルの数に制限されます。DMAC を使用するためにアクティベーション ソースを有効にする必要はありません。DMAC 転送が完了すると、DMAC 割り込みが呼び出されます。アクティベーション ソース割り込みが有効になっている場合は、転送がトリガーされるのと同時に発生します。DMAC 割り込みが有効にされている場合は、すべての転送が完了した後で発生します。たとえば、長さが 16 である通常モード転送がタイマによってトリガされる場合、タイマ割り込みは各転送が発生するのと同時に発生し、DMAC 割り込みは 16 番目の転送が完了した後で発生します。DMAC ではチェーンされた転送はサポートされません。

DMAC HAL モジュールの動作に関する重要な注意事項と制限事項

ノーマル モード

通常モードでは、単一転送はアクティベーションソースイベントが発生するたびにトリガされます。 単一転送は、サイズパラメータで選択されている設定に応じて、1 バイト、2 バイト、または 4 バイトです。転送が発生するたびに、転送長が 1 だけデクリメントされます。転送長が 0 に達すると、転送は完了です。

リピート モード

リピートモードでは、アクティベーションソースイベントが発生するたびに単一転送がトリガされます。 単一転送は、サイズパラメータで選択されている設定に応じて、1 バイト、2 バイト、または 4 バイトです。転送が発生するたびに、転送長が 1 だけデクリメントされます。転送長が 0 に達すると、初期値が転送長にリロードされて、転送が再度開始します。リピートエリアが転送元に設定されている場合、転送が再度開始するときに転送元レジスタにも初期値がリロードされます。また、リピートエリアが転送先に設定されている場合は、転送が再度開始するときに転送先レジスタに初期値がリロードされます。

ブロック モード

ブロックモードでは、アクティベーションソースイベントが発生するたびに転送長全体が転送されます。たとえば、アクティベーションソースがタイマ、バイトサイズが 2、バイト長が 12 である転送がブロックモードで構成されている場合、アクティベーションイベントが発生するたびに 24 バイトが転送されます。転送が発生するたびに、転送長が 1 だけデクリメントされます。長さが 0 に達すると、初期値が転送長にリロードされて、転送が再度開始します。リピート エリアがソースに設定されている場合、転送の再開始時にソース レジスタにも初期値がリロードされます。また、リピートエリアが転送先に設定されている場合は、転送が再度開始するときに転送先レジスタに初期値がリロードされます。

アドレス モード

サイズ（1 バイト、2 バイト、または 4 バイト）が転送されるたびに、転送元ポインタは src_addr_mode で、転送先ポインタは dest_addr_mode で、それぞれ調整されます。

たとえば、src_addr_mode が TRANSFER_ADDR_MODE_INCREMENTED に設定され、サイズが TRANSFER_SIZE_4_BYTE に設定されている場合、p_dest は転送が終わるたびに 4（転送サイズ）だけインクリメントされます。

TRANSFER_ADDR_MODE_FIXED に設定されている場合、ポインタは変化しません。
このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.14.4 アプリケーションへの DMAC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに DMAC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

転送ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(転送ドライバーのデフォルト名は g_dmac0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

DMAC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ transfer0 Transfer Driver on r_dmac	Threads	New Stack > Driver > Transfer> Transfer Driver on r_dmac

次の図に示すように、r_dmac の転送ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

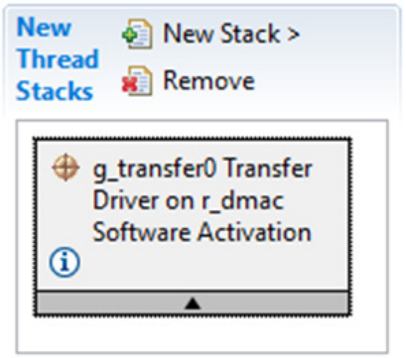


図 196:DMAC HAL モジュールのスタック

4.2.14.5 DMAC HAL モジュールの構成

ユーザーは必要な動作に合わせて DMAC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_dmac 上の DMAC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Name	g_transfer0	Module name
Channel	0	Channel selection
Mode	Normal, Repeat, Block Default: Normal	Mode selection
Transfer Size	1 Byte, 2 Bytes, 4 Bytes Default: 2 Bytes	Transfer size selection
Destination Address Mode	Fixed, Incremented, Destination Default: Fixed	Destination address mode selection
Source Address Mode	Fixed, Incremented, Destination Default: Fixed	Source address mode selection

ISDE Property	Value	Description
Repeat Area (Unused in Normal Mode	Destination, Source Default: Source	Repeat area selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source	Software Activation, Peripheral Events Default: Software Activation	Activation source selection
Auto Enable	True, False Default: True	Auto enable selection
Callback	NULL	Callback selection
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

DMAC HAL モジュールのクロック構成

DMAC 周辺モジュールは、ICLK をそのクロックソースとして使用します。ICLK の周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

DMAC HAL モジュールのピン構成

DMAC HAL モジュールはいずれのピンとも関連付けられていません。

4.2.14.6 アプリケーションでの DMAC HAL モジュールの使用

アプリケーションで DMAC HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、DMAC HAL モジュールを初期化します。
2) enable API を使用して、DMAC HAL モジュールを有効にします（自動的に有効化されない場合）。
3) 必要に応じて、他の API を使用して転送を管理します。
4) 必要に応じて、close API を使用して DMAC HAL モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

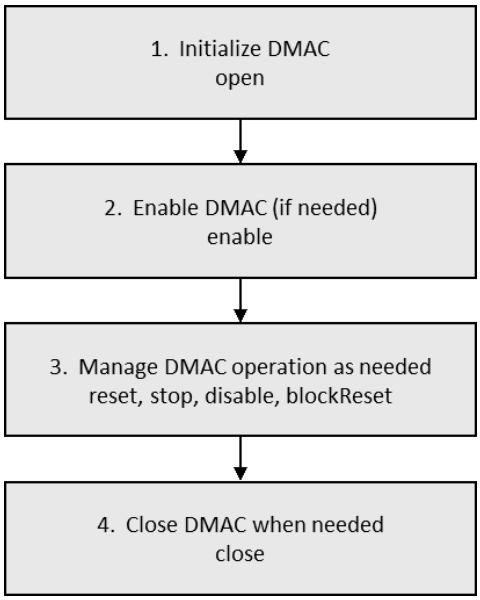


図 197:一般的な DMAC HAL モジュールアプリケーションのフロー図

4.2.15 DTC ドライバー

データトランスファコントローラ（DTC）HAL モジュールは、データ転送アプリケーション用のハイレベル API を提供し、Synergy MCU 上の DTC ペリフェラルを使用します。ユーザー定義のコールバックを作成し、転送イベント発生時に CPU に通知できます。

4.2.15.1 DTC HAL モジュールの特長

データトランスファコントローラ（DTC）HAL モジュールは、割り込みまたはイベントの発生時に、ユーザー指定の転送元からユーザー指定の転送先にデータを移動します。

- Synergy MCU の DTC モジュールをサポートします
- 必要に応じて、割り込みをサポートします。

- 複数の転送モードをサポートします
  - 単一転送
  - リピート転送
  - ブロック転送
  - アドレスインクリメントモードまたは固定モード
  - チェーン転送
- 複数のチャネルをサポートします（選択されている実装に依存）
  - チャネルの数は、RAM ベースのベクタテーブルのサイズによってのみ制限されます

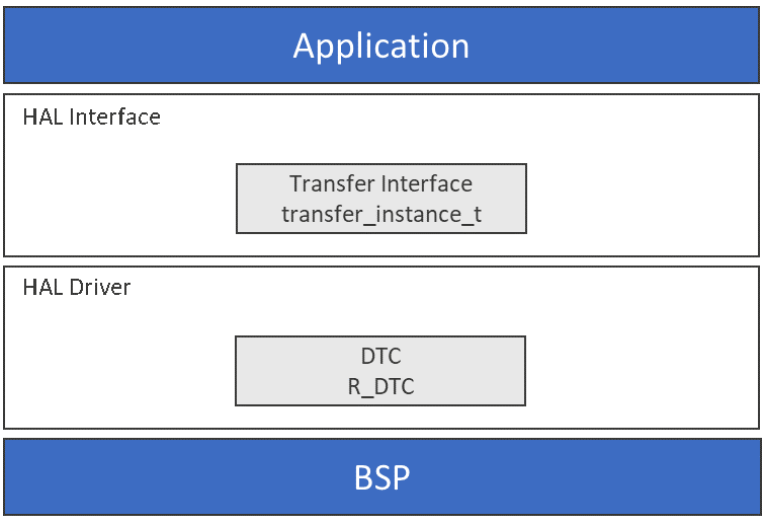


図 198:DTC HAL モジュールのブロック図

4.2.15.2 DTC HAL モジュールの API の概要

DTC HAL モジュールでは、オープン、クローズ、リセット、有効化、無効化、開始、停止のための API が定義されています。DTC と DMAC では、DTC と DMA の実装の変更を容易にするため、同じ転送インタフェースが使用されていることに注意してください。API の呼び出しは同じであり、ローレベルの実装から独立しています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

DTC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_transfer0.api->open(g_transfer0.p_ctrl, g_transfer0.p_cfg) Initial configuration. Enables the transfer if auto_enable is true and p_src, p_dest, and length are valid. Transfers can also be enabled using enable or reset.
close	g_transfer0.api->close(g_transfer0.p_ctrl) Close device channel. Turns off hardware if last channel open.
reset	g_transfer0.api->reset(g_transfer0.p_ctrl, &source, &destination, number_of_transfers) Reset source address pointer, destination address pointer, and/or length, keeping all other settings the same. Enable the transfer if p_src, p_dest, and length are valid.
start	g_transfer0.api->start(g_transfer0.p_ctrl, mode) Start transfer in software.
stop	g_transfer0.api->stop(g_transfer0.p_ctrl) Stop transfer in software. The transfer will stop after completion of the current transfer.
enable	g_transfer0.api->enable(g_transfer0.p_ctrl) Enable transfer. Transfers occur after the activation source event (or when start is called if ELC_EVENT_ELC_SOFTWARE_EVENT_0 or ELC_EVENT_ELC_SOFTWARE_EVENT_0 is chosen as activation source).

Function Name	Example API Call and Description
disable	g_transfer0.api->disable(g_transfer0.p_ctrl) Disable transfer. Transfers do not occur after the transfer_info_t::activation source event (or when start is called if ELC_EVENT_ELC_SOFTWARE_EVENT_0 or ELC_EVENT_ELC_SOFTWARE_EVENT_0 is chosen as transfer_info_t::activation_source).
versionGet	g_transfer0.api->versionGet(&version) Gets version and stores it in provided pointer version.
infoGet	g_transfer0.api->infoGet(g_transfer0.p_ctrl, &info) Provides information about this transfer.
blockReset	g_transfer0.api->blockReset(g_transfer0.p_ctrl, &source, &destination, length, size, number_of_transfers) Reset source address pointer, destination address pointer, and/or length, for block transfer keeping all other settings the same. Enable the transfer if p_src, p_dest, and length are valid.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_ASSERTION	Parameter has invalid value.

Name	Description
SSP_ERR_NOT_OPEN	The channel is not opened.
SSP_ERR_UNSUPPORTED	Operation not configured correctly.
SSP_ERR_IN_USE	The channel specified has already been opened. No configurations were changed. Call the associated Close function or use associated Control commands to reconfigure the channel.
SSP_ERR_HW_LOCKED	The DTC hardware resource is locked.
SSP_ERR_IRQ_BSP_DISABLED	IRQ not enabled in BSP.
SSP_ERR_NOT_ENABLED	Operation failed.
SSP_ERR_NOT_OPEN	The channel is not opened.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.15.3 DTC HAL モジュールの動作の概要

Synergy MCU 内でデータを移動するには、ダイレクトメモリアクセスコントローラ（DMAC）とデータトランスファコントローラ（DTC）を使用できます。どちらの実装を使用するか選択するときは、考慮事項がいくつかあります。以下の動作の概要には、アプリケーションに最適な実装を決定するのに役立つ情報が含まれます。ほとんどの汎用転送アプリケーションでは DTC モジュールが推奨されますが、基本的な転送機能にはどちらのモジュールでも使用できます。各転送モジュールのユースケースは次のとおりです。

DMAC HAL モジュールの選択

DMAC HAL モジュールは、割り込みまたはイベントの発生時に、ユーザー指定の移動元からユーザー指定の移動先にデータを移動します。DMAC HAL モジュールは DMAC 周辺レジスタを使用するため、システムでの転送回数はデバイス上の DMAC チャネルの数に制限されます。DMAC を使用するためにアクティベーション ソースを有効にする必要はありません。DMAC 転送が完了すると、DMAC 割り込みが呼び出されます。アクティベーション ソース割り込みが有効になっている場合は、転送がトリガーされるのと同時に発生します。DMAC 割り込みが有効にされている場合は、すべての転送が完了した後で発生します。たとえば、長さが 16 である通常モード転送がタイマによってトリガされる場合、タイマ割り込みは各転送が発生するのと同時に発生し、DMAC 割り込みは 16 番目の転送が完了した後で発生します。DMAC HAL モジュールでは、チェーンされた転送はサポートされません。

DTC HAL モジュールの選択

DTC HAL モジュールでは、システム内のすべての割り込みに対するスロットを持つ、RAM ベースのベクタテーブルが使用されます。DTC 転送が完了すると、アクティベーション ソース割り込みが呼び出されます。DTC を使用するには、アクティベーション ソース割り込みを有効にする必要があります。一般に、アクティベーションソース割り込みは、TRANSFER_IRQ_EACH が構成で指定されていない限り、転送が完了するまで DTC によってミュートされます。たとえば、長さが 16 である通常モード転送がタイマによってトリガされた場合、転送が有効である間、最初の 15 回のタイマ割り込みは発生しません。タイマー割り込みは 16 番目の転送の後で発生します。DTC ではチェーンされた転送も許可されます。つまり、単一のアクティベーションソース割り込みの後で複数の転送を実行できます。この機能はドライバーによってサポートされていますが、ISDE の外部で構成する必要があります。

DTC HAL モジュールの動作に関する重要な注意事項と制限事項

ノーマル モード

通常モードでは、アクティベーションソースイベントが発生するたびに単一の転送がトリガされます。単一転送は、サイズパラメータで選択されている設定に応じて、1 バイト、2 バイト、または 4 バイトです。転送が発生するたびに、転送の長さが 1 だけデクリメントされます。転送の長さが 0 に達すると、転送は完了です。

リピート モード

リピートモードでは、アクティベーションソースイベントが発生するたびに単一の転送がトリガされます。単一転送は、サイズパラメータで選択されている設定に応じて、1 バイト、2 バイト、または 4 バイトです。転送が発生するたびに、転送長が 1 だけデクリメントされます。転送長が 0 に達すると、初期値が転送長にリロードされて、転送が再度開始します。リピート エリアがソースに設定されている場合、転送の再開始時にソース レジスタにも初期値がリロードされます。また、リピート エリアが宛先に設定されている場合、転送の再開始時に宛先レジスタに初期値がリロードされます。

ブロック モード

ブロックモードでは、アクティベーションソースイベントが発生するたびに転送長全体が転送されます。たとえば、アクティベーションソースがタイマ、バイトサイズが 2、バイト長が 12 である転送がブロックモードで構成されている場合、アクティベーションイベントが発生するたびに 24 バイトが転送されます。転送が発生するたびに、転送長が 1 だけデクリメントされます。転送長が 0 に達すると、初期値が転送長にリロードされて、転送が再度開始します。リピート エリアがソースに設定されている場合、転送の再開始時にソース レジスタにも初期値がリロードされます。また、リピート エリアが宛先に設定されている場合、転送の再開始時に宛先レジスタに初期値がリロードされます。

アドレス モード

サイズ（1 バイト、2 バイト、または 4 バイト）が転送されるたびに、転送元ポインタと転送先ポインタはそれぞれ Source Address Mode と Destination Address Mode の構成設定に基づいて調整されます。たとえば、Source Address Mode が TRANSFER_ADDR_MODE_INCREMENTED に設定され、サイズが TRANSFER_SIZE_4_BYTES, に設定されている場合、転送先ポインタは転送が終わるたびに 4（転送サイズ）だけインクリメントされます。構成設定が TRANSFER_ADDR_MODE_FIXED. の場合、ポインタは変化しません。

チェーンされた転送

チェーンされた転送は DTC でのみサポートされています。チェーンされた転送を使用するには、transfer_info_t 構造体の配列を作成します。最後の転送を除くすべての転送について、transfer_chain_mode_t を TRANSFER_CHAIN_MODE_EACH または TRANSFER_CHAIN_MODE_END に設定します。最後の転送は、TRANSFER_CHAIN_MODE_DISABLED. にする必要があります。

p_info を、transfer_info_t 構造体用の配列内にある最初の構造体のベースに設定します。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.15.4 アプリケーションへの DTC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに DTC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

転送ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（転送ドライバーのデフォルト名は g_transfer0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

DTC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ transfer0 DTC Driver on r_dtc	Threads > HAL/Common	New Stack > Driver > Transfer > Transfer Driver on r_dtc

次の図に示すように、r_dtc の転送ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

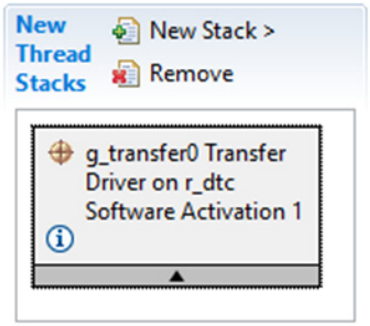


図 199:DTC HAL モジュールのスタック

4.2.15.5 DTC HAL モジュールの構成

ユーザーは必要な動作に合わせて DTC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_dtc 上の DTC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection.
Name	g_transfer0	Module name
Mode	Normal, Repeat, Block Default: Normal	Mode selection
Transfer Size	2 Bytes	Transfer size selection
Destination Address Mode	Fixed, Incremented, Destination Default: Fixed	Destination address mode selection
Source Address Mode	Fixed, Incremented, Destination Default: Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination, Source Default: Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection

ISDE Property	Value	Description
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events Default: Software Activation 1	Activation source selection
Auto Enable	True, False Default: True	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

DTC HAL モジュールのクロック構成

DTC 周辺モジュールは、ICLK をクロックソースとして使用します。ICLK の周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clocks] タブを使用するか、実行時に CGC インタフェースを使用します。

DTC HAL モジュールのピン構成

DTC はいずれのピンとも関連付けられていません。

4.2.15.6 アプリケーションでの DTC HAL モジュールの使用

アプリケーションで DTC HAL モジュールを使用する一般的な手順は次のとおりです。

1) transfer_api_t::open API を使用して DTC を初期化します。
2) transfer_api_t::enable API を使用して DTC を有効にします（自動的に有効化されない場合）。
3) 必要に応じて他の API を使用して転送を管理します。

4) 必要に応じて transfer_api_t::close を使用して DTC を閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

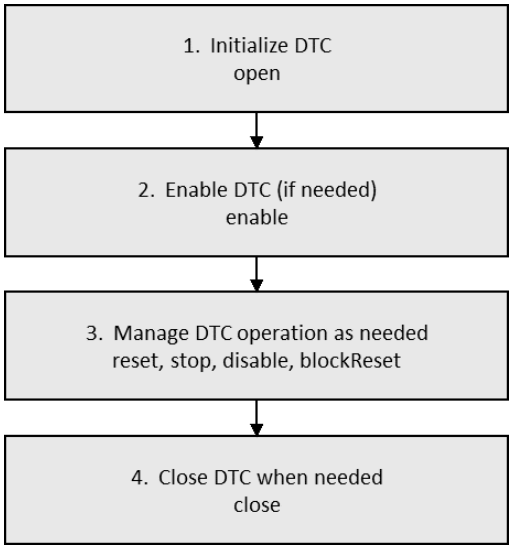


図 200:一般的な DTC HAL モジュールアプリケーションのフロー図

4.2.16 ELC ドライバー

イベントリンクコントローラ（ELC）HAL モジュールは、自律動作のためにさまざまな MCU ペリフェラルを接続するハイレベルの API を提供し、Synergy MCU 上の ELC ペリフェラルを使用します。ELC HAL モジュールに関連付けられたコールバックはありません。e^2 studio 統合ソリューション開発環境（ISDE）のプロジェクトコンフィギュレータには、ELC HAL モジュールがデフォルトですべてのプロジェクトに含まれています。ELC HAL モジュールを構成するには、[Threads] タブの [HAL/Common module] で ELC HAL モジュールを選択し、次に [HAL/Common Stacks] ウィンドウで ELC HAL モジュールをクリックします。

4.2.16.1 ELC HAL モジュールの特長

ELC HAL モジュールは、次の機能を実行できます。

- 2 つのブロック間にイベント リンクを作成します。
- その 2 つのブロック間のイベント リンクを切断します。
- CPU に割り込む 2 つのソフトウェアイベントのうち 1 つを生成します。

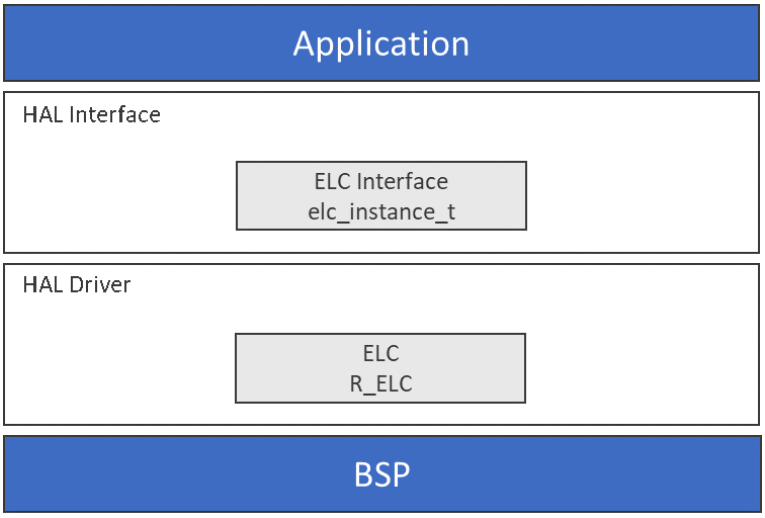


図 201:**ELC HAL モジュールのブロック図**

4.2.16.2 ELC HAL モジュールの API の概要

ELC HAL モジュールでは、モジュール間のイベントリンクの初期化、有効化、無効化、作成、切断のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

ELC HAL モジュールの API の要約

Function Name	Example API Call and Description
init	g_elc.p_api->init(g_elc.p_cfg) Initialize all links in the Event Link Controller.
softwareEventGenerate	g_elc.p_api->softwareEventGenerate(event_num) Generate a software event in the Event Link Controller.
linkSet	g_elc.p_api->linkSet(peripheral, signal) Create a single event link.

Function Name	Example API Call and Description
linkBreak	g_elc.p_api->linkBreak(peripheral) Break an event link.
enable	g_elc.p_api->enable() Enable the operation of the Event Link Controller.
disable	g_elc.p_api->disable() Disable the operation of the Event Link Controller.
versionGet	g_elc.p_api->versionGet(&version) Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successfully completed.
SSP_ERR_ASSERTION	p_version is NULL.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.16.3 ELC HAL モジュールの動作の概要

ELC HAL モジュールを使用すると、開発者は、ある Synergy MCU ペリフェラルによって生成されたイベントを使用して別の Synergy MCU ペリフェラルの動作の開始をトリガすることで、さまざまなペリフェラルの動作をリンクするイベントを作成できます。ELC HAL モジュールの API を使用すると、2 つのブロック間のリンクを簡単に作成できます（たとえば、周期的なスキャンのインターバルを制御するためのタイマから ADC へのリンク）。このようにしてさまざまな周辺機器を接続することにより、CPU による介入がほとんど、またはまったく必要のない、インテリジェントな機能を構築できます。

次の図は ELC の簡単なブロック図であり、入力イベントソースと、イベントによってトリガできる周辺機器が示されています。入力トリガと出力トリガの数は S7G2 MCU に固有です。他の Synergy デバイスは、異なる数のイベントをサポートします。

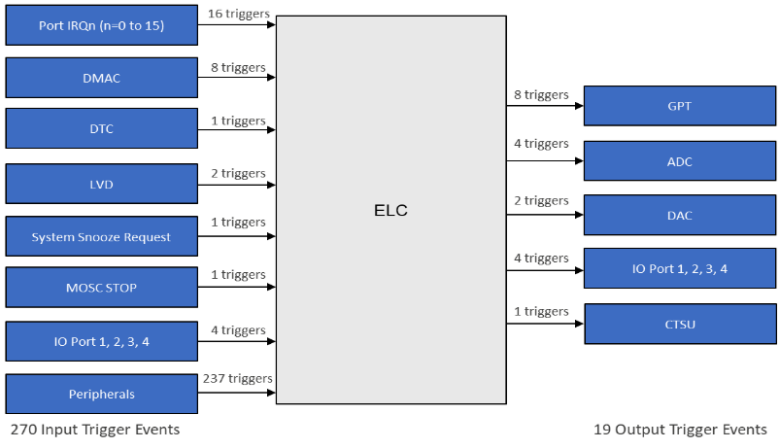


図 202:ELC HAL モジュールのトリガイベント

bsp_elc.h, ファイルの Synergy によって生成されたコードでは、ELC ペリフェラルのマッピングを確認できます。S7G2 MCU で使用可能なペリフェラルおよび関連イベントを次の図に示します。関連する『MCU ユーザーズマニュアル』では、ELC の動作についての追加情報もわかります。

```
/** Possible peripherals to be linked to event signals */
typedef enum e_elc_peripheral
{
    ELC_PERIPHERAL_GPT_A                                    =   (0),
    ELC_PERIPHERAL_GPT_B                                    =   (1),
    ELC_PERIPHERAL_GPT_C                                    =   (2),
    ELC_PERIPHERAL_GPT_D                                    =   (3),
    ELC_PERIPHERAL_GPT_E                                    =   (4),
    ELC_PERIPHERAL_GPT_F                                    =   (5),
    ELC_PERIPHERAL_GPT_G                                    =   (6),
    ELC_PERIPHERAL_GPT_H                                    =   (7),
    ELC_PERIPHERAL_ADC0                                     =   (8),
    ELC_PERIPHERAL_ADC0_B                                   =   (9),
    ELC_PERIPHERAL_ADC1                                     =  (10),
    ELC_PERIPHERAL_ADC1_B                                   =  (11),
    ELC_PERIPHERAL_DAC0                                     =  (12),
    ELC_PERIPHERAL_DAC1                                     =  (13),
    ELC_PERIPHERAL_IOPORT1                                  =  (14),
    ELC_PERIPHERAL_IOPORT2                                  =  (15),
    ELC_PERIPHERAL_IOPORT3                                  =  (16),
    ELC_PERIPHERAL_IOPORT4                                  =  (17),
    ELC_PERIPHERAL_CTSU                                     =  (18),
} elc_peripheral_t;


/** Sources of event signals to be linked to other peripherals or the CPU1
 * @note This list may change based on device. This list is for S7G2.
 * */
typedef enum e_elc_event
{
    ELC_EVENT_ICU_IRQ0                                      =   (1),
    ELC_EVENT_ICU_IRQ1                                      =   (2),
    ELC_EVENT_ICU_IRQ2                                      =   (3),
    ELC_EVENT_ICU_IRQ3                                      =   (4),
.
.
.
    ELC_EVENT_ICU_IRQ12                                     =  (13),
    ELC_EVENT_ICU_IRQ13                                     =  (14),
    ELC_EVENT_ICU_IRQ14                                     =  (15),
    ELC_EVENT_ICU_IRQ15                                     =  (16),
.
.
.
    ELC_EVENT_GLCDC_LINE_DETECT                             = (506),
    ELC_EVENT_GLCDC_UNDERFLOW_1                             = (507),
    ELC_EVENT_GLCDC_UNDERFLOW_2                             = (508),
    ELC_EVENT_DRW_INT                                       = (509),
    ELC_EVENT_JPEG_JEDI                                     = (510),
    ELC_EVENT_JPEG_JDTI                                     = (511),
} elc_event_t;
```

図 203:**ELC HAL のペリフェラルマップ**

ELC HAL モジュールの動作に関する重要な注意事項と制限事項

ELC HAL モジュールでは、ピン、クロック、または割り込みを構成する必要はありません。ELC HAL モジュールは周辺機能の間の " 接続 " メカニズムに過ぎません。ただし、ELC を介して I/O ポートをリンクする場合は、I/O ピンを入力または出力として構成する必要があります。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.16.4 アプリケーションへの ELC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに ELC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

ELC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(ELC ドライバーのデフォルト名は g_elc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

ELC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_elc ELC Driver on r_elc	Threads	New Stack> Driver> System> ELC Driver on r_elc

次の図に示すように、r_elc の ELC ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

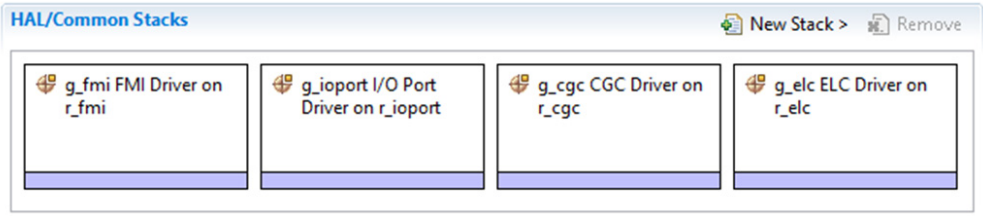


図 204:ELC HAL モジュールのスタック

4.2.16.5 ELC HAL モジュールの構成

ユーザーは必要な動作に合わせて ELC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_elc 上の ELC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_elc	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ELC HAL モジュールのクロック構成

ELC HAL モジュールには、特定のクロック構成は必要ありません。

ELC HAL モジュールのピン構成

ELC HAL モジュールに直接関連付けられている、構成の必要なピンはありません。

4.2.16.6 アプリケーションでの ELC HAL モジュールの使用

アプリケーションで ELC HAL モジュールを使用する一般的な手順は次のとおりです。

1) init および enable API を使用して、ELC を初期化します（ISDE によって自動的に行われます）。
2) linkSet API を使用して、ペリフェラルとイベントをリンクします。
3) enable API を使用して、リンクを有効にします。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

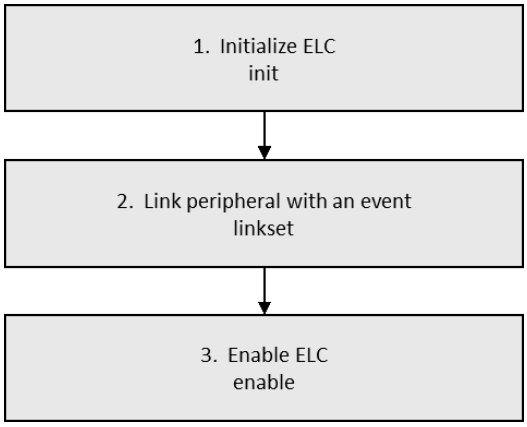


図 205:一般的な ELC HAL モジュールアプリケーションのフロー図

4.2.17 外部 IRQ ドライバー

外部 IRQ HAL モジュールでは、Synergy MCU 上の外部 IRQ ピンを構成し、使用するための API が提供されます。外部 IRQ HAL モジュールは、Synergy MCU の割り込みコントローラユニット（ICU）を使用します。

4.2.17.1 外部 IRQ HAL モジュールの特長

- 対象の Synergy MCU で使用可能な外部割り込み要求ピンをサポートします
- 複数の機能オプションをサポートします
  - 割り込みの生成の有効化と無効化
  - IRQ ノイズフィルタの有効化と無効化
  - 外部ピン IRQ トリガの設定（IRQ ピンでの立ち上がりエッジ、立ち下がりエッジ、ローレベル）
- ユーザーコールバック関数の構成をサポートします。この関数は、外部ピン割り込みが生成されると、HAL モジュールによって呼び出されます。

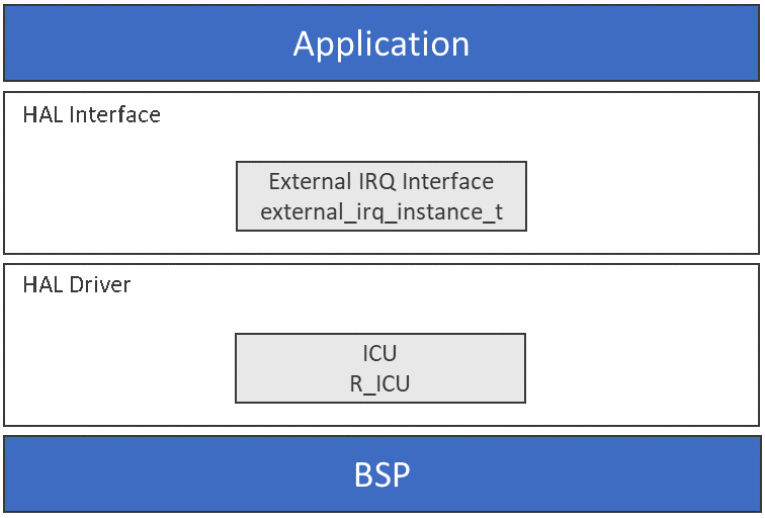


図 206:外部 IRQ HAL モジュールのブロック図

4.2.17.2 外部 IRQ HAL モジュールの API の概要

外部 IRQ HAL モジュールでは、オープン、クローズ、および外部ピンからの割り込みイベントの待機のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

外部 IRQ HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_external_irq.p_api->open(g_external_irq.p_ctrl, g_external_irq.p_cfg) Open instance and initialize.
enable	g_external_irq.p_api->enable(g_external_irq.p_ctrl) Enable callback when IRQ occurs.
disable	g_external_irq.p_api->disable(g_external_irq.p_ctrl) Disable callback when IRQ occurs.

Function Name	Example API Call and Description
triggerSet	g_external_irq.p_api->triggerSet(g_external_irq.p_ctrl, trigger) Set trigger.
filterEnable	g_external_irq.p_api->filterEnable(g_external_irq.p_ctrl) Enable noise filter.
filterDisable	g_external_irq.p_api->filterDisable(g_external_irq.p_ctrl) Disable noise filter.
close	g_external_irq.p_api->close(g_external_irq.p_ctrl); Close instance.
versionGet	g_external_irq.p_api->wait(&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successful.
SSP_ERR_ASSERTION	Assertion error.
SSP_ERR_INVALID_ARGUMENT	Callback is not NULL but ISR is not enabled.
SSP_ERR_IN_USE	Device in use.
SSP_ERR_NOT_OPEN	Device unopened.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.17.3 外部 IRQ HAL モジュールの動作の概要

外部 IRQ HAL モジュールでは、外部割り込み要求を制御するための API 関数のセットが提供されます。割り込みは、外部 IRQ ピンでの入力信号の立ち上がりエッジ、立ち下がりエッジ、両方のエッジ、またはローレベルでトリガできます。デジタルフィルタ機能を有効にして、入力信号のノイズをある程度除去できます。ユーザーコールバック関数がサポートされており、IRQ イベントが発生するたびにトリガされます。

構成されている外部 IRQ イベントが発生したときに、DMAC または DTC 周辺機能を使用してデータの転送をトリガするには、アクティベーションソースを ELC_EVENT_PORTn_IRQ（n は IRQ チャネル番号）に設定して、DMAC または DTC 転送を構成します。

イベントリンクコントローラ（ELC）を使用して、外部割り込み要求から他の周辺機能をトリガして開始できます。ELC HAL モジュールの詳細については、『SSP ユーザーズマニュアル』ユーザーガイドを参照してください。

外部 IRQ HAL モジュールの動作に関する重要な注意事項と制限事項

- 外部割り込み要求機能をサポートするポートピンを調べるときや、特定のポートピンの外部 IRQ 番号が知りたいときは、対象の Synergy デバイスのデータシートを参照してください。
- 外部 IRQ 番号は、ISDE の [Properties] ウィンドウでの外部 IRQ HAL モジュールに対するチャネル設定に対応します。
- 予想されるハードウェアイベントが発生したことをモジュールに通知するには、PORTn（n は IRQ 番号）の割り込みを BSP で有効にする必要があります。
- ユーザーコールバック関数を open API に登録できます。このコールバック関数が指定されている場合、IRQn がトリガされるたびに割り込みサービス ルーチン (ISR) から呼び出されます。注 : コールバックは ISR から呼び出されるため、ブロッキング呼び出しを使用したり、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に悪影響を及ぼす可能性があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.17.4 アプリケーションへの外部 IRQ HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに外部 IRQ HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

外部 IRQ ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(外部 IRQ ドライバーのデフォルト名は g_icu0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

外部 IRQ HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
r_icu0 External IRQ Driver on r_icu	Threads	New Stack> Driver> Input> External IRQ Driver on r_icu

次の図に示すように r_icu の外部 IRQ ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

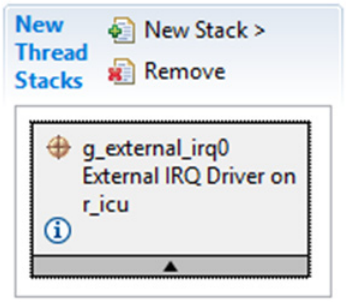


図 207:外部 IRQ HAL モジュールのスタック

4.2.17.5 外部 IRQ HAL モジュールの構成

ユーザーは必要な動作に合わせて外部 IRQ HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_icu 上の外部 IRQ HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter checking setting enables or disables the addition of parameter checking code.
Name	g_external_irq0	Module name.
Channel	0	Specifies the hardware IRQ channel used.
Trigger	Falling, Rising, Both Edges, Low Level Default: Rising	Selection for trigger event mode
Digital Filtering	Enabled, Disabled Default: Disabled	Digital filter enable/disable.
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCLK/1, PLCK/8, PLCK/32, PCLK/64 Default: PCKL/64	Sets noise filter sampling period.
Interrupt enabled after initialization	True, False Default: True	Determines if the interrupt is enabled immediately after initialization.

ISDE Property	Value	Description
Callback	NULL	A user callback function can be registered in . If this callback function is provided, it is called from the interrupt service routine (ISR) each time the IRQn triggers. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Pin Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

外部 IRQ HAL モジュールのクロック構成

IRQ 周辺モジュールには、特定のクロック設定は必要ありません。

外部 IRQ HAL モジュールのピン構成

外部 IRQ 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では IRQ ピンの選択例を示します。

r_icu 上の外部 IRQ HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
IRQ	Pins	Select Peripherals > Input: IRQ > IRQ0

注 : この選択シーケンスでは、IRQ0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_icu 上の外部 IRQ HAL モジュールのピン構成設定

Property	Value	Description
Operation Mode	Disabled, Enabled Default: Disabled	Select Enabled to enable interrupts
NMI	None, P200 Default: None	Non-maskable interrupt Pin
IRQ00:14	None, Pnn, Pmm Default: None	Interrupt request Pin

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

4.2.17.6 アプリケーションでの外部 IRQ HAL モジュールの使用

アプリケーションで外部 IRQ HAL モジュールを使用するときの一般的な手順は次のとおりです。

1) open API を使用して、外部 IRQ HAL モジュールを初期化します。
2) enable API を使用して、IRQ を有効にします（必要な場合）。
3) filterEnable API を使用して、ノイズフィルタを有効にします（必要な場合）。
4) triggerSet API を使用して、トリガ条件を変更します（誤ったイベントを避けるため、それ以前にモジュールが閉じられている場合のみ）。
5) filterDisable API を使用して、ノイズフィルタを無効化します（有効になっている場合）。
6) close API を使用して、モジュールを閉じます（必要な場合）。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

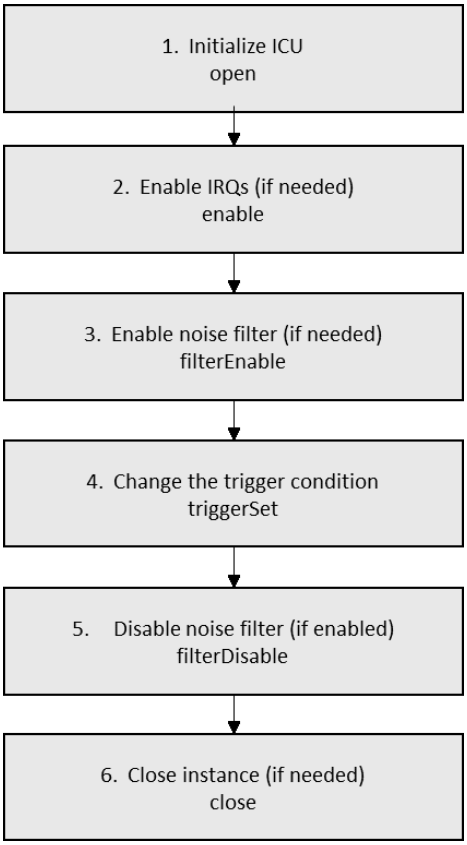


図 208:一般的な外部 IRQ HAL モジュールアプリケーションのフロー図

4.2.18 フラッシュ ドライバー

r_flash_lp と r_flash_hp. という 2 つの独立したフラッシュモジュールがあります。これらのモジュールは、フラッシュメモリプログラミングアプリケーション用のハイレベル API を実装します。高性能フラッシュモジュール（Flash_HP）は、MCU の S7 ファミリおよび S5 ファミリのプログラミングに使用されます。ローパワーフラッシュモジュール（Flash_LP）は、MCU の S3 ファミリおよび S1 ファミリのプログラミングに使用されます。この 2 つを相互に交換することはできませんが、これらのモジュールの API および他の機能はよく似ています。

4.2.18.1 フラッシュ HAL モジュールの特長

フラッシュ HAL モジュールの API により、アプリケーションは MCU 内に存在するデータと ROM 両方のフラッシュエリアのリード、ライト、消去を行うことができます。使用可能なフラッシュメモリの量は MCU パーツごとに異なりますが、API 関数はすべてのデバイスに適用されます。フラッシュ HAL モジュールの主な特長を以下に示します。

- データフラッシュのブロッキングおよび非ブロッキング消去、リード、ライト、およびブランクチェックのサポート。
- コードフラッシュのブロッキング消去、リード、ライト、およびブランクチェックのサポート。
- 非ブロッキングデータフラッシュ操作の完了時のコールバック関数のサポート。

- コードフラッシュの指定したエリアだけを消去またはライトできるようにする、ROM フラッシュのアクセスウィンドウ（ライト保護）のサポート。
- スタートアッププログラムを最初に消去することなく安全にリライトできるようにする、ブートブロックスワッピングのサポート。

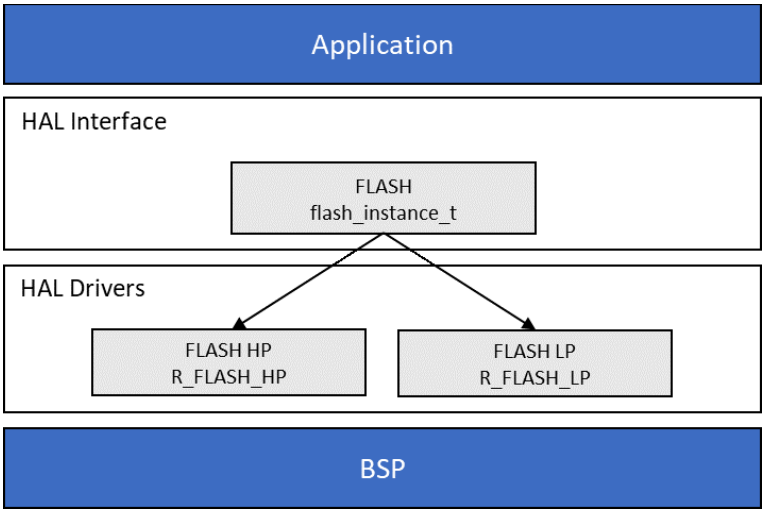


図 209:フラッシュ HAL モジュールのブロック図

4.2.18.2 フラッシュ HAL モジュールの API の概要

フラッシュ HAL モジュールでは、フラッシュメモリのオープン、リード、消去、クローズなどの操作のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

フラッシュ HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_flash0.p_api->open(g_flash0.p_ctrl, g_flash0.p_cfg); Open FLASH device.
write	g_flash0.p_api->write(g_flash0.p_ctrl,(uint32_t) write_buffer, FLASH_CF_32KB_BLOCK55, CODE_BLOCK_SIZE_32KB); Write FLASH device.

Function Name	Example API Call and Description
read	g_flash0.p_api->read(g_flash0.p_ctrl, read_buffer, DATA_FLASH_ADDR, num_bytes); Read FLASH device.
erase	g_flash0.p_api->erase(g_flash0.p_ctrl, FLASH_CF_32KB_BLOCK55,num_sectors) ; Erase FLASH device.
blankCheck	g_flash0.p_api->blankCheck(g_flash0.p_ctrl , FLASH_CF_32KB_BLOCK55, FLASH_DATA_BLOCK_SIZE, &blankCheck); Blank check FLASH device.
close	g_flash0.p_api->close(g_flash0.p_ctrl); Close FLASH device.
statusGet	g_flash0.p_api->statusGet(g_flash0.p_ctrl); Get Status for FLASH device.
accessWindowSet	g_flash0.p_api->accessWindowSet(g_flash 0.p_ctrl, FLASH_CF_32KB_BLOCK1, FLASH_CF_32KB_BLOCK3); Set Access Window for FLASH device.
accessWindowClear	g_flash0.p_api->accessWindowClear(g_flas h0.p_ctrl); Clear any existing Code Flashcode-flash access window for FLASH device.

Function Name	Example API Call and Description
reset	g_flash0.p_api->reset(g_flash0.p_ctrl); Reset function for FLASH device.
updateFlashClockFreq	g_flash0.p_api-> updateFlashClockFreq (g_flash0.p_ctrl); Update Flash clock frequency (FCLK) and recalculate timeout values.
startupAreaSelect	g_flash0.p_api->startupAreaSelect(g_flash0 .p_ctrl, FLASH_STARTUP_AREA_BLOCK1); Select which block - Default (Block 0) or Alternate (Block 1) is used as the start-up area block. Refer to the following table for all the possible values for parameter2.
versionGet	g_flash0.p_api->versionGet(&version); Retrieve the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

startupAreaSelect のパラメータ 2 のオプション

Swap Type	Is_temporary	Operation
FLASH_STARTUP_AREA_ BLOCK0	False	On next reset, Startup area will be Block 0.
FLASH_STARTUP_AREA_ BLOCK0	False	On next reset, Startup area will be Block 0.
FLASH_STARTUP_AREA_ BLOCK1	False	On next reset, Startup area will be Block 1.

Swap Type	Is_temporary	Operation
FLASH_STARTUP_AREA_BLOCK1	True	Startup area is immediately, but temporarily switched to Block 1.
FLASH_STARTUP_AREA_BTFLG	True	Startup area is immediately, but temporarily switched to the Block determined by the Configuration BTFLG.

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successful.
SSP_ERR_IN_USE	Device in use error.
SSP_FLASH_ERR_FAILURE	Flash failure error.
SSP_ERR_FCLK	FCLK must be a minimum of 4 MHz for Flash operations.
SSP_ERR_TIMEOUT	Timeout error.
SSP_ERR_INVALID_SIZE	Invalid size error.
SSP_ERR_INVALID_ADDRESS	Invalid address error.
SSP_ERR_ASSERTION	Assertion error.
SSP_ERR_INVALID_BLOCKS	Invalid number of blocks specified.
SSP_ERR_INVALID_ARGUMENT	Invalid argument error.
SSP_ERR_HW_LOCKED	Peripheral already in use.
SSP_ERR_CMD_LOCKED	FCU is in locked state, typically as a result of attempting to Erase an area that is protected by an Access Window.
SSP_ERR_NOT_OPEN	Flash has not yet been opened.
SSP_ERR_IRQ_BSP_DISABLED	Caller is requesting BGO (background mode operation) but the Flash interrupt is not enabled.

Name	Description
SSP_ERR_WRITE_FAILED	Write operation failed. This may be returned if the requested Flash area is not blank.
SSP_ERR_PE_FAILURE	Failed to enter P/E mode
false	Supplied address is valid flash address on this MCU.
true	Supplied address is valid and p_block info contains the details on this address's block.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.18.3 フラッシュ HAL モジュールの動作の概要

フラッシュ API により、オンチップ フラッシュ エリアのプログラミングと消去のプロセスをより容易に実行できます。コード（ユーザー ROM）フラッシュエリアとデータフラッシュエリアの両方がサポートされています。この API の最も単純な形式を使用して、ブロッキング消去およびプログラム操作を実行できます。" ブロッキング " という用語は、プログラムまたは消去関数が呼び出されたときに、操作が完了するまで関数が戻らないことを意味します。この API はコードとデータフラッシュの両方のブロッキングをサポートしていますが、非ブロッキング操作（BGO、バックグラウンド操作）はデータフラッシュ操作でのみ使用できます。コードフラッシュ操作が実行中のとき、ユーザーがコードフラッシュエリアにアクセスすることはできません。コードフラッシュ操作が進行中のときにコードフラッシュ エリアにアクセスしようとすると、フラッシュコントロールユニットがエラー状態に遷移します。

コードフラッシュ操作はブロッキングですが、一部の状況では操作がブロッキングのときでもコードフラッシュにアクセスできる場合があり、そのようなときにアクセスしてはならないということを憶えておくことが重要です。次のような場合です。

- ベクタテーブルが ROM に存在する場合のベクタテーブルアクセス。
- ベクタテーブル自体は ROM に存在しない場合であっても、ROM アドレスをベクタが示している割り込みによる ROM アクセス。コードフラッシュ操作がブロッキング中に複数のスレッドが実行継続を許可されているマルチスレッドアプリケーション。

フラッシュ HAL モジュールの動作に関する重要な注意事項と制限事項

データフラッシュ BGO の使用上の注意

データフラッシュ BGO を使用しているときは、ユーザー ROM、RAM、および外部メモリに引き続きアクセスできます。データフラッシュ操作の間に、データフラッシュにアクセスしないようにする必要があります。これには、データフラッシュにアクセスする可能性のある割り込みが含まれます。

コードフラッシュの使用上の注意

BGO モードはコードフラッシュでサポートされていないため、操作が完了する前にコードフラッシュ操作から戻ることはありません。デフォルトでは、ベクタテーブルはユーザー ROM（コードフラッシュ）内に存在します。ROM 操作中に割り込みが発生すると、割り込みの開始アドレスをフェッチするために ROM へのアクセスが行われ、エラーが発生します。

最も単純な回避策は、コードフラッシュ操作中の割り込みを無効化することです。別のオプションは、ベクタテーブルを RAM にコピーし、それに応じて VTOR（ベクタテーブルオフセットレジスタ）を更新して、割り込みサービスルーチンが RAM 外で実行されるようにすることです。同様に、マルチスレッド環境では、コードフラッシュ操作の進行中に、ROM から実行するスレッドがアクティブになることができないようにする必要があります。

ブランク チェック

blankCheck API 関数は、コードまたはデータフラッシュの内容がブランクかどうかをチェックします。フラッシュ（コードまたはデータ）には、最初に消去しない限り書き込みができないことに注意してください。blankCheck 関数

は、指定されたエリアがブランクで書き込み可能かどうかを判定します。ほとんどすべての場合、フラッシュの内容を 0xFF と比較するだけでは、エリアがブランクかどうかを判定するのに十分ではありません。唯一の例外はフラッシュ HP コードフラッシュです。Flash_HP コードフラッシュの 0xFF は、確実にブランクを示します。すべての場合において、blankCheck API 関数を使用することを強くお勧めします。

フラッシュ ステータス

statusGet API 関数を使用すると、アプリケーションがフラッシュの「準備完了」状態をクエリできるようになります。これは、ユーザーがコールバック関数を使用しないことを選択したためにデータフラッシュ操作の完了が非同期的に通知されないデータフラッシュ BGO 操作で有用です。この場合、データフラッシュは BGO モードで動作するように構成されているため、操作（たとえば消去）が開始されると、呼び出しは即座に戻り、操作はバックグラウンドで実行されます。statusGet API 関数を呼び出すことにより、ユーザーは、操作が安全に完了したかエラーを生成し、別のフラッシュ操作に安全に移行できるようになったことを判定できます。

ブロックのスワップ

startupAreaSelect API 関数を使用すると、スタートアップエリアブロックとして使用されるブロック（デフォルト（ブロック 0）または代替（ブロック 1））を選択できます。提供されているパラメータは、どのブロックがアクティブスタートアップブロックになり、そのアクションが次のリセットの後すぐに（ただし一時的に）実行されるのか、恒常的に実行されるのかを決定します。

一時的な切り替えには限定的なメリットしかないように見えますが、ブロック 0 が書き込み保護となるようなアクセスウィンドウがある場合、アクセスウィンドウに触らずに、一時的な切り替えを行い、ブロックを更新して、元に切り替えることができます。

フラッシュ クロック (FCLK)

FCLK は、すべてのフラッシュ操作の実行においてフラッシュ周辺機能により使用されるクロックです。> フラッシュ操作が成功するためには、4MHz 以上でなければなりません。open 関数の中でフラッシュクロックが確認されて、4MHz 未満の場合、<open は SSP_ERR_FCLK. を返します。フラッシュ API が開かれた後で FCLK の周波数を変更する場合は、updateFlashClockFreq API 関数を呼び出して API に変更を通知する必要があります。そうしないと、フラッシュ操作が失敗し、パーツが損傷する可能性があります。

割り込み

フラッシュ対応割り込みを有効にする必要があるのは、データフラッシュ BGO を使用する予定のある場合だけです。このモードでは、アプリケーションはデータフラッシュ操作を開始した後、ユーザー提供のコールバック関数を使用してその完了またはエラーの通知を非同期的に受けることができます。コールバック関数には、コールバックイベント（つまり、FLASH_EVENT_ERASE_COMPLETE）のソースを示すイベント情報を含む構造体が渡されます。

FLASH FRDYI 割り込みが有効になっている場合、対応する ISR がフラッシュドライバーで定義されます。ISR は、ユーザーコールバック関数が open API で登録されている場合は、それをコールします。

注 : フラッシュ HP は追加のフラッシュエラー割り込みをサポートしており、フラッシュ HP で BGO モードが有効になっている場合は、フラッシュ対応割り込みとフラッシュエラー割り込みの両方を有効にする（プライオリティを割り当てる）必要があります。

アクセスウィンドウ

アクセス ウィンドウは、コード フラッシュ内でプログラミング / 消去が有効な連続した領域を定義します。この領域はブロック境界上にあり、開始アドレスと終了アドレスが accessWindowSet に提供されます。開始アドレスを含むブロックが最初のブロックです。エンドアドレスを含むブロックが最後のブロックです。したがって、アクセスウィンドウは、先頭のブロックから最後のブロックまでとなります（先頭のブロックと最後のブロックを含む）。この範囲の外側はすべて、ライト保護されます。無効なアドレス情報が accessWindowSet に提供されると、SSP_ERR_INVALID_ADDRESS. が返されます。accessWindowClear API 関数をコールすることにより、アクセスウィンドウを削除できます。

- 高性能フラッシュモジュール (Flash_HP) は、MCU の S7 ファミリおよび S5 ファミリのプログラミングに使用される API です。
- ローパワーフラッシュモジュール (Flash_LP) は、MCU の S3 ファミリおよび S1 ファミリのプログラミングに使用される API です。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.18.4 アプリケーションへのフラッシュ HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにフラッシュ HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

フラッシュドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(フラッシュドライバーのデフォルト名は g_flash0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

フラッシュ HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_flash0 Flash Driver on r_rflash_hp	Threads	New Stack > Driver > Storage > Flash Driver on r_flash_hp
g_flash0 Flash Driver on r_rflash_lp	Threads	New Stack > Driver > Storage > Flash Driver on r_flash_lp

次の図に示すように、r_flash_hp または r_flash_lp のフラッシュドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

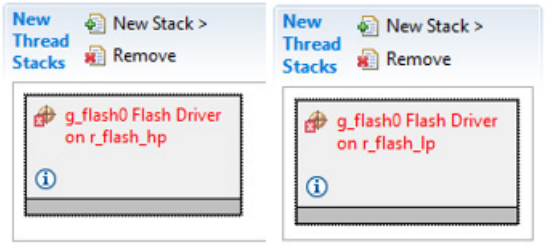


図 210:フラッシュ HAL モジュールのスタック

4.2.18.5 フラッシュ HAL モジュールの構成

ユーザーは必要な動作に合わせてフラッシュ HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_flash_hp 上のフラッシュ HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Controls whether to include code for API parameter checking.
Code Flash Programming Enable	Enable, Disabled Default: Disabled	Controls whether or not Code Flash programming is enabled. Disabling reduces the amount of ROM used by the API.
Name	g_flash0	Module name.
Data Flash Background Operation	Enabled, Disabled Default: Enabled	Enabling allows Flash API calls that reference Data Flash to return immediately, with the operation continuing in the background.

ISDE Property	Value	Description
Callback	NULL	Callback function called when a Data Flash BGO operation completes or errors. A user callback function can be registered in open. Warning: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Flash Ready Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Flash ready interrupt priority selection.
Flash Error Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Flash error interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_flash_lp 上のフラッシュ HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Controls whether to include code for API parameter checking.

ISDE Property	Value	Description
Code Flash Programming Enable	Enable, Disabled Default: Disabled	Controls whether or not Code Flash programming is enabled. Disabling reduces the amount of ROM used by the API.
Name	g_flash0	Module name.
Data Flash Background Operation	Enabled, Disabled Default: Enabled	Enabling allows Flash API calls that reference Data Flash to return immediately, with the operation continuing in the background.
Callback	NULL	Callback function called when a Data Flash BGO operation completes or errors. A user callback function can be registered in open. Warning: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Flash Ready Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Flash ready interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

デフォルト以外の設定が望ましい場合もあります。たとえば、コードフラッシュプログラミングを無効にすると、ドライバーのコードサイズを減らすのに役立つことがあります。

フラッシュ HAL モジュールのクロック構成

フラッシュ対応割り込みを有効にする必要があるのは、データフラッシュ BGO（バックグラウンドモード操作）を使用する予定がある場合だけです。このモードでは、アプリケーションはデータフラッシュ操作を開始した後、ユーザー提供のコールバック関数を使用してその完了（またはエラー）の通知を非同期的に受けることができます。コー

ルバック関数には、コールバックイベント (FLASH_EVENT_ERASE_COMPLETE.) のソースを示すイベント情報が含まれる構造体が渡されます。

割り込みを有効にするには、e^2 studio のプロジェクト コンフィギュレーターの [ICU] タブで、FCU > FRDYI 割り込みの優先度を設定します。これにより、synergy_cfg/ssp_cfg/bsp/bsp_irq_cfg.h 内の BSP_IRQ_CFG_FCU_FRDYI が、選択したプライオリティレベルに設定されます。

BSP で FLASH FRDYI 割り込みが有効になっている場合、対応する ISR がフラッシュドライバーで定義されます。ISR は、ユーザーコールバック関数が open で登録されている場合は、それを呼び出します。

注 : フラッシュ HP は追加のフラッシュエラー割り込みをサポートしており、フラッシュ HP で BGO モードが有効になっている場合は、FRDYI および FIFERR 割り込みの両方にプライオリティを与える必要があります。

フラッシュ HAL モジュールのピン構成

フラッシュ回路は FCLK をそのクロックとして使用します。FCLK は 4MHz 以下である必要があります。<open API をコールした後でこのクロックレートを変更した場合は、updateFlashClockFreq をコールしてフラッシュ API にその変更を通知する必要があります。

4.2.18.6 アプリケーションでのフラッシュ HAL モジュールの使用

アプリケーションでフラッシュ HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、フラッシュ HAL を初期化します。
2) 割り込みを無効にします。
3) blankCheck API を使用して、コードフラッシュエリアのブランクチェックを行います。
4) erase API を使用して、1 つまたは複数のコードフラッシュブロックを消去します。
5) write API を使用して、コードフラッシュを書き込みます。
6) 割り込みを有効にします。
7) blankCheck API を使用して、データフラッシュエリアのブランクチェックを行います。
8) erase API を使用して、1 つまたは複数のデータフラッシュブロックを消去します。
9) write API を使用して、データフラッシュを書き込みます。
10) データフラッシュ BGO モードを有効にし、コールバック関数を割り当てます。
11) erase API を使用して、1 つまたは複数のデータフラッシュブロックを消去します。
12) コールバックに渡されたイベント情報をチェックして、消去が正常に完了したことを確認します。
13) すべてのフラッシュ操作が終了したら、close API を使用して閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

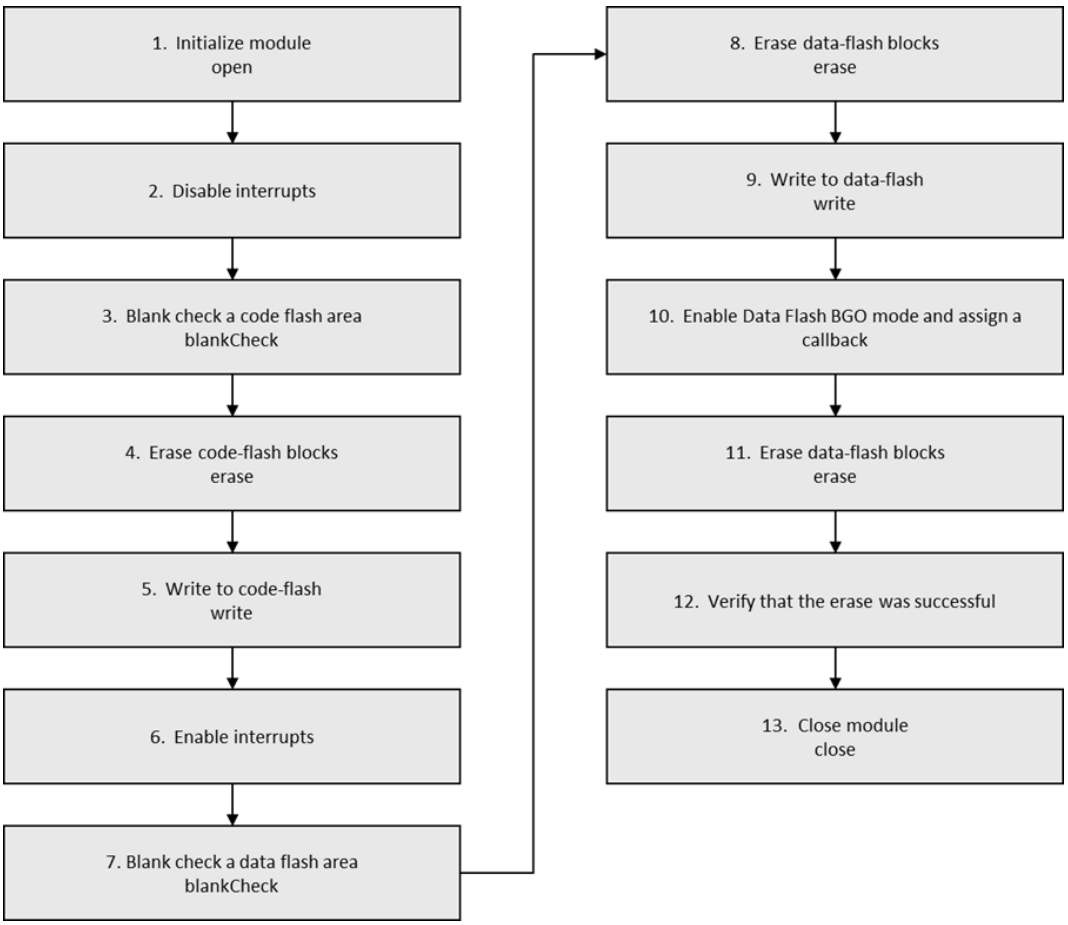


図 211:一般的なフラッシュ HAL モジュールアプリケーションのフロー図

4.2.19 FMI ドライバー

FMI HAL モジュールは、ファクトリ MCU 情報（FMI）フラッシュテーブルからレコードを読み取るアプリケーション用のハイレベル API を提供し、Synergy MCU 上のフラッシュインタフェースを使用します。

4.2.19.1 FMI HAL モジュールの特長

FMI HAL モジュールは、Synergy マイクロコントローラの FMIFRT（ファクトリー MCU 情報フラッシュルートテーブル）を読み取って、フラッシュ内のテーブルの先頭アドレスを取得します。このモジュールは、テーブルの製品情報レコードに呼び出し側のポインタを設定します。この情報を使用して、この MCU パッケージ固有の機能を判定できます。FMI HAL モジュールからは次の情報を利用できます。

- 製品情報（製品名、パッケージ、ピン数、温度範囲）
- 製品の機能（メジャーバージョン、マイナーバージョン、バリエーションデータ）
- 割り込みやイベントなどのイベント情報

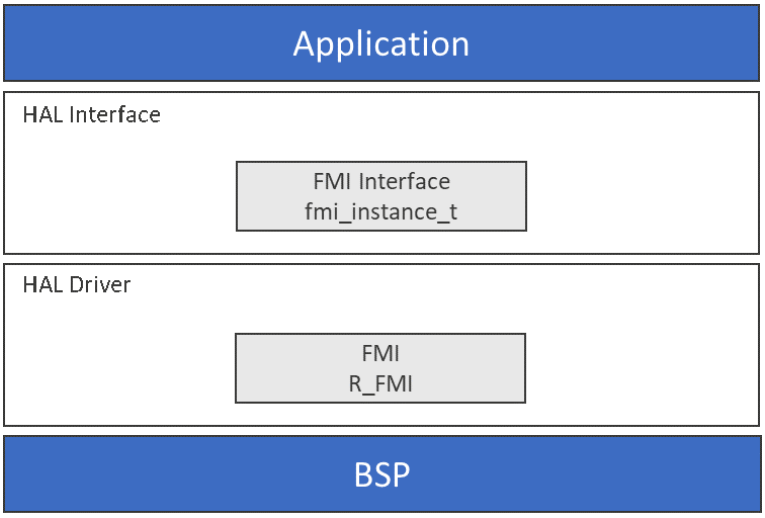


図 212:**FMI HAL モジュールのブロック図**

4.2.19.2 FMI HAL モジュールの API の概要

FMI HAL モジュールでは、FMIFRT にアクセスするための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

FMI HAL モジュールの API の要約

Function Name	Example API Call and Description
init	g_fmi.p_api->init(); Initialize the FMI base pointer.
productInfoGet	g_fmi.p_api->productInfoGet(&g_pp_product_info); Get product information record address into g_pp_product_info pointer.
uniqueIdGet	g_fmi.p_api->uniqueIdGet(&g_p_unique_id); Copy the unique ID into the g_p_unique_id pointer.

Function Name	Example API Call and Description
productFeatureGet	g_fmi.p_api->productFeatureGet(&g_ssp_feature, &g_feature_info); Get feature information and store it in g_feature_info pointer.
eventInfoGet	g_fmi.p_api->eventInfoGet(&g_ssp_feature, SSP_SIGNAL_GPT_COUNTER_OVERFLOW, &g_event_info); Get event information and store it in g_event_info pointer.
versionGet	g_fmi.p_api->versionGet(&g_p_version); Get the driver version based on compile time macros.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_INVALID_FMI_DATA	The FMI data table provided is not valid
SSP_ERR_IP_CHANNEL_NOT_PRESENT	Requested channel does not exist on this MCU
SSP_ERR_IP_UNIT_NOT_PRESENT	Requested unit does not exist on this MCU
SSP_ERR_INTERNAL	Requested feature is in a format not supported at this time
SSP_ERR_IRQ_BSP_DISABLED	Event information could not be found
SSP_ERR_ASSERTION	Caller's pointer is null
SSP_ERR_INVALID_FACTORY_FLASH	Factory flash is not valid

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.19.3 FMI HAL モジュールの動作の概要

FMI HAL モジュールは、productInfoGet API を使用して、製品情報レコードのアドレスを取得し、fmi_product_info_t 構造体に設定します。

FMI HAL モジュールは、uniqueIdGet API を使用して、一意の ID をコピーし、fmi_unique_id_t 構造体に設定します。

FMI HAL モジュールは、productFeatureGet API を使用して、機能の情報を取得し、fmi_feature_info_t 構造体に設定します。

FMI HAL モジュールは、eventInfoGet API を使用して、イベントの情報をフェッチし、fmi_event_info_t 構造体に設定します。

FMI HAL モジュールは、versionGet API を使用して、コードのバージョンと API のバージョンを取得し、ssp_version_t 構造体に設定します。

詳細については、FMI HAL モジュールのソースコードおよび SSP のユーザーマニュアルを参照してください。

FMI HAL モジュールの動作に関する重要な注意事項と制限事項

- fmi_product_info_t::unique_id は非推奨です。ファクトリー MCU 情報がアプリケーションコードによってリンクされている場合、一意の ID は含まれません。一意の識別子には uniqueIdGet を使用してください。
- FMI HAL インタフェースとその実装の制限事項については、SSP の最新のリリースノートを参照してください。
- FMI ドライバーは、FMIFRT 周辺機能レジスタを使用して、S7G2（WS2）Synergy マイクロコントローラファミリでテストされています。現在、ファクトリー MCU 情報テーブルのデータでプログラミングされている唯一の Synergy MCU です。
- S5D9 MCU の場合、uniqueIdGet API はエラーを返します。回避策はありません。

4.2.19.4 アプリケーションへの FMI HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに FMI HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

FMI ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（FMI ドライバーのデフォルト名は g_fmi0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

FMI HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_fmi FMI Driver on r_fmi	Threads > HAL/Common	New Stack> Driver> System> FMI Driver on r_fmi

次の図に示すように、r_fmi の FMI ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

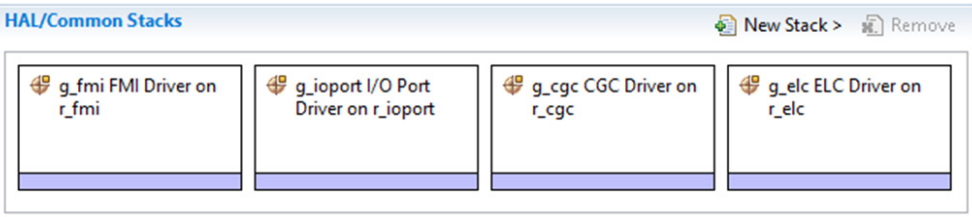


図 213:**FMI HAL モジュールのスタック**

4.2.19.5 FMI HAL モジュールの構成

ユーザーは必要な動作に合わせて FMI HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_fmi 上の FMI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Controls whether to include code for API parameter checking.
SSP MCU Information Symbol Name	g_fmi_data	This symbol maps to the base address where the factory flash table information will be found. It should not be modified.

ISDE Property	Value	Description
Part Number Mask	0xFE00	Each bit represents one character in the Synergy part number, where the MSB is the first character in the part number ('R'). Set bits to ensure the part number in the MCU factory flash matches the part number in the SSP MCU Information. The default mask checks everything except operating temperature, software ID, and quality ID.
Name	g_fmi	Module instance name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

FMI HAL モジュールのクロック構成

FMI HAL モジュールには、特定のクロック構成は必要ありません。

FMI HAL モジュールのピン構成

FMI HAL モジュールには、特定のピン構成は必要ありません。

4.2.19.6 アプリケーションでの FMI HAL モジュールの使用

アプリケーションで FMI HAL モジュールを使用する一般的な手順は次のとおりです。

1) init API を使用して FMI を初期化します。これは、リセットの後で自動的に行われます。
2) productInfoGet API を使用して、製品の情報を取得します。
3) uniqueIdGet API を使用して、一意の ID を取得します。
4) productFeatureGet API を使用して、機能の情報を取得します。
5) eventInfoGet API を使用して、イベントの情報を取得します。
6) versionGet API を使用して、ドライバーのバージョンの情報を取得します。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

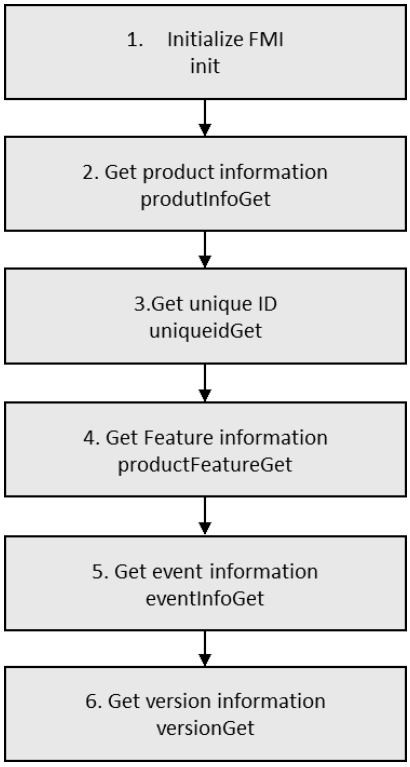


図 214:一般的な FMI HAL モジュールアプリケーションのフロー図

4.2.20 GPT ドライバー

汎用 PWM タイマ（GPT）HAL モジュールは、タイマアプリケーション用のハイレベル API を提供し、Synergy MCU 上の GPT ペリフェラルを使用します。ユーザー定義のコールバックを作成し、タイマイベントに応答できます。

4.2.20.1 GPT HAL モジュールの特長

GPT HAL モジュールは、ユーザーが指定した時間にタイマを構成します。この期間が経過すると、以下のいずれかのイベントが発生します。

- ユーザーコールバック関数が指定されている場合はそれを呼び出す CPU 割り込み
- ポートピンのトグル
- DMAC/DTC が転送インタフェースで構成されている場合はそれを使用したデータ転送
- イベントおよび周辺定義で構成されている別の周辺機能の開始

汎用 PWM タイマー (GPT)

- コアクロックとしての PCLKD
- チャネルごとに 2 個の I/O ピン

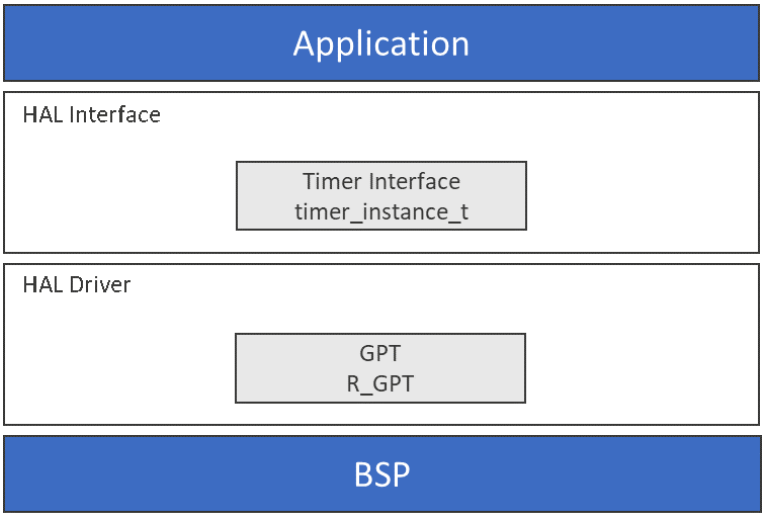


図 215:GPT HAL モジュールのブロック図

4.2.20.2 GPT HAL モジュールの API の概要

GPT HAL モジュールでは、モジュールのオープン、開始、停止、ステータスのリード、トリム、クローズのための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

GPT HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_timer0.p_api->open(g_timer0.p_ctrl, g_timer0.p_cfg) Initial configuration.
stop	g_timer0.p_api->stop(g_timer0.p_ctrl) Stop the counter.
start	g_timer0.p_api->start(g_timer0.p_ctrl) Start the counter.

Function Name	Example API Call and Description
reset	g_timer0.p_api->reset(g_timer0.p_ctrl) Reset the counter initial value.
counterGet	g_timer0.p_api->counterGet(&value) Get current counter value and store it in the provided pointer, value.
periodSet	g_timer0.p_api->periodSet(g_timer0.p_ctrl, period, unit) Set the time until the timer expires.
dutyCycleSet	g_timer0.p_api->dutyCycleSet(g_timer0.p_ctrl, period, unit, pin) Sets the time until the duty cycle expires.
infoGet	g_timer0.p_api->infoGet(&info) Get the time until the timer expires in clock counts and store it in provided pointer, info.
close	g_timer0.p_api->close(g_timer0.p_ctrl) Allows driver to be reconfigured and may reduce power consumption.
versionGet	g_timer0.p_api->versionGet(g_timer0.p_ctrl, &version) Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Operation is successful.
SSP_ERR_ASSERTION	Parameter is NULL or configuration setting is not allowed.
SSP_ERR_IN_USE	The channel specified has already been opened.
SSP_ERROR_NOT_OPEN	The channel is not open.
SSP_ERR_INVALID_ARGUMENT	Invalid argument provided.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.20.3 GPT HAL モジュールの動作の概要

GPT HAL モジュールは、ユーザーが指定した時間にタイマを構成します。時間が経過した時点で、CPU の割り込み、ポートピンの切り替え、DMAC または DTC を使用したデータ転送の開始、別の周辺機器の動作開始のトリガなどを行うことができます。

次の図は、指定した時間後にポートピンの切り替えまたは CPU 割り込みの生成を行うためのフローチャートを示します。

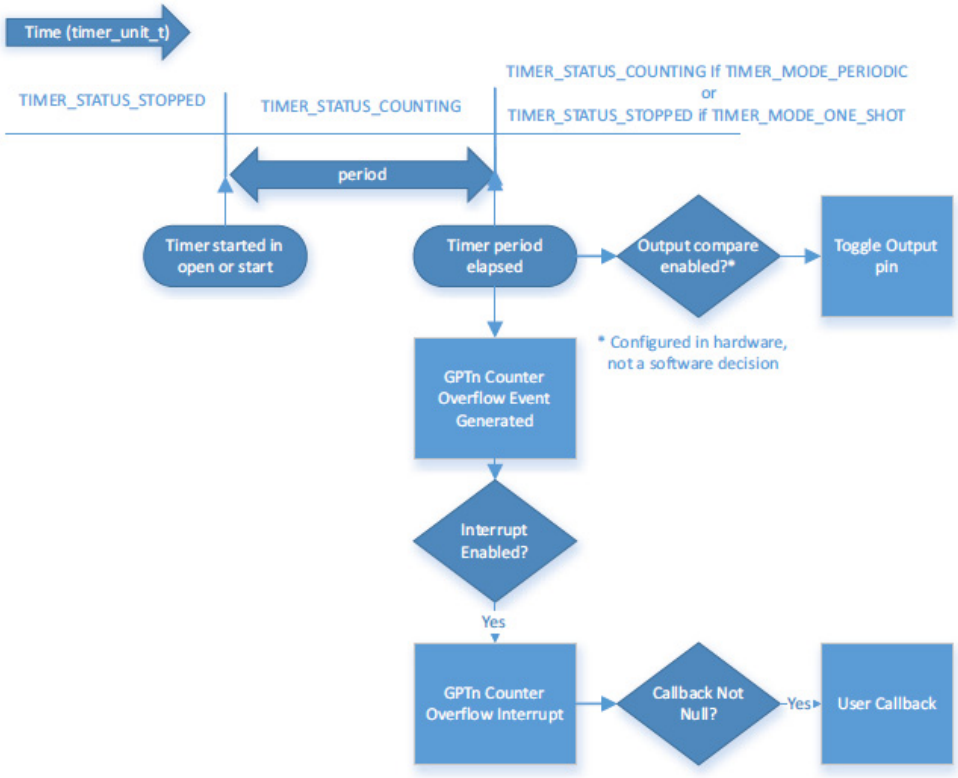


図 216:GPT HAL モジュールタイマ - 周期またはワンショットモード

SSP では、GPT と AGT の 2 種類のタイマモジュールがサポートされます。以下のセクションでは、開発者が特定のアプリケーションに対する各モジュールの機能を比較して対照できるように、両方のモジュールについての情報を提供します。AGT の追加情報については、『AGT ユーザーズガイド』を参照してください。

GPT モジュールは、ほとんどの汎用タイマアプリケーションに推奨されますが、基本的なタイマ機能にはどちらのモジュールでも使用できます。以下のユースケースでは、一方のタイマモジュールが他方より推奨される理由を説明します。

GPT タイマモジュールを選択した場合：GPT モジュールは、PCLKD のみでクロック供給可能な、高分解能の 32 ビットカウンタを使用します。Synergy デバイスでは AGT チャネルより GPT チャネルの方が多いので、GPT を使用することでリソースが競合する可能性が低くなります。

AGT タイマモジュールを選択した場合：AGT モジュールは、PCLKB、LOCO、または Fsub でクロック供給可能な 16 ビットカウンタを使用します。LOCO または Fsub でクロック供給された場合、AGT 割り込みを使用して MCU をスリープ モードから起動できます。2 つのチャネルがあり、チャネル 1 はチャネル 0 のアンダーフローによってクロック供給できるため、実質的に 32 ビットのカスケードされたタイマが作成されます。

GPT HAL モジュールの動作に関する重要な注意事項と制限事項

最大の時間間隔は、タイマの種類と入力クロック周波数に依存します。

- 120MHz で動作する PCLKD を使用する 32 ビット分解能の GPT では、最大期間は約 36650 秒、10 時間強です（GPT カウントクロックは PCLKD/1024 です）。
- 32MHz で動作する PCLKD を使用する 16 ビット分解能の GPT では、最大期間は約 2.09 秒です（GPT カウントクロックは PCLKD/1024 です）。

以下の状況では、GPT カウンタオーバーフロー割り込みを有効にする必要があります。

1) タイマ期間が経過したらソフトウェア割り込みを取得するため。
2) ワンショットモードを使用するため。

カウンタオーバーフロー割り込みが有効な場合、対応する ISR がベクタテーブル内でリンクされます。ISR は、ユーザーコールバック関数が open で登録されていれば、それをコールします。

注：irq が TRANSFER_IRQ_END. に設定された DTC ペリフェラルとともに使用した場合、割り込みがスキップされることがあります。

GPT 出力タイマ信号：タイマ出力が設定されている場合（GTIOCA/B 出力有効に True が設定されている場合）、出力ピンは GTIOCA/B 停止レベルで開始され、期間が経過するたびに切り替わります。最初の切り替えは、タイマ起動後に初めて期間が経過したときに起こります。

ワンショット モードでは、タイマーのカウントが開始されるときにも切り替わるように出力が設定されます。これによりパルスが生成されます。タイマーは、カウントが始まるときに停止レベルから切り替わり、カウントが終了するときに停止レベルに戻ります。

タイマ期間の計算：タイマ期間は、タイマが期限切れになるまでの時間として定義されます。出力比較を使用する場合、出力ピンが期間あたり 1 回トグルされるため、従来の期間（立ち上がりエッジから立ち上がりエッジ）は、ソフトウェアで指定された期間の 2 倍になります。

現在のクロック設定に基づく実行時の時間間隔の計算は、open および periodSet から使用できます。

- 指定された期間単位が raw カウントと異なる場合は、現在のタイマクロック周波数を使用して期間が計算されます。現在のタイマクロック周波数を確認するには、systemClockFreqGet API を使用します。この周波数は、次の表の適切な式で、clk_freq_hz. として使用されます。

タイマー期間の計算

Timer Units	Formula
TIMER_UNIT_PERIOD_NSEC	Counts = (period * clk_freq_hz) / 1000000000
TIMER_UNIT_PERIOD_USEC	Counts = (period * clk_freq_hz) / 1000000
TIMER_UNIT_PERIOD_MSEC	Counts = (period * clk_freq_hz) / 1000
TIMER_UNIT_PERIOD_SEC	Counts = (period * clk_freq_hz)
TIMER_UNIT_FREQUENCY_HZ	Counts = (clk_freq_hz) / period
TIMER_UNIT_FREQUENCY_KHZ	Counts = (clk_freq_hz) / 1000 * period

必要な時間間隔がカウンタのサイズ（32 ビットまたは 16 ビット）より長い場合、ドライバーは結果がカウンタのサイズに収まる最も小さい除数を選択します。カウンタ値が、最大の除数（1024）を持つカウンタサイズよりも大きい場合は、エラーコード (SSP_ERR_INVALID_ARGUMENT) が返されます。

GPT を使用した DMAC/DTC のトリガー

タイマの時間間隔が経過したときに、DMAC または DTC の周辺機器を使用してデータの転送をトリガするには、activation_source を ELC_EVENT_GPTn_COUNTER_OVERFLOW（n は GPT チャネル番号）に設定して DMAC/DTC 転送を設定します。詳細については、DMAC または DTC のガイドを参照してください。

*注 :DTC でワンショットモードのタイマを使用した場合、IRQ が TRANSFER_IRQ_END. に設定されていると、割り込みによりタイマが停止する前に、転送全体が完了します。タイマ期間が経過した後で 1 回だけ転送を生成するには、IRQ を TRANSFER_IRQ_EACH, に設定するか、DMAC を使用して転送します。*GPT を使用した ELC イベントのトリガ

GPT タイマは、他の周辺機器の起動をトリガできます。ELC のガイドには、すべての使用可能な周辺機器のリストがあります。

フリーランカウンタモード

GPT をフリーランカウンタとして使用するには、Synergy 構成ツールの [Properties] ウィンドウで、[Period] を 32 ビットタイマの場合は 0xFFFFFFFF に、16 ビットタイマの場合は 0xFFFF に設定し、[Period Unit] を [Raw Counts] に設定します。タイマの開始と停止には、stop API および start API を使用します。カウンタの値の確認には、counterGet API を使用します。タイマのリセットには、reset API を使用します。カウンタがオーバーフローした場合は、コールバックでカウンタオーバーフローを処理します。

GPT PWM モード

PWM モードで GPT を使用するには、[Period] および [Duty cycle] を設定し、デューティサイクルの範囲を選択します。

ドライバーには、デューティサイクルの範囲を選択するために 2 つのオプションが用意されています。

1) [Shortest duty cycle] がオフ：この場合、取得される最も低いデューティサイクルは 2 raw カウントに制限されます。ただし、構成は 1 raw カウントに制限されます（ON 時にハードウェアが 1 クロックサイクルを追加するので、デューティサイクルが 1 raw カウントに構成されていると、ユーザーにとっては ON 時に 2 raw カウントとなるため）。
2) [Shortest duty cycle] がオン：この場合、最も低いデューティサイクルである 1 raw カウントが達成され、構成する最も低いデューティサイクルは 1 raw カウントに、最長は [Period] 値 –2 に制限されます。また、OFF 時にハードウェアによって 1 クロックサイクルが追加されます。

- GPT の電源オフの場合、GPT モジュールは close API で GPT のモジュールストップビット（MSTP）を設定しません。これは意図的なもので、GPT モジュールストップビットは複数の GPT チャネルを制御し、GPT モジュールには他の GPT モジュールがアプリケーションで使用されているかどうかを判断することはできないためです。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.20.4 アプリケーションへの GPT HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに GPT HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

タイマドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（タイマドライバーのデフォルト名は g_timer0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

GPT HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_timer0 Timer Driver on r_gpt	Threads -> HAL/Common	New Stack> Driver> Timers> Timer Driver on r_gpt

次の図に示すように、r_gpt のタイマドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

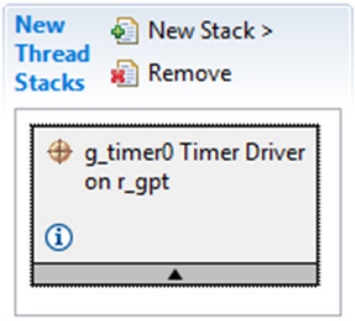


図 217:GPT HAL モジュールのスタック

4.2.20.5 GPT HAL モジュールの構成

ユーザーは必要な動作に合わせて GPT HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_gpt 上の GPT HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_timer0	Module name.
Channel	0	Channel selection.
Mode	Periodic, One Shot, PWM Default: Periodic	Warning: One Shot functionality is not available in the GPT hardware, so it is implemented in software by stopping the timer in the ISR called when the period expires. For this reason, ISR's must be enabled for one-shot mode even if the callback is unused.
Period Value	10	See Timer Period Calculation
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz Default: Milliseconds	See Timer Period Calculation
Duty Cycle Value	50	Duty cycle value selection
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	Duty cycle unit selection
Auto Start	True, False Default: True	Set to true to start the timer after configuring or false to leave the timer stopped until is called.

ISDE Property	Value	Description
GTIOCA Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
GTIOCB Output Enabled	True, False Default: False	Set to true to output the timer signal on a port pin configured for GPT. Set to false for no output of the timer signal.
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	Controls output pin level when the timer is stopped.
Callback	NULL	A user callback function can be registered in . If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the timer period elapses. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Overflow Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Overflow interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

GPT HAL モジュールのクロック構成

GPT タイマは、PCLKD 周波数に基づいてクロック供給されます。PCLKD の周波数は、ISDE の [Configuring Clocks] タブで、または実行時に CGC インタフェースで、クロックコンフィギュレータを使用して設定できます。

GPT HAL モジュールのピン構成

GPT 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では対応するピンの選択例を示します。

r_gpt 上の GPT HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
GPT	Pins	Select Peripherals > Timer: GPT>GPT0

注 : この選択シーケンスでは、GPT0 がドライバーに必要なハードウェアターゲットであることを想定しています。

GPT HAL ドライバーのピン構成設定

Property	Value	Description
Pin Group Selection	Mixed, _A Only, _B Only Default: Mixed	Select pin group mapping.
Operation Mode	Disabled, GTIOCA or GTIOCB, GTIOCA and GTIOCB Default: Disabled	Select timer operation mode.
GTIOCA	None, P300, P512 Default: P512	GTIOCA Pin.
GTIOCB	None, P108, P511 Default: P511	GTIOCB Pin.

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

4.2.20.6 アプリケーションでの GPT HAL モジュールの使用

アプリケーションで GPT HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、GPT HAL モジュールを初期化します。
2) [Auto Start] プロパティが [False] の場合は、start API をコールすることによって GPT HAL モジュールを開始します。
3) 必要に応じて、タイマコールバックに応答します（アプリケーションコード）。

注 :GPT 期間とデューティサイクルのパラメータは、アプリケーションのニーズに基づいて periodSet および dutyCycleSet を使用して再設定できます。

PWM モードでは、デューティサイクルの範囲で
GPT_SHORTEST_LEVEL_OFFが選択されていればON時に、デューティサイクルの範囲でGPT_SHORTEST_LEVEL_ON.
が選択されていれば OFF 時に、ハードウェアによって PCLK が 1 つ追加されます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

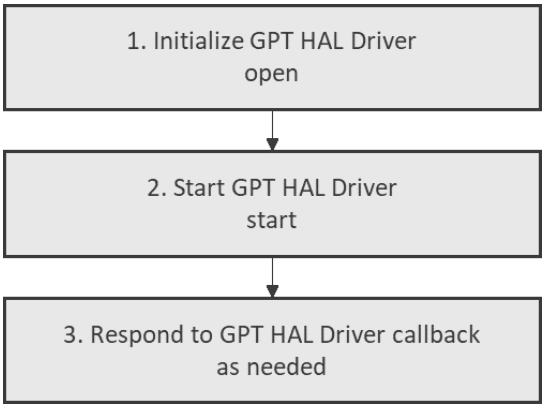


図 218:一般的な GPT HAL モジュールアプリケーションのフロー図

4.2.21 I2C SCI ドライバー

I2C SCI マスタ HAL モジュールは、I2C 業界標準シリアルデバイス通信アプリケーション用のハイレベル API を提供し、Synergy MCU デバイス上の SCI ペリフェラルを使用します。転送完了用と受信完了用のコールバックが用意されています。

4.2.21.1 I2C SCI HAL モジュールの特長

- I2C SCI 動作のサポート

- スレーブ I2C SCI デバイスでの以下の動作のサポート
  - 読み取り
  - 書き込み
  - リセット
- コールバックのサポート
  - 転送中断
  - 送信完了（送信済みバイト数が提供されます）
  - 受信完了（受信済みバイト数が提供されます）

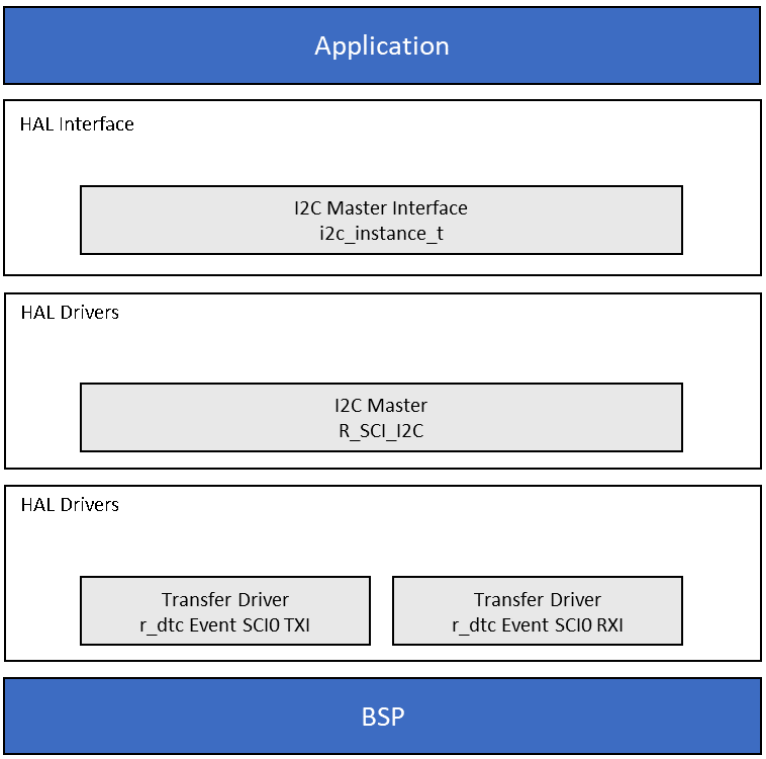


図 219:I2C SCI HAL モジュールのブロック図

4.2.21.2 I2C SCI HAL モジュールの API の概要

I2C SCI HAL モジュールでは、マスタ I2C デバイスを使用したリードとライトのための API が定義されています。次の表には、使用可能なすべての API 関数のリスト、API 関数コールの例、各 API 関数の簡単な説明が記載されています。ステータス戻り値の表は API 要約表の後にあります。

I2C SCI HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_i2c.p_api->open(g_i2c.p_ctrl, g_i2c.p_cfg); Open the instance and initialize the hardware.
close	g_i2c.p_api->close(g_i2c.p_ctrl); Closes the driver and releases the I2C device.
read	g_i2c.p_api->read(g_i2c.p_ctrl, &destination, bytes, restart); Performs a read operation on an I2C device.
write	g_i2c.p_api->write(g_i2c.p_ctrl, &destination, bytes, restart); Performs a write operation on an I2C device.
reset	g_i2c.p_api->reset(g_i2c.p_ctrl); Reset the peripheral.
slaveAddressSet	g_i2c.p_api->slaveAddressSet(g_i2c.p_ctrl, slave, addr_mode); Sets address of the slave device without reconfiguring the bus.
versionGet	g_i2c.p_api->versionGet(&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_IN_USE	Attempted to open an already open device instance.
SSP_ERR_ABORTED	Device was closed while a transfer was in progress.
SSP_ERR_INVALID_RATE	The requested rate cannot be set
SSP_ERR_ASSERTION	The parameter p_ctrl is NULL.
SSP_ERR_NOT_OPEN	Device was not even opened.
SSP_ERR_IRQ_BSP_DISABLED	Event information could not be found.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.21.3 I2C SCI HAL モジュールの動作の概要

I2C SCI マスタ HAL モジュールは、I2C スレーブデバイスでのトランザクションをサポートします。送信が完了または中断したとき、または受信が完了したときに CPU に割り込むためのコールバックが提供されます。I2C SCI HAL モジュールは、バッファ内の受信バイト数または送信バイト数、ユーザーが指定したコンテキストへのポインタ、およびイベント i2c_event_t を示す i2c_callback_args_t 引数を使用して、コールバックを呼び出します。

I2C SCI HAL モジュールの動作に関する重要な注意事項と制限事項

割り込み

- 選択したチャネルの I2C 割り込み（SCI エラー（EEI)、受信バッファフル（RXI)、送信バッファエンプティ（TXI)、および送信終了（TEI））を、ユーザーがコールバックを使用するかどうかにかかわらず、ボードサポートパッケージ（BSP）で有効にする必要があります。
- 割り込みを異なる優先度に設定すると、適切に動作しなくなる可能性があります。

IIC レート計算

- I2C SCI HAL モジュールは、構成されている転送レートに基づいて内部ボーレートの設定を計算し、それを open に渡します。現在の PCLKB 設定で達成可能な最も近いボー レート (要求されたレートより小さいか等しいもの) が計算されて使用されます。
- 有効なクロック レートを計算できなかった場合は、エラーが返されます。

IIC を使用した DMAC/DTC のトリガ

- DTC 転送のサポートは、コンフィギュレータにおいてデフォルトで追加されますが、CPU 転送を行う場合は削除できます。DTC はモジュール内で構成されます。これに関してユーザーの構成は必要ありません。

- DMA 転送はサポートされていません。

IIC を使用した ELC イベントのトリガ

- I2C SCI HAL モジュールは、他の周辺機能の開始をトリガできます。詳細については、『ELC User Guide』を参照してください。

同じバス上の複数のデバイス

同じバス上の複数のスレーブデバイスと通信する場合、スレーブデバイスの構成設定が同じであれば、slaveAddressSet API 関数を使用することにより、再設定することなく（閉じてから開く必要なく）スレーブデバイスを切り替えることができます。制御インスタンスとバスの構成は同じままですが、スレーブアドレスとアドレシングモードは変わります。この場合、I2C SCI HAL モジュールのインスタンスは 1 つで十分です。

同じバス上の複数のスレーブデバイスと通信する場合、各スレーブデバイスで異なる構成設定が必要なときは、r_sci_i2c の複数のインスタンスを使用し、必要に応じてそれぞれを各スレーブデバイスに合わせて構成します。r_sci_i2c の各インスタンスは、ターゲットスレーブデバイスとの通信に使用されるたびに開いたり閉じたりされます。

スレーブデバイスが混在している状況では、この 2 つの方法を組み合わせて使用することができます。たとえば、2 台のスレーブデバイスが同じ構成設定を共有している場合、これらは同じ r_sci_i2c インスタンスを共有し、slaveAddressSet API 関数を使用して切り替えることができます。この同じバス上で 3 台の別のスレーブデバイスが同じ構成設定を共有していて、最初の 2 台とは異なる設定である場合、これらには別の r_sci_i2c インスタンスが必要です。この 3 台のスレーブデバイスも、slaveAddressSet API 関数を使用して切り替えることができます。異なる r_sci_i2c, インスタンスを使用するスレーブデバイスに切り替えるには、open および close による方法を使用する必要があります。原則として、最後にアクセスしたスレーブデバイスの構成設定が「記憶」され、必要に応じて再利用されます。最後にアクセスされたスレーブと異なる構成をスレーブが必要とする場合は、r_sci_i2c モジュールを閉じてから、必要な構成設定で開き直す必要があります。

同じチャネルに接続された複数のデバイスを使用するアプリケーションでは、プロジェクトのプリプロセッサの設定で以下のマクロを定義する必要があります（定義しないと、プロジェクトが正しくビルドされない可能性があります）。

SSP_SUPPRESS_ISR_<device_name>

<device_name> は、同じチャネルに接続される追加デバイスの名前です。

- IRQ モードの I2C SCI HAL モジュールは、特定のスレーブデバイスでは動作しない場合があります。そのようなデバイスでモジュールを動作させるには、DTC 転送モードを有効にする必要があります。
- IRQ モードで動作しているときに高ビットレートのデータ転送をサポートするためには、DTC 転送モードを有効にすることが推奨されます。これにより、割り込みに直接反応するときの CPU のオーバーヘッドが削減されるため、高ビットレートのデータ転送を正常に実装する際の潜在的な障壁が取り除かれます。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.21.4 アプリケーションへの I2C SCI HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに I2C SCI HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

I2C マスタドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(I2C マスタドライバーのデフォルト名は g_i2c0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

I2C SCI HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_i2c0 I2C Master Driver on r_sci_i2c	Threads	New Stack> Driver> Communications> I2C Master Driver on r_sci_i2c

次の図に示すように、r_sci_i2c の I2C マスタドライバーモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

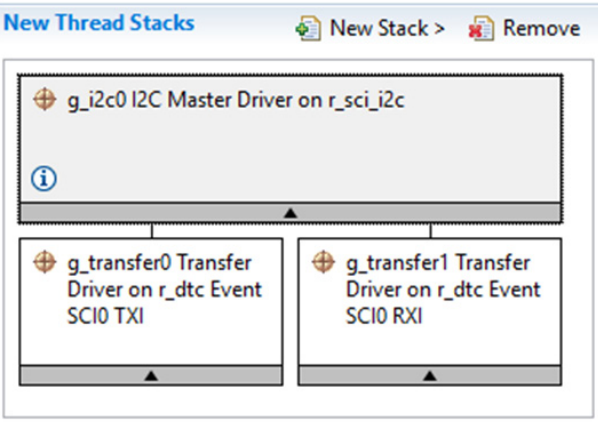


図 220:I2C SCI HAL モジュールのスタック

4.2.21.5 I2C SCI HAL モジュールの構成

ユーザーは必要な動作に合わせて I2C SCI HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスでき

るすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_sci_i2c 上の I2C SCI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Name	g_i2c0	Module name.
Channel	0 to 9	Specify the SCI channel to be used with this configuration. SCI has channels as follows: Series S7 : 0 1 2 3 4 5 6 7 8 9; Series S3 : 0 1 2 3 4 - - - - 9; Series S1 : 0 1 - - - - - - - 9 .
Rate	Standard, Fast-mode Default: Standard	Select the I2C data rate.
Slave Address	0x00	Specify the slave address.
Address Mode	7-Bit, 10-Bit Default: 7-Bit	Only 7-bit addresses are currently supported.
SDA Output Delay (nano seconds)	300	SDA output delay in nanoseconds.
Bit Rate Modulation Enable	Enable, Disable Default: Enable	Enables bitrate modulation function.

ISDE Property	Value	Description
Callback	NULL	A user callback function can be registered in . If this callback function is provided, it will be called from the interrupt service routine (ISR) for each of the conditions defined in i2c_event_t. Warning: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Select the receive interrupt priority.
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Select the transmit interrupt priority.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Select the transmit end interrupt priority.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

I2C SCI HAL モジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_dtc Event SCI0 TXI 時の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Software Start	Enabled, Disabled Default: Disabled	Include code for software start in the build.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Section to place the DTC vector table.
Name	g_transfer0	Module name.
Mode	Block	Specify the hardware channel.
Transfer Size	1 Byte	Select the transfer mode.
Destination Address Mode	Fixed	Select the transfer size.
Source Address Mode	Incremented	Select the address mode for the destination.
Repeat Area (Unused in Normal Mode	Source	Select the address mode for the source.
Interrupt Frequency	After all transfers have completed	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.
Number of Transfers	0	Specify the number of transfers.

ISDE Property	Value	Description
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source (Must enable IRQ)	Event SCI0 TXI	Select the DTC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback (Only valid with Software start)	NULL	A user callback that is called at the end of the transfer.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Software Start	Enabled, Disabled Default: Disabled	Include code for software start in the build.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Section to place the DTC vector table.
Name	g_transfer1	Module name.
Mode	Normal	Specify the hardware channel.

ISDE Property	Value	Description
Transfer Size	1 Byte	Select the transfer mode.
Destination Address Mode	Incremented	Select the transfer size.
Source Address Mode	Fixed	Select the address mode for the destination.
Repeat Area (Unused in Normal Mode	Destination	Select the address mode for the source.
Interrupt Frequency	After all transfers have completed	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.
Number of Transfers	0	Specify the number of transfers.
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source (Must enable IRQ)	Event SCI0 RXI	Select the DTC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback (Only valid with Software start)	NULL	A user callback that is called at the end of the transfer.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

I2C SCI HAL モジュールのクロック構成

SCI ペリフェラルモジュールは PCLKB をそのクロックソースとして使用します。実際の I2C 転送レートは、ドライバーにより（選択された転送レートに応じて）計算されて内部的に設定されます。選択された内部レートを実現できないように PCLKB が構成されている場合は、ドライバーを初期化するときにエラーが返されます。

I2C SCI HAL モジュールのピン構成

SCI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

注：動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決まります。

r_sci_i2c 上の I2C SCI HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SCI	Pins	Select Peripherals > Connectivity: SCI > SCI0

注 : この選択シーケンスでは、SCI0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_sci_i2c 上の I2C SCI HAL モジュールのピン構成設定

Pin Configuration Property	Value	Description
Pin Group Selection	_A only, _B only, Mixed Default: _A only	Pin group selection
Operation Mode	Enabled, Disabled Default: Disabled	Enable or disable peripheral module
SDA	None, P401, P407 Default: None	SDA Pin

Pin Configuration Property	Value	Description
SCL	None, P400, P204 Default: None	SCL Pin

注 : 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.2.21.6 アプリケーションでの I2C SCI HAL モジュールの使用

一般的なアプリケーションで I2C SCI HAL モジュールを使用する手順は次のとおりです。

1) open API を使用して、I2C SCI HAL モジュールを初期化して開きます。
2) write API を使用して、スレーブにデータを転送します。
3) read API を使用して、スレーブからデータを受信します。
4) 受信したデータをアプリケーションの必要に応じて操作します。
5) reset API を使用して、モジュールをリセットします（オプション）。
6) アプリケーションコードにより、スレーブデバイスでトランザクションを実行します。
7) slaveAddressSet API を使用して、スレーブアドレスを変更します（オプション）。
8) アプリケーションコードにより、スレーブデバイスでトランザクションを実行します（オプション）。
9) close API を使用して、チャネルを閉じます（オプション）。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

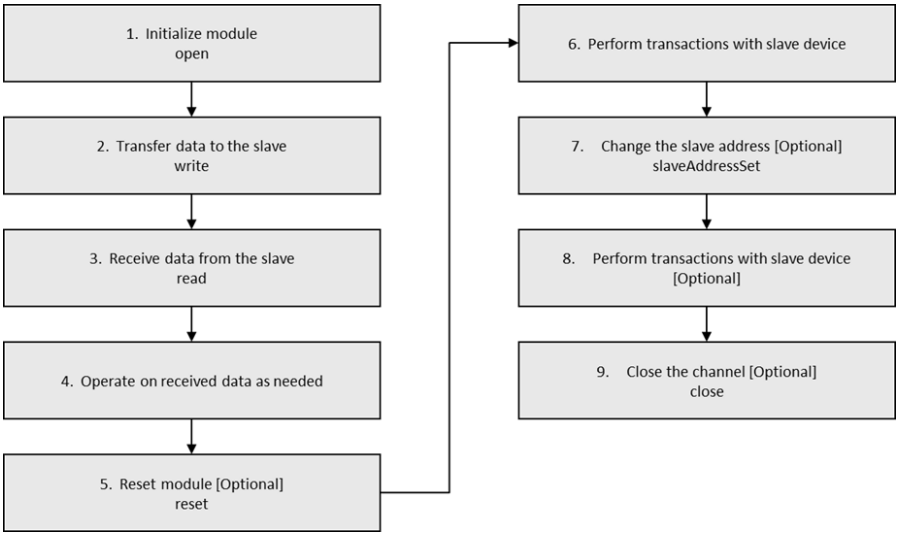


図 221:一般的な I2C SCI HAL モジュールアプリケーションのフロー図

4.2.22 I2C マスタドライバー

RIIC 上の I2C マスタ HAL モジュールは、業界標準の I2C シリアル通信アプリケーション用のハイレベル API を提供し、Synergy MCU 上の IIC ペリフェラルを使用します。転送完了と受信完了のイベント通知用のコールバックが用意されています。

4.2.22.1 I2C マスタ HAL モジュールの特長

- I2C RIIC 動作のサポート
  - 標準（最大 100kHz）
  - I2C 高速モード（最大 400kHz）
  - I2C 高速モードプラス（S7G2 および S5D9 MCU ファミリのチャネル 0（SCL0-A、SDA0-A）で最大 1MHz）
- RIIC モジュールの初期化
- スレーブ デバイスへの書き込み
- I2C 周辺機器のリセット
- MCU I2C ペリフェラルのリセット
- スレーブデバイスのアドレスの設定
- コールバックのサポート
  - 転送中断
  - 送信完了（送信済みバイト数が提供されます）
  - 受信完了（受信済みバイト数が提供されます）

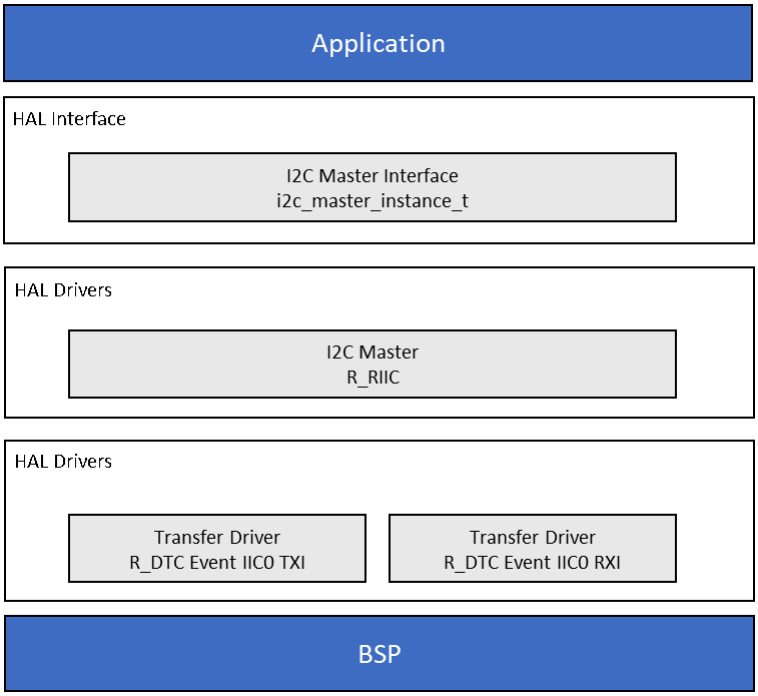


図 222:I2C マスタ HAL モジュールのブロック図

4.2.22.2 I2C マスタ HAL モジュールの API の概要

RIIC（I2C RIIC）上の I2C マスタ HAL モジュールでは、マスタ I2C デバイスを使用したリードとライトなどのための API 関数が定義されています。次の表には、使用可能なすべての API 関数のリスト、API 関数コールの例、各 API 関数の簡単な説明が記載されています。ステータス戻り値の表は API 要約表の後にあります。

I2C マスタ HAL モジュールの API の要約の表

Function Name	Example API Call and Description
open	g_i2c.p_api->open(g_i2c.p_ctrl, g_i2c.p_cfg); Open the instance and initialize the hardware.
close	g_i2c.p_api->close(g_i2c.p_ctrl); Closes the driver and releases the I2C device.

Function Name	Example API Call and Description
read	g_i2c.p_api->read(g_i2c.p_ctrl, &destination, bytes, restart); Performs a read operation on an I2C device.
write	g_i2c.p_api->write(g_i2c.p_ctrl, &destination, bytes, restart); Performs a write operation on an I2C device.
reset	g_i2c.p_api->reset(g_i2c.p_ctrl); Reset the peripheral.
versionGet	g_i2c.p_api->versionGet(&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_POINTER	Pointer is NULL
SSP_ERR_IN_USE	Attempted to open an already open device instance.
SSP_ERR_ABORTED	Device was closed while a transfer was in progress.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value
SSP_ERR_INVALID_RATE	The requested rate cannot be set

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.22.3 I2C マスタ HAL モジュールの動作の概要

RIIC 上の I2C マスタ HAL モジュールは、I2C スレーブデバイスでのトランザクションをサポートします。送信または受信が完了したときに CPU に割り込むためのコールバックが提供されます。RIIC HAL モジュールは、バッファ内の受信バイト数または送信バイト数、ユーザーが指定したコンテキストへのポインタ、およびイベント i2c_event_t を示す引数 i2c_callback_args_t を使用して、コールバックを呼び出します。

I2C マスタ HAL モジュールの動作に関する重要な注意事項と制限事項

割り込み

- 選択した使用チャネルの RIIC エラー（EEI）、受信バッファフル（RXI）、送信バッファエンプティ（TXI）、および送信終了（TEI）割り込みを、ユーザーがコールバックを使用するかどうかにかかわらず、選択したデバイスのプロパティで有効にする必要があります。
- 上記の割り込みすべてに同じプライオリティレベルを設定します。割り込みを異なる優先度に設定すると、適切に動作しなくなる可能性があります。

IIC レート計算

- I2C マスタモジュールは、構成されている転送レートに基づいて内部ボーレートの設定を計算し、それを open に渡します。現在の PCLKB 設定で達成可能な最も近いボーレート（要求されたレートより小さいか等しいもの）が計算されて使用されます。
- 有効なクロック レートを計算できなかった場合は、エラーが返されます。

IIC を使用した DMAC/DTC のトリガ

- DTC 転送のサポートは、コンフィギュレータにおいてデフォルトで追加されます。CPU 転送の場合は、これを削除できます。DTC はモジュール内で構成されます。これに関してユーザーの構成は必要ありません。
- DMA 転送はサポートされていません。

IIC を使用した ELC イベントのトリガ

- I2C マスタモジュールは、他の周辺機能の開始をトリガできます。詳細については、『ELC User Guide』のイベントと周辺機器の定義を参照してください。

バス上の複数のデバイス

- 複数のデバイスが同じバスに接続されている場合、一度に開くことのできるデバイスは 1 つのみです。
- 複数のスレーブデバイスが同じバス上にあり、構成が異なる場合、アプリケーションプログラムは複数の I2C マスタモジュール（構成ごとに 1 つずつ）を使用する必要があります。
- アプリケーションで、バスを開いたり閉じたりすることなしにデバイスを切り替える必要がある場合は、slaveAddressSet API を使用します。g_i2c.p_ctrl は、最後に開いたデバイスで使用されていたものと同じ制御インスタンスです。このモジュールでは、アプリケーションプログラムが連続して read または write API 関数をコールするとき、新しいデバイスとの通信に同じバス構成が使用されます。

リスタート条件の使用法

- write/read API の restart パラメータに値「true」を渡すと、指定されたバイト数（長さ）の後でリスタート条件が生成されます。マスタによってバスはタイムアウトすることなく継続的にビジー（ロー）に保持されるので、現在のマスタが次の write/read API をトリガできます。

マルチマスタのサポート

同じバスに複数のマスタが接続されている場合、I2C マスタは通信を開始する前にバスのビジー状態を検出できます。

- サポートされている IIC チャネルであればどれでも、マスタとスレーブのどちらのモードの動作にも構成できますが、両方に構成することはできません。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.22.4 アプリケーションへの I2C マスタ HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに I2C マスタ HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

I2C マスタドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(I2C マスタドライバーのデフォルト名は g_i2c0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

I2C マスタ HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_i2c0 I2C Master Driver on r_riic	Threads	New Stack> Driver> Communications> I2C Master Driver on r_riic

次の図に示すように、r_riic の I2C マスタ HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

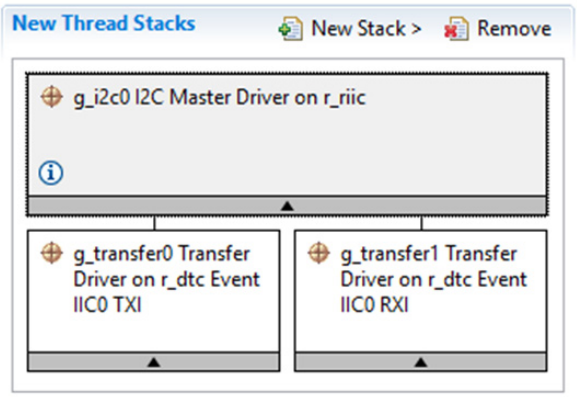


図 223:I2C マスタ HAL モジュールのスタック

4.2.22.5 I2C マスタ HAL モジュールの構成

ユーザーは必要な動作に合わせて I2C マスタ HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_riic 上の I2C マスタ HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.
Name	g_i2c0	Module name.
Channel	0	Specify the IIC channel to be used with this configuration.

ISDE Property	Value	Description
Rate	Standard, Fast-mode, Fast-mode Plus Default: Standard	Standard, Fast, and Fast-plus. (See IIC Rate Calculation.)
Slave Address	0x00	Set the address of the slave device the I2C master will be communicating with.
Address Mode	7-Bit, 10-Bit Default: 7-Bit	Only 7-bit addresses are currently supported.
Callback	NULL	A user callback function can be registered in . If this callback function is provided, it will be called from the interrupt service routine (ISR) for each of the conditions defined in i2c_event_t. Warning: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Receive interrupt priority selection.
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Transmit interrupt priority selection.

ISDE Property	Value	Description
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Transmit end interrupt priority selection.
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Error interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

I2C マスタ HAL モジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_dtc Event IIC0 TXI 時の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection

ISDE Property	Value	Description
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event IIC0 TXI	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event IIC0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event IIC0 RXI	Activation source selection
Auto Enable	FALSE	Auto enable selection

ISDE Property	Value	Description
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

I2C マスタ HAL モジュールのクロック構成

IIC ペリフェラルモジュールは PCLKB をそのクロックソースとして使用します。実際の I2C 転送レートは、選択された転送レートに応じて、ドライバーにより計算されて内部的に設定されます。選択された内部レートを実現できないような設定が PCLKB で行われた場合は、ドライバーを初期化するときにエラーが返されます。

I2C マスタ HAL モジュールのピン構成

IIC 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

注：動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決まります。

r_riic 上の I2C マスタ HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
IIC	Pins	Select Peripherals > Connectivity: IIC > IIC0

注 : この選択シーケンスでは、IIC0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_sci_uart 上の I2C マスタ HAL モジュールのピン構成設定

Pin Configuration Property	Value	Description
Pin Group Selection	_A only, _B only, Mixed Default: _A only	Pin group selection

Pin Configuration Property	Value	Description
Operation Mode	Enabled, Disabled Default: Disabled	Enable or disable peripheral module
SDA	None, P401, P407 Default: None	SDA Pin
SCL	None, P400, P204 Default: None	SCL Pin
SCK	None, P412, P102 Default: P412	SCK Pin
CTS_RTS_SS	None, P413, P103 Default: None	CTS Pin
SDA	Disabled	SDA Pin (when Simple I2C is used)
SCL	Disabled	SCL Pin (when Simple I2C is used)

注 : 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.2.22.6 アプリケーションでの I2C マスタ HAL モジュールの使用

一般的なアプリケーションで I2C マスタ HAL モジュールを使用する手順は次のとおりです。

1) open API を使用して、I2C RIIC HAL モジュールを初期化して開きます。
2) write API を使用して、スレーブにデータを転送します。
3) read API を使用して、スレーブからデータを受信します。
4) reset API を使用して、モジュールをリセットします（オプション）。
5) slaveAddressSet API を使用して、スレーブアドレスを変更します（オプション）。
6) write API を使用して、スレーブにデータを転送します（オプション）。
7) read API を使用して、スレーブからデータを受信します（オプション）。

8) close API を使用して、モジュールを閉じます（オプション）。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

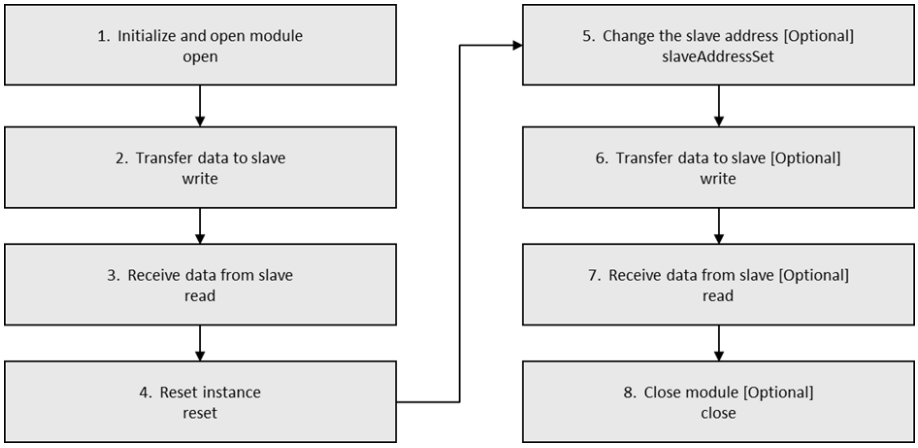


図 224:一般的な I2C マスタ HAL モジュールアプリケーションのフロー図

4.2.23 I2C スレーブドライバー

RIIC スレーブ上の I2C スレーブ HAL モジュールは、I2C スレーブアプリケーション用のハイレベル API を提供し、Synergy MCU 上の RIIC ペリフェラルを使用します。転送完了と受信完了の通知用のコールバックが用意されています。

4.2.23.1 I2C スレーブ HAL モジュールの特長

- I2C スレーブ動作のサポート
- I2C マスタデバイスとのトランザクションのサポート
  - 読み取り
  - 書き込み
- コールバックのサポート
  - 送信完了（送信済みバイト数が提供されます）
- 受信完了（受信済みバイト数が提供されます）

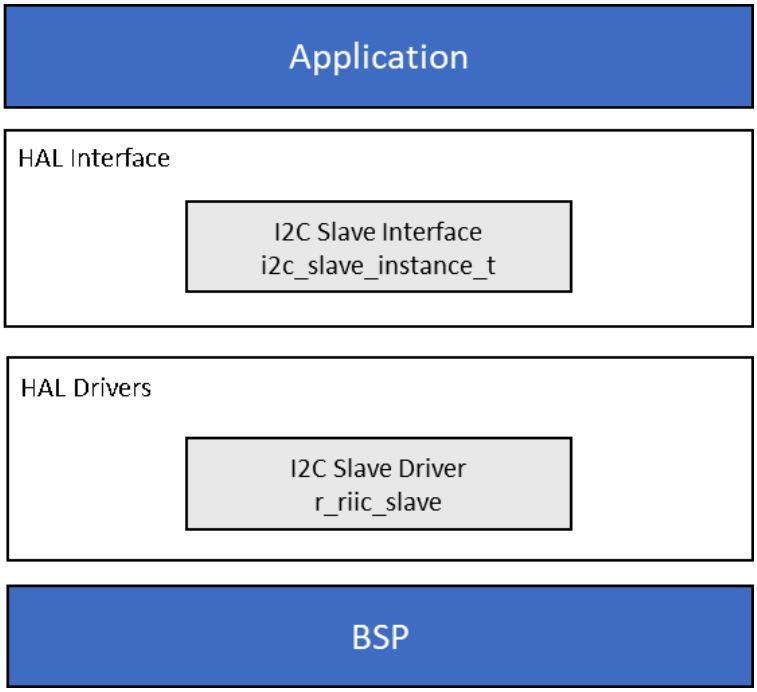


図 225:I2C スレーブ HAL モジュールのブロック図

4.2.23.2 I2C スレーブ HAL モジュールの API の概要

I2C RIIC スレーブ HAL モジュールでは、マスタ I2C デバイスのリードとライトのための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

I2C スレーブ HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_i2c.p_api->open(g_i2c.p_ctrl, g_i2c.p_cfg); Open the instance and initialize the hardware.
close	g_i2c.p_api->close(g_i2c.p_ctrl); Closes the driver and releases the I2C device.

Function Name	Example API Call and Description
masterWriteSlaveRead	g_i2c.p_api->masterWriteSlaveRead(g_i2c.p_ctrl, &destination, bytes); Performs a read operation on an I2C device.
masterReadSlaveWrite	g_i2c.p_api->masterReadSlaveWrite(g_i2c.p_ctrl, &source, bytes); Performs a write operation on an I2C device.
versionGet	g_i2c.p_api->versionGet(&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_POINTER	Pointer is NULL
SSP_ERR_IN_USE	Attempted to open an already open device instance.
SSP_ERR_ABORTED	Device was closed while a transfer was in progress.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.23.3 I2C スレーブ HAL モジュールの動作の概要

I2C RIIC スレーブ HAL モジュールは、I2C マスタデバイスへの転送をサポートします。送信または受信プロセスが完了したときに CPU に割り込むためのコールバックが提供されます。

I2C スレーブ HAL モジュールの動作に関する重要な注意事項と制限事項

- 選択した使用チャネルの RIIC エラー (EEI)、受信バッファ フル (RXI)、送信バッファ エンプティ (TXI)、および送信完了 (TEI) 割り込みは、ユーザーがコールバックを使用するかどうかにかかわらず、BSP で有効にする必要があります。
- 上記の割り込みすべてに同じプライオリティレベルを設定します。割り込みを異なる優先度に設定すると、適切に動作しなくなる可能性があります。
- RIIC および RIIC_Slave モジュールを同じボード上で使用する場合は、TXI、TEI、EI に同じ割り込み優先順位を設定し、RXI の割り込み優先順位をそれらよりも高く設定することを推奨します。

これは I2C RIIC スレーブドライバーの最初のバージョンであり、基本的な機能だけが実装されています。以下の制限事項があることがわかっています。

1) 次のいずれかの動作が発生した場合、ドライバーは I2C バスをロックアップします。
   - マスタが読み取っているバイト数が "M" であるのに対し、スレーブが書き込むことのできるバイト数が "N" であるとき。この場合、（M < N）または（M > N）です。
   - マスタとスレーブの両方が同時にバスに書き込んでいるとき。
   - マスタとスレーブの両方が同時にバスから読み取っているとき。

   注 : バスのロックアップ状態を解決するには、構成の際にコールバックを使用することが推奨されます。アプリケーションは、コールバックの待機に対するタイムアウトを選択でき、タイムアウト後にスレーブモジュールのインスタンスを閉じて、その後の操作のために再度開くことができます。
2) サポートされている IIC チャネルであればどれでも、マスタとスレーブのどちらのモードの動作にも構成できますが、両方に構成することはできません。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.23.4 アプリケーションへの I2C スレーブ HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに I2C スレーブ HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

2C スレーブドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（I2C スレーブドライバーのデフォルト名は g_i2c0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

I2C スレーブ HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_i2c0 I2C Slave Driver on r_riic_slave	Threads	New Stack> Driver> Communications> I2C Slave Driver on r_riic_slave

次の図に示すように、r_riic_slave の I2C スレーブドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

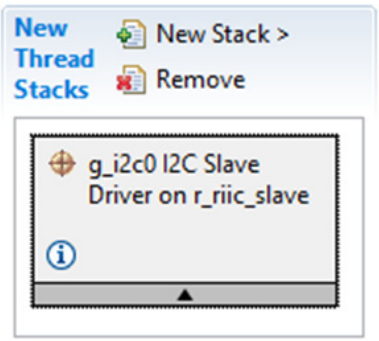


図 226:I2C スレーブ HAL モジュールのスタック

4.2.23.5 I2C スレーブ HAL モジュールの構成

ユーザーは必要な動作に合わせて I2C スレーブ HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_riic_slave 上の I2C スレーブ HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.
Name	g_i2c0	Module name.
Channel	0	Specify the IIC channel to be used with this configuration.
Rate	Standard, Fast-mode, Fast-mode plus Default: Standard	Transfer rate to which the IIC peripheral should be configured to operate.
Slave Address	0x00	Set the address of the device as the I2C slave address. Both 7-bit and 10-bit addresses are supported.
Address Mode	7-Bit, 10 Bit Default: 7-Bit	Address mode selection.
Callback	NULL	A user callback function can be registered in . If this callback function is provided, it will be called from the interrupt service routine (ISR) for each of the conditions defined in i2c_event_t. Warning: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.

ISDE Property	Value	Description
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Receive interrupt priority selection
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit interrupt priority selection
Error Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Error interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

I2C スレーブ HAL モジュールのクロック構成

RIIC 周辺モジュールは、PCLKB をそのクロックソースとして使用します。

I2C スレーブ HAL モジュールのピン構成

RIIC 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

注 : 一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号および必要な MCU ピンが決定されます。

r_riic_slave 上の I2C スレーブ HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
I2C	Pins	Select Peripherals > Connectivity: IIC > IIC0

注 : この選択シーケンスでは、IIC0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_riic_slave 上の I2C スレーブ HAL モジュールのピン構成設定

Pin Configuration Property	Value	Description
Pin Group Selection	_A only, _B only, Mixed (Default: _A only)	Select Simple I2C as the Operation Mode for I2C on SCI
Operation Mode	Enabled, Disabled (Default: Disabled)	Enable or disable peripheral module
SDA	None, P401, P407 (Default: None)	SDA Pin
SCL	None, P400, P204 (Default: None)	SCL Pin

注 : 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと Synergy MCU では、使用可能なピン構成設定が異なる場合があります。

4.2.23.6 アプリケーションでの I2C スレーブ HAL モジュールの使用

アプリケーションで I2C スレーブ HAL モジュールを使用するときの一般的な手順は次のとおりです。

1) open API を使用して、I2C スレーブ HAL モジュールを初期化して開きます。
2) masterReadSlaveWrite API を使用して、マスタにデータを転送します。
3) masterWriteSlaveRead API を使用して、マスタからデータを受信します。
4) close API を使用して、チャネルを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

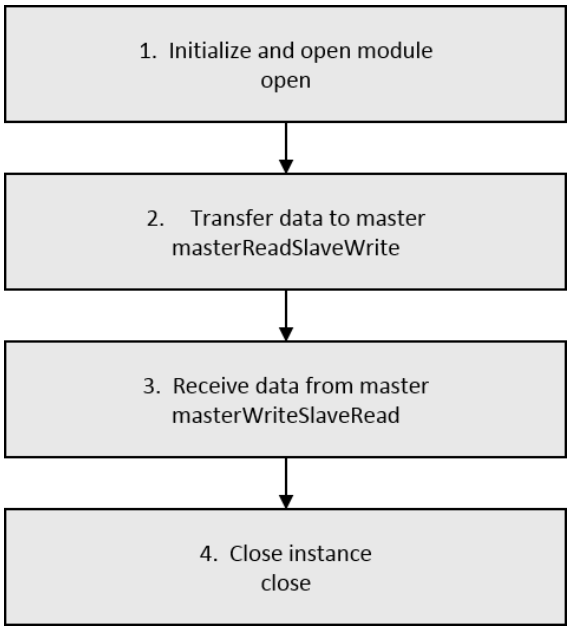


図 227:一般的な I2C スレーブ HAL モジュールアプリケーションのフロー図

4.2.24 I2S ドライバー

I2S HAL モジュールは、標準的な I2S オーディオシリアル通信プロトコル用のハイレベル API を提供します。マスタモードでの非圧縮オーディオデータの送受信に使用されます。

4.2.24.1 I2S HAL モジュールの特長

I2S マスタモードの SSI ペリフェラルで使用される I2S HAL モジュールは、(標準的な I2S プロトコルに加えて) 次の機能をサポートしています。

- 全二重 I2S 通信 (SSI チャネル 0 のみ)
- 割り込み駆動型のデータ送信と受信
- DTC 転送モジュールとの統合。
- 追加データの必要性に応じて作成されるユーザー定義のコールバック

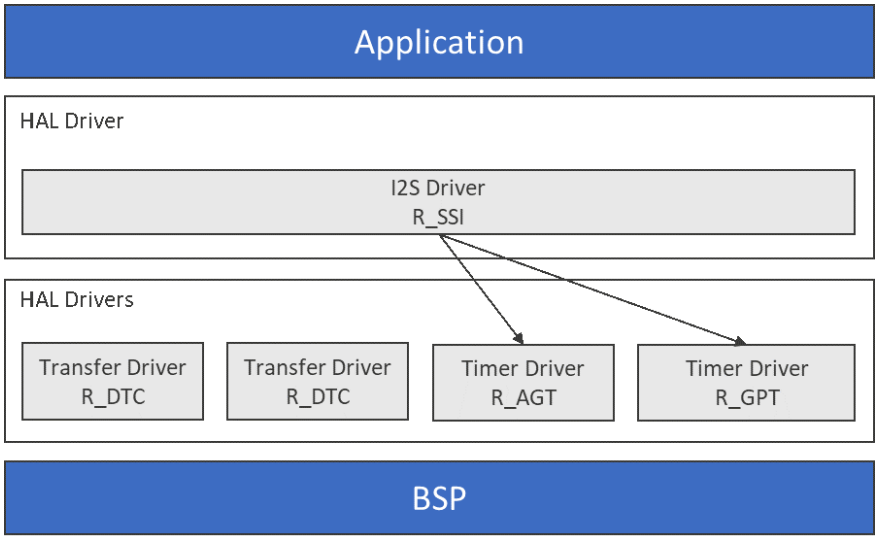


図 228:I2S HAL モジュールのブロック図

4.2.24.2 I2S HAL モジュールの API の概要

I2S HAL モジュールでは、オープン、ミュート、ライト、リードなどの操作のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

I2S HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_i2s0.p_api->open(g_i2s0.p_ctrl, g_i2s0.p_cfg); Initial configuration.
stop	g_i2s0.p_api->stop(g_i2s0.p_ctrl, direction_to_stop); Stop communication. Transmission is stopped when callback is called with I2S_EVENT_IDLE. Reception is stopped when callback is called with I2S_EVENT_RX_EMPTY.

Function Name	Example API Call and Description
mute	g_i2s0.p_api->mute(g_i2s0.p_ctrl, mute_enable); Enable or disable mute.
write	g_i2s0.p_api->write(g_i2s0.p_ctrl, &data, bytes); Write I2S data. All transmit data is queued when callback is called with I2S_EVENT_TX_EMPTY. Transmission is complete when callback is called with I2S_EVENT_IDLE.
read	g_i2s0.p_api->read(g_i2s0.p_ctrl, &data, bytes); Read I2S data. Reception is complete when callback is called with I2S_EVENT_RX_EMPTY.
writeRead	g_i2s0.p_api->writeRead(g_i2s0.p_ctrl, &source, &destination, bytes); Simultaneously write and read I2S data. Transmission and reception are complete when callback is called with I2S_EVENT_IDLE.
infoGet	g_i2s0.p_api->infoGet(g_i2s0.p_ctrl, &info); Get instance specific information and store it in provided pointer info.
close	g_i2s0.p_api->close(g_i2s0.p_ctrl); Allows driver to be reconfigured and may reduce power consumption.
versionGet	g_i2s0.p_api->versionGet(&version); Get version and store it in provided pointer version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successful.
SSP_ERR_OUT_OF_MEMORY	The number of streams open at once is limited to SF_AUDIO_PLAYBACK_CFG_MAX_STREAMS. If this number is exceeded, an out of memory error occurs.
SSP_ERR_TIMEOUT	Timeout occurred before playback finished.
SSP_ERR_ASSERTION	A critical assertion has failed
SSP_ERR_IN_USE	Channel is running/busy
SSP_NOT_OPEN	Requested channel is not configured or API not open

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.24.3 I2S HAL モジュールの動作の概要

I2S HAL モジュールは、I2S プロトコルを使用するオーディオ通信をサポートします。このドライバーは、I2S マスタモードの Synergy MCU 上の SSI 周辺機能をサポートします。圧縮されていないオーディオデータを送信および受信できます。全二重の I2S 通信（SSI チャネル 0 のみ）、割り込み駆動型のデータの送信と受信、および DTC 転送モジュールとの統合を提供します。

I2S HAL モジュールの動作に関する重要な注意事項と制限事項

チャネル 0 でオーディオデータを受信できるようにするには、SSI0 RXI 割り込みを有効にします。チャネル 0 でオーディオデータを送信できるようにするには、SSI0 TXI 割り込みを有効にします。チャネル 0 で送信も受信もできるようにするには、SSI0 TXI と SSI0 RXI の両方の割り込みを有効にします。チャネル 1 で送信または受信できるようにするには、SSIn TXI RXI 割り込みを有効にします。すべての場合に、SSIn INT 割り込みを有効にします。

割り込みが BSP で有効になっている場合、対応する ISR が I2S ドライバーで定義されます。ISR は、open で登録されたユーザーコールバック関数をコールします。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.24.4 アプリケーションへの I2S HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに I2S HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

I2S ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(I2S ドライバーのデフォルト名は g_i2s0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

I2S HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_i2s0 I2S Driver on r_ssi	Threads-> HAL/Common Stacks	New Stack> Driver> Connectivity> I2S Driver on r_ssi

次の図に示すように、r_ssi の I2S ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

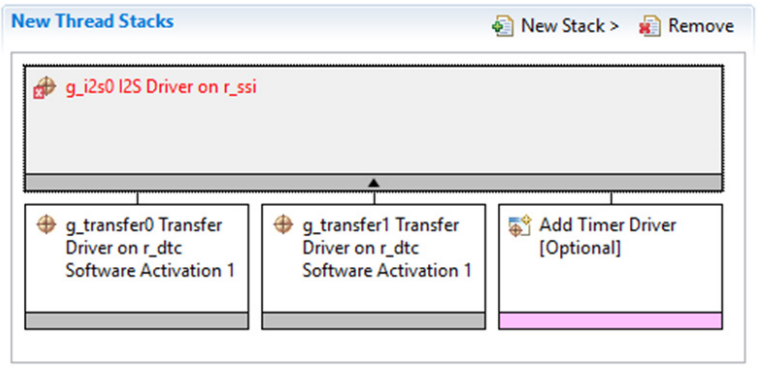


図 229:I2S HAL モジュールのスタック

4.2.24.5 I2S HAL モジュールの構成

ユーザーは必要な動作に合わせて I2S HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の

手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_ssi 上の I2S HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_i2s0	Module name.
Channel	0	Physical hardware channel.
Audio Clock Frequency (Hertz)	2822400	Input audio clock frequency, used to generate the I2S clock. Must be a multiple between 1 and 128 of: (sampling_freq_hz * word_length_in_bits)
Sampling Frequency (Hertz)	44100	Sampling frequency of audio data.
Data Bits	8 bits, 16, 18, 20, 22, 24 Default: 16 bits	Bit depth of audio data, which is the size in bits of one sample of audio data.
Word Length	8 bits, 16, 24, 32 Default: 16 bits	Word length of audio data, must be at least the same size as the bit depth (Data Bits field).
WS Continue Mode	Enabled, Disabled Default: Disabled	Enable WS continue mode to continue to output the word select line when the peripheral is idle. Disable to stop outputting the word select line when the peripheral is idle.

ISDE Property	Value	Description
Name of I2S callback function to be defined by user	NULL	A user callback function must be registered in open. The callback will be called from the interrupt service routine (ISR) when the transmission FIFO reaches the high watermark point after all data for transmission is transmitted or when reception is complete (the requested number of bytes have been received). Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Transmit interrupt priority selection
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Receive interrupt priority selection
Idle/Error Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Priority 12	Idle/error interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、ターゲットハードウェアの実装に基づいた DAC チャネルまたは I2S チャネルの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

I2S HAL モジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_dtc Software Activation の転送ドライバーの構成可能な設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection.
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Driver name.
Mode	Normal	Mode selection.
Transfer Size	4 Bytes	Transfer size selection.
Destination Address Mode	Fixed	Destination address mode selection.
Source Address Mode	Incremented	Source address mode selection.
Repeat Area (Unused in Normal Mode	Source	Repeat area selection.
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection.

ISDE Property	Value	Description
Destination Pointer	NULL	Destination pointer selection.
Source Pointer	NULL	Source pointer selection.
Number of Transfers	0	Number of transfers selection.
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection.
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events Default: Software Activation 1	Activation source selection.
Auto Enable	False	Auto enable selection.
Callback (Only valid with Software start)	NULL	Callback selection.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC software event interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Software Activation 1 の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection.

ISDE Property	Value	Description
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer1	Driver name.
Mode	Normal	Mode selection.
Transfer Size	4 Bytes	Transfer size selection.
Destination Address Mode	Incremented	Destination address mode selection.
Source Address Mode	Fixed	Source address mode selection.
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection.
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection.
Destination Pointer	NULL	Destination pointer selection.
Source Pointer	NULL	Source pointer selection.
Number of Transfers	0	Number of transfers selection.
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection.
Activation Source (Must enable IRQ)	Software Activation 1, Software Activation 2, Peripheral Events Default: Software Activation 1	Activation source selection.
Auto Enable	False	Auto enable selection.
Callback (Only valid with Software start)	NULL	Callback selection.

ISDE Property	Value	Description
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC software event interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_agt 上のタイマドライバーの構成可能な設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection
Name	g_timer0	Module name
Channel	0	Channel selection
Mode	Periodic	Mode selection
Period Value	2822400 *2	Period value selection
Period Unit	Hertz	Period unit selection
Auto Start	FALSE	Auto start selection
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub Default: PCLKB	Count source selection
AGTO Output Enabled	True, False Default: False	AGTO output selection

ISDE Property	Value	Description
AGTIO Output Enabled	True, False Default: False	AGTIO output selection
Output Inverted	True, False Default: False	Output inverted selection
Enable comparator A output pin	True, False Default: False	Enable comparator A output pin selection
Enable comparator B output pin	True, False Default: False	Enable comparator B output pin selection
Callback	NULL	Callback selection
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_gpt 上のタイマドライバーの構成可能な設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection
Name	g_timer0	Module name
Channel	0	Channel selection

ISDE Property	Value	Description
Mode	Periodic	Mode selection
Period Value	2822400 *2	Period value selection
Period Unit	Hertz	Period unit selection
Duty Cycle Value	50	Duty cycle value selection
Duty Cycle Unit	Unit Raw Counts, Unit Percent, Unit Percent x 1000 Default: Unit Raw Counts	Duty cycle unit selection
Auto Start	FALSE	Auto start selection
GTIOCA Output Enabled	True, False Default: False	GTIOCA output enabled selection
GTIOCA Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCA stop level selection
GTIOCB Output Enabled	True, False Default: False	GTIOCB output enabled selection
GTIOCB Stop Level	Pin Level Low, Pin Level High, Pin Level Retained Default: Pin Level Low	GTIOCB stop level selection
Callback	NULL	Callback selection
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

I2S HAL モジュールのクロック構成

SSI モジュールでは、[Clock configuration] ウィンドウにある周辺クロック (PCLKB) を使用します。また、AUDIO_CLK ピンへの外部クロック入力も使用します。

I2S HAL モジュールのピン構成

SSI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。

選択した SSI チャネルに対し、SSI RX ピンと SSI TX ピンのどちらか一方または両方を構成します (【Pins】 タブ > 【Peripherals】 > 【SSI】 > 【SSIn】 > 【SSITXD0】/【SSIRXD0】)。チャネル 0 の場合は、これらのピンのうちいずれかまたは両方を有効化します。チャネル 1 の場合は、SSIDATA1 ピンを有効化します。

ワード選択およびクロック ピンを設定します (【ピン】 タブ > 【周辺機器】 > 【SSI】 > 【SSIn】 > 【SSITWSn および SSISCKn】)。このピンは大抵、どちらとも必要となります。必要なピンについては、使用する I2S デバイスのデータシートを確認してください。

オーディオクロックピンを SSI 用に設定します (【Pins】 タブ >【Peripherals】>【SSI】> 【SSIO_SSI1_AUDIO_CLK】)。外部オーディオ クロックをこのクロック入力ピンに接続します。オーディオ クロックの作成に GPT タイマを使用する場合は、GPT タイマ出力ピンを設定し (【ピン】 タブ > 【周辺機器】 > 【GPT1】 > 【GPTn】 > 【GTIOCx】)、使用する GPT 出力ピンをオーディオ クロック入力ピンに接続します。

次の表は 【SSP configuration】 ウィンドウ内でのピンの選択方法を示し、その次の表は関連するピンの選択例を示しています。

r_ssi 上の I2S HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
SSI	Pins	Select Peripherals > Connectivity:SSI

注 : この選択シーケンスでは、SSI0 がドライバーに必要なハードウェアターゲットであることを想定しています。

SSI 上の I2S ドライバーのピンの構成可能な設定

Pin Configuration Property	Settings	Description
Pin Group Selection	_A only, Mixed	Pin group for I2S port.
Operation Mode	Custom, Disabled	Operation selection.
SSISCK	None, P400 Default: None	AUDIO_CLK pin(P400), used in this application

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

SSI0 上の I2S ドライバーのピン構成設定

Pin Configuration Property	Settings	Description
Pin Group Selection	_A only, _B only, Mixed	Pin group for I2S port.
Operation Mode	Enabled, Custom, Disabled	Operation selection.
SSISCK	None, P112 Default: None	SSISCK pin(P112), used in this application
SSIWS	None, P113 Default: None	SSIWS pin(P113), used in this application
SSITXD	None, P115 Default: None	SSITXD pin(P115), used in this application
SSIRXD	None, P114 Default: None	SSIRXD pin(P114), used in this application

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

SSI1 上の I2S ドライバーのピン構成設定

Pin Configuration Property	Settings	Description
Pin Group Selection	_A only, _B only, Mixed	Pin group for I2S port.
Operation Mode	Enabled, Custom, Disabled	Operation selection.

Pin Configuration Property	Settings	Description
SSISCK	None, P204 Default: None	SSI Serial Clock, not used in this application.
SSIWS	None, P205 Default: None	SSI Stereo pin selection, not used in this application.
SSIDATA	None, P206 Default: None	SSI Data pin selection, not used in this application.

注 : 例の値は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.2.24.6 アプリケーションでの I2S HAL モジュールの使用

アプリケーションで I2S HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、I2S HAL モジュールを開きます。
2) write API を使用して、I2S バスにオーディオデータを書き込みます。
3) I2S_EVENT_TX_EMPTY でのコールバックを待ってから、ソースバッファを解放します。
4) read API を使用して、I2S バスからデータを読み取ります。
5) I2S_EVENT_RX_FULL でのコールバックを待ってから、宛先バッファにアクセスするか、次のバッファを読み取ります。
6) 必要に応じて、アプリケーションで他の API を使用します。
7) close API を使用して、モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

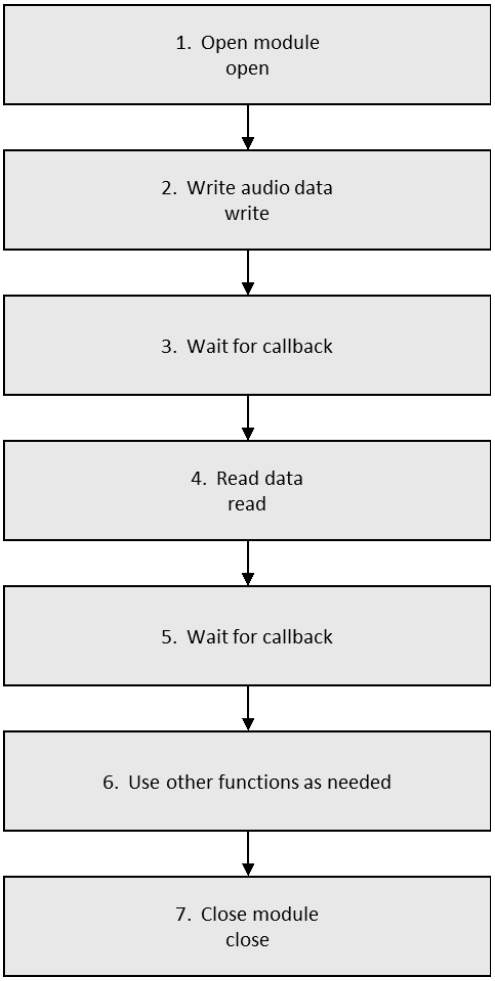


図 230:一般的な I2S HAL モジュールアプリケーションのフロー図

4.2.25 入力キャプチャ ドライバー

入力キャプチャ HAL モジュールでは、入力のパルス幅およびパルス期間を測定するための API が提供されます。また、入力キャプチャ HAL モジュールは、Synergy MCU 上の GPT ペリフェラルで使用する入力キャプチャパラメータを構成します。ユーザー定義のコールバックを作成し、新しい測定が完了するたびに値を取得することができます。

4.2.25.1 入力キャプチャ HAL モジュールの特長

入力キャプチャ HAL モジュールは、入力キャプチャ機能用の GPT を構成します。

- 入力キャプチャ HAL を使用すると、次のタスクを実行できます。
  - モジュールを初期化する
  - 入力キャプチャの測定を有効にする

- 入力キャプチャの測定を無効にする
- 測定カウンタの状態（実行中かどうか）を取得する
- 最後にキャプチャされたタイマ / オーバーフローカウンタの値を取得する
- 入力キャプチャ動作を閉じる

• 入力キャプチャ HAL モジュールは以下のものをサポートします。

- パルス幅の測定とパルス期間の測定
- 立ち上がりエッジまたは立ち下がりエッジの測定の開始
- ワンショットモードまたは周期モード
- キャプチャを有効にするためのハードウェアイネーブル信号（ローイネーブル / ハイイネーブル）
- 以下のイベントでのコールバック関数 :
- カウンタオーバーフロー
- 入力キャプチャ発生
- 割り込みイベントに関するデータを提供するコールバック構造体 (input_capture_callback_args_t)。発生した割り込みおよび関連するカウンタ値を含みます。

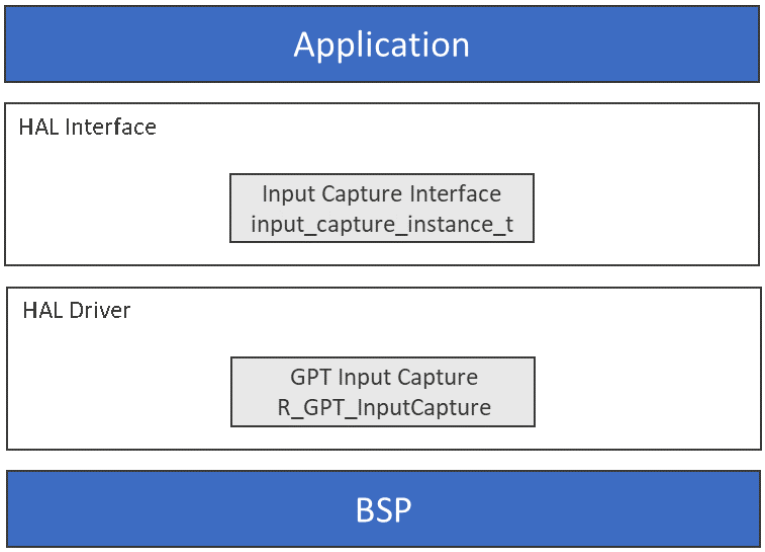


図 231:**入力キャプチャ HAL モジュールのブロック図**

4.2.25.2 入力キャプチャ HAL モジュールの API の概要

入力キャプチャ HAL モジュールインタフェースでは、オープン、クローズ、有効化、無効化、状態情報へのアクセス、入力キャプチャでの汎用 PWM タイマ（GPT）を使用した最新のキャプチャ値へのアクセスのための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は、HAL モジュールの API 要約の後にあります。

入力キャプチャ HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_input_capture.p_api->open(g_input_capture.p_ctrl, g_input_capture.p_cfg); Opens the Input Capture HAL and initializes configuration.
close	g_input_capture.p_api->close(g_input_capture.p_ctrl); Closes the input capture operation. Allow drive to be reconfigured, and may reduce power consumption.
enable	g_input_capture.p_api->enable(g_input_capture.p_ctrl); Enables input capture measurement.
disable	g_input_capture.p_api->disable(g_input_capture.p_ctrl); Disables input capture measurement.
infoGet	g_input_capture.p_api->infoGet(g_input_capture.p_ctrl, &input_capture_info); Gets the status (running or not) of the measurement counter.
lastCaptureGet	g_input_capture.p_api->lastCaptureGet(g_input_capture.p_ctrl, &input_capture_counter); Gets the last captured timer/overflows counter value.

Function Name	Example API Call and Description
versionGet	g_input_capture.p_api->versionGet(&input_capture_version); Retrieve the API version with the input_capture_version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_ASSERTION	One of the parameters is NULL. Or the channel requested in the p_cfg parameter may not be available on the device selected in r_bsp_cfg.h. Or, p_cfg->mode is invalid.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value. Or ISR is not enabled.
SSP_ERR_IN_USE	Attempted to open an already open device instance.
SSP_ERR_NOT_OPEN	The channel is not opened.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.25.3 入力キャプチャ HAL モジュールの動作の概要

入力キャプチャHAL モジュールは、ユーザーの構成に従って、Synergy マイクロコントローラ上の GPT HAL モジュールユニットを制御します。RTOS 要素を使用しないで GPT ハードウェアに直接アクセスし、開発が簡単になる便利な API を提供します。

コールバックを使用できる場合は（割り込みが有効）、通常の測定が完了すると、入力キャプチャ HAL モジュールはコールバックを呼び出して引数 input_capture_callback_args_t を渡します。

引数 input_capture_callback_args_t では、チャネル、input_capture_event_t イベント、割り込み発生時にキャプチャされたタイマの値、この測定の間に発生したカウンタオーバーフローの回数が示されています。

割り込みが有効になっていない場合、API によって読み取られる値は、最後にキャプチャされたタイマ / オーバーフローカウンタの値になります。

入力キャプチャ HAL モジュールの動作に関する重要な注意事項と制限事項

入力キャプチャ測定モードを取得する

入力キャプチャインタフェースでは、ワンショット測定および周期的な測定のサポートが提供されています。GPT ハードウェアは、ワンショット機能をネイティブにはサポートしていません。タイマを停止およびクリアするために、コードが自動的に割り込みサービスルーチン（ISR）に組み込まれます。そのため、コールバックを使用しない場合でも、ワンショットモードのために ISR を有効にする必要があります。

GPT 入力キャプチャ信号

入力キャプチャ測定は、入力キャプチャ信号ピン（GTIOCA/GTIOCB）で入力キャプチャ信号エッジ（立ち下がりまたは立ち下がり）が検出され、かつイネーブル条件が満たされていると、開始されます。イネーブル条件は、イネーブルレベルによって定義されており、無効化するか（なし）、入力キャプチャイネーブルピン（GTIOCA/GTIOCB）のレベル（Low または High）を指定することができます。入力キャプチャイネーブルピンは、入力キャプチャ信号ピンとして使用されていないピンです。

測定カウントの時間への変換

測定が完了すると、未加工のカウントデータとオーバーフロー数がコールバック関数でユーザーに返されます。

必要に応じて、未加工の測定データをコールバックまたはユーザーのアプリケーション内で論理的な時間単位に変換できます。未加工のデータを変換するには、現在の PCLKD クロック周波数とプリスケーラの値、オーバーフロー数、最大カウンタ値、測定カウントを考慮する必要があります。測定カウントとオーバーフロー数は、コールバック引数 input_capture_callback_args_t で提供されます。

現在の PCLKD 周波数を取得するには、systemClockFreqGet API を使用する方法が推奨されます。入力クロック周波数は、PCLKD 周波数をプリスケーラの値で分周したものです。次に示す「入力キャプチャの時間計算」の表では、clk_freq_hz として表されます。

S7G2（全チャネル）、S3A7（全チャネル）、S124（チャネル 0）の最大カウンタ値は、0xFFFFFFFF です。S124 (チャネル 1-6) の最大カウンタ値は 0xFFFF です。次の表では、この最大カウンタ値に 1 を加えた値（カウンタは 0 から始まるため）が max_counts として示されています。

入力キャプチャの時間計算

Desired Time Units	Formula
Nanoseconds (ns)	time_ns = ((overflows * max_counts) + counter) * 1000000000 / clk_freq_hz
Microseconds (us)	time_ns = ((overflows * max_counts) + counter) * 1000000 / clk_freq_hz
Milliseconds (ms)	time_ns = ((overflows * max_counts) + counter) * 1000 / clk_freq_hz
Seconds (s)	time_ns = ((overflows * max_counts) + counter) / clk_freq_hz

- 現在、入力キャプチャ HAL モジュールがサポートしているのはパルス幅の測定とパルス期間の測定だけです。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.25.4 アプリケーションへの入力キャプチャ HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに入力キャプチャ HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

入力キャプチャドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(入力キャプチャドライバーのデフォルトの名前は g_input_capture0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

入力キャプチャ HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_input_capture Input Capture Driver on r_gpt_input_capture	Threads-> HAL/Common Stacks	New Stack > Driver > Timers > Input Capture Driver on r_gpt_input_capture

次の図に示すように、r_gpt_input_capture の入力キャプチャドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

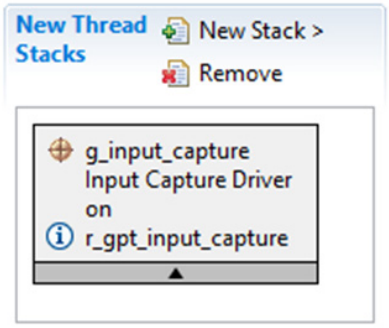


図 232:**入力キャプチャ HAL モジュールのスタック**

4.2.25.5 入力キャプチャ HAL モジュールの構成

ユーザーは必要な動作に合わせて入力キャプチャ HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_gpt_input_capture 上の入力キャプチャ HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_input_capture	Module name.
Channel	0	Physical hardware channel.
Mode	Pulse Width, Period Default: Pulse Width	Measures inputs from the Signal edge until the opposite edge.
Signal Edge	Rising, Falling Default: Rising	Start measurement on rising or falling edge. Measurement stops on the opposite edge.
Repetition	Periodic, One Shot Default: Periodic	Capture a single measurement, then disable captures (one shot) until enable is called, or capture measurements continuously (periodic).
Auto Start	True, False Default: True	Set to true to enable measurements after configuring or false to leave the measurements disabled until enable is called.

ISDE Property	Value	Description
Callback	NULL	A user callback function must be registered in open. The callback will be called from the interrupt service routine (ISR) each time the timer period elapses. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Input Capture Signal Pin	GTIOCA, GTIOCB Default: GTIOCA	Select the input pin used to trigger the start of a measurement.
GTIOCx Signal Filter	None, PCLK/1, PCLK/4, PCLK/16, PCLK/64 Default: None	The noise filter samples the external signal at intervals of the PCLK divided by one of the values. When 3 consecutive samples are at the same level (high or low), that level is passed on as the observed state of the signal.
Clock Divider	PCLK/1, PCLK/4, PCLK/16, PCLK/64, PCKL/256, PCLK/1024 Default: PCLK/1	Clock divider used to scale the measurement counter.

ISDE Property	Value	Description
Input Capture Enable Level	None, Low, High Default: None	Each GPT channel has 2 I/O pins (GTIOCA and GTIOCB). One must be selected as the Input Capture Signal Pin. The other GPT I/O pin can be used as a hardware enable signal to enable captures. Select None and captures are always enabled, select low and captures are enabled only while the enable input pin is low, select high and captures are enabled only while the enable input pin is high.
Input Capture Enable Filter	None (No filtering), PCLK/1 (Fast sampling), PCLK/4, PCLK/16, PCLK/64 (Slow sampling) Default: None (No filtering)	The enable filter samples the enable signal at intervals of the PCLK divided by one of the values. When 3 consecutive samples are at the same level (high or low), that level is passed on as the observed state of the signal.
Capture Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Capture interrupt priority selection.
Overflow Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Overflow interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

入力キャプチャ HAL モジュールのクロック構成

GPT HAL モジュールは、PCLKD をそのクロックソースとして使用します。PCLKD の周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clocks] タブを使用するか、実行時に CGC インタフェースを使用します。

入力キャプチャ HAL モジュールのピン構成

特定のチャネルとピンにアクセスするには、ISDE の [Pins] タブで GTIOCx ピンを設定する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は GTIOCx ピンの選択例を示しています。

r_gpt_input_capture 上の入力キャプチャ HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
GPT Input Capture	Pins	Select Peripherals > Timer: GPT > GPT0

注 : この選択シーケンスでは、GPT0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_kint 上の入力キャプチャ HAL モジュールのピン構成設定

Property	Value	Description
Pin Group Selection	Mixed, _A Only, _B Only Default: Mixed	Pin grouping selection
Operation Mode	Disabled, GTIOCA or GTIOCB, GTIOCA and GTIOCB Default: Disable	Select GTIOCA or GTIOCB as the Operation Mode for Input Capture on GPT
GTIOCA	None, P300, P512 Default: None	GTIOCA Pin
GTIOCB	None, P108, P511 Default: None	GTIOCB Pin

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

4.2.25.6 アプリケーションでの入力キャプチャ HAL モジュールの使用

アプリケーションで入力キャプチャ HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、モジュールを初期化します。
2) lastCaptureGet API を使用して、メインループルーチン内で、またはコールバック関数内で目的の値を見つけることができます。
3) disable API を使用して、キャプチャ割り込みとオーバーフロー割り込みを無効化し、タイマを停止することができます。
4) enable API を使用して、キャプチャ割り込みとオーバーフロー割り込みを有効にし、タイマを開始することができます。
5) infoGet API を使用して、キャプチャされているカウンタの状態（実行中または停止）を照会できます。
6) 終了したら、close API を使用してモジュールを閉じることができます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

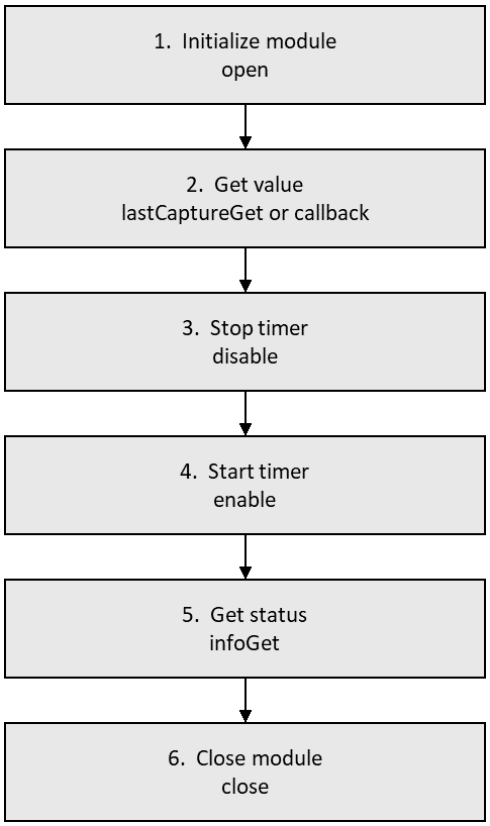


図 233:通常の入力キャプチャ HAL モジュールアプリケーションのフロー図

4.2.26 I/O ポート ドライバー

I/O ポート HAL モジュールは、I/O ピンの制御、ボードのピンの構成、および I/O ピンの操作を行うためのハイレベルの API を実装します。I/O ピンの動作状態は、Synergy コンフィギュレータを使用して設定できます。Synergy プロジェクトが構築されると、ピン構成ファイルが作成されます。アプリケーションが実行されると、BSP がこのドキュメントで説明されているものと同じ API 関数を使用して、MCU の I/O ポートを適切に構成します。

4.2.26.1 I/O ポート HAL モジュールの特長

I/O ポート HAL モジュールは、次の機能を実行できます。

- 2 つのブロック間にイベント リンクを作成します。
- その 2 つのブロック間のイベント リンクを切断します。
- CPU に割り込む 2 つのソフトウェアイベントのうち 1 つを生成します。

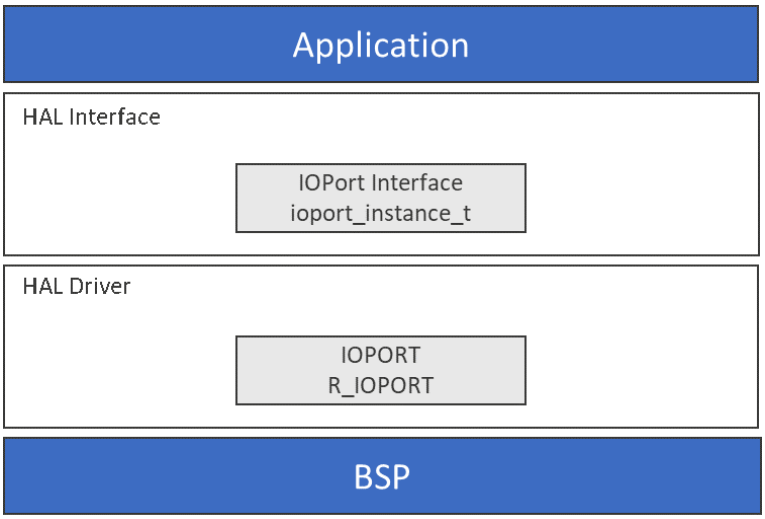


図 234:I/O ポート HAL モジュールのブロック図

4.2.26.2 I/O ポート HAL モジュールの API の概要

I/O ポート HAL モジュールでは、特定のピンとポートのリードおよびライトを行うための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

I/O ポート HAL モジュールの API の要約

Function Name	Example API Call and Description
init	g_ioport.p_api->init(g_ioport.p_cfg); Initialize configuration of multiple pins.

Function Name	Example API Call and Description
pinCfg	g_ioport.p_api->pinCfg(IOPORT_PORT_00_PIN_00, IOPORT_CFG_IRQ_ENABLE OR IOPORT_CFG_PORT_DIRECTION_INPUT); Configure settings for an individual pin.
pinDirectionSet	g_ioport.p_api->pinDirectionSet(IOPORT_PORT_00_PIN_00, IOPORT_DIRECTION_INPUT); Set the pin direction of a pin.
pinEventInputRead	g_ioport.p_api->pinEventInputRead(IOPORT_PORT_00_PIN_00, &pin_level); Read the event (ELC) input data of the specified pin and return the level.
pinEventOutputWrite	g_ioport.p_api->pinEventOutputWrite(IOPORT_PORT_00_PIN_00, IOPORT_PIN_LEVEL_HIGH); Write pin event (ELC) data.
pinEthernetModeCfg	g_ioport.p_api->pinEthernetModeCfg(IOPORT_ETHERNET_CHANNEL_0, IOPORT_ETHERNET_MODE_MII); Configure the PHY mode of the Ethernet channels.
pinRead	g_ioport.p_api->pinRead(IOPORT_PORT_00_PIN_00, &pin_level); Read level of a pin.
pinWrite	g_ioport.p_api->pinWrite(IOPORT_PORT_00_PIN_00, IOPORT_PIN_LEVEL_HIGH); Write specified level to a pin.

Function Name	Example API Call and Description
portDirectionSet	g_ioport.p_api->portDirectionSet(IOPORT_PORT_00, direction_values, mask); Set the direction of one or more pins on a port.
portEventInputRead	g_ioport.p_api->portEventInputRead(IOPORT_PORT_00, &pin_levels); Read captured event (ELC) data for a port.
portEventOutputWrite	g_ioport.p_api->portEventOutputWrite(IOPORT_PORT_00, pin_levels, mask); Write event (ELC) output data for a port.
portRead	g_ioport.p_api->portRead(IOPORT_PORT_00, &pin_levels); Read states of pins on the specified port.
portWrite	g_ioport.p_api->portWrite(IOPORT_PORT_00, pin_levels, mask); Write to multiple pins on a port.
versionGet	g_ioport.p_api->versionGet(&version); Retrieve version information using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_INVALID_ARGUMENT	The port/pin/mask/direction/level (etc.) not valid.
SSP_ERR_ASSERTION	Unexpected null pointer.
SSP_ERR_UNSUPPORTED	Feature not supported – for instance Ethernet configuration not supported on the device.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.26.3 I/O ポート HAL モジュールの動作の概要

I/O ポート HAL モジュールは、ビットとポートの両方のレベルでデバイスの I/O ポートにアクセスするための機能を提供します。ポートとピン両方の方向を変更できます。さらに、個々のピンの機能を変更するために数々の構成 API が提供されています。

I/O ポート HAL モジュールでは、ピンを構成するために以下の動作が提供されています。

- ドライバーを初期化します – 次のコールによって実行されます：init
  - パラメーター チェックを実行して、エラー状態を処理します。
  - VBATT ドメインのピン構成を処理します。
  - ピンの PFS レジスタを書き込みます。
- ピンを構成します – pinCfg API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ピン番号 pin、VBATT のサポートのチェック）。
  - ピンの PFS レジスタに書き込みます。
- ピンレベルを読み取ります – pinRead API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ピン番号 pin のチェック）。
  - ピンの PFS レジスタを読み取ります。
- ポートのすべてのピンレベルを読み取ります – portRead API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ポート番号 port のチェック）。
  - 指定されたポートの PCNTR レジスタ値の現在の値を読み取ります。
- ピンレベルを書き込みます – pinWrite API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ピン番号 pin および書き込まれるレベル level のチェック）。

  - ピンの PFS レジスタに書き込みます。
- ポートの複数のピンレベルを書き込みます – portWrite API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ポート番号 port およびピンマスク mask のチェック）。
  - 指定されたポートの PCNTR レジスタから現在の構成を読み取ります。
  - マスクに従って指定されたポートの PCNTR レジスタにピンレベルを書き込み、スコープ外のピンレベルを保持します。
- ポートの複数のピンの方向を設定します – portDirectionSet API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ポート番号 port およびピンマスク mask のチェック）。
  - 指定されたポートの PCNTR レジスタから現在の構成を読み取ります。
  - マスクに従って指定されたポートの PCNTR レジスタにピンレベルを書き込み、スコープ外のピンの方向を保持します。
- ピンの方向を書き込みます – pinDirectionSet API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ピン番号 pin および書き込まれる方向 direction のチェック）。
  - ピンの PFS レジスタに書き込みます。
- イベント（ELC）ポートの入力を読み取ります – portEventInputRead API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ポート番号 port のチェック）。
  - 指定されたポートの PCNTR レジスタ値の現在の値を読み取ります。
- イベント（ELC）ピンの入力を読み取ります – pinEventInputRead API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ピン番号 pin のチェック）。
  - 指定されたピンのポートの PCNTR レジスタ値の現在の値を読み取ります。
  - PCNTR レジスタの値にピンマスクを適用することにより、ピンレベルを取得します。
- イベント（ELC）ポートの出力を書き込みます – portEventOutputWrite API をコールすることによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ポート番号 port およびピンマスク mask_value). のチェック）。
  - 指定されたポートの PCNTR レジスタから現在の構成を読み取ります。
  - マスクに従って指定されたポートの PCNTR レジスタにピンレベルを書き込み、イベントスコープ外のピンレベルを保持します。
- イベント（ELC）ピンの出力を書き込みます – pinEventOutputWrite API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（ピン番号 pin および書き込まれるレベル pin_value). のチェック）。
  - イベントスコープ外のピンレベルを保持するマスクに従って指定されたピンのポートの PCNTR レジスタにピンレベルを書き込みます。

- イーサネットチャネルの PHY モードを構成します – pinEthernetModeCfg API を呼び出すことによって実行されます。
  - パラメータチェックを実行して、エラー状態を処理します（イーサネットチャネル channel およびモード mode のチェック）。
  - イーサネットコントロールレジスタ（PFENET）を更新します。

I/O ポート HAL モジュールの動作に関する重要な注意事項と制限事項

- ポートの特定のピンのリードやライトを行うには、16 ビットのビットマスクを適用する必要があります。ポートには、0（LSB）から 15（MSB）までの番号が付いています。
- イーサネットポートのない MCU で pinEthernetModeCfg API 関数を使用してイーサネットポートを構成すると（RMII または MII 出力フォーマットの選択）、サポートされていないことを示すエラーメッセージが返されます。これを避けるには、開発者はターゲット MCU でイーサネットポートが使用できるかどうかを確認する必要があります。この情報は、ターゲット MCU ハードウェアのユーザーズマニュアルで簡単に見つけることができます。開発を開始する前に、ターゲット MCU でイーサネットペリフェラルが使用可能かどうかを確認するだけです。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.26.4 アプリケーションへの I/O ポート HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに I/O ポート HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

I/O ポートドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(I/O ポートドライバーのデフォルト名は g_ioport0. ですこの名前は、対応する [Properties] ウィンドウで変更できます。)

I/O ポート HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ioport I/O Port driver on r_ioport	Threads	Highlight HAL/Common and select New > Driver > System > I/O Port Driver on r_ioport

次の図に示すように、r_ioport の I/O ポートドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」とい

うテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

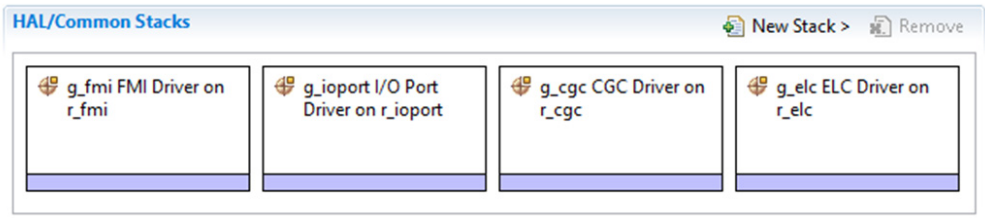


図 235:I/O ポート HAL モジュールのスタック

4.2.26.5 I/O ポート HAL モジュールの構成

ユーザーは必要な動作に合わせて I/O ポート HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_ioport 上の I/O ポート HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_ioport	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

I/O ポート HAL モジュールのクロック構成

I/O ポート HAL モジュールには、特定のクロック構成は必要ありません。

I/O ポート HAL モジュールのピン構成

I/O ポート HAL モジュールに直接関連付けられている、構成の必要なピンはありません。

4.2.26.6 アプリケーションでの I/O ポート HAL モジュールの使用

アプリケーションで I/O ポート HAL モジュールを使用する一般的な手順は次のとおりです。

1) init API を使用して、ドライバーを初期化します。
2) pinCfg API を使用して、ピンを構成します。
3) pinRead および portRead API を使用して、指定したピンとポートから読み取ります。
4) pinWrite および portWrite API を使用して、指定したピンとポートに書き込みます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

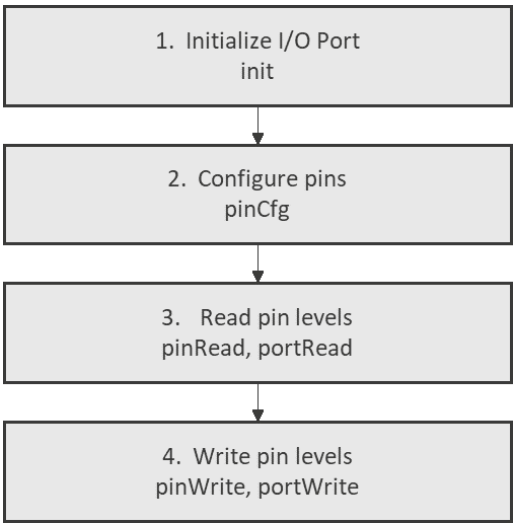


図 236:一般的な I/O ポート HAL モジュールアプリケーションのフロー図

4.2.27 独立ウォッチドッグドライバー

独立ウォッチドッグタイマ (IWDT) HAL モジュールは、ウォッチドッグタイマアプリケーション用のハイレベル API を提供し、Synergy MCU 上の IWDT ペリフェラルを使用します。ユーザー定義のコールバックを作成し、イベント通知に応答できます。

4.2.27.1 独立ウォッチドッグタイマ HAL モジュールの特長

IWDT HAL モジュールには、以下の主要な特長があります。

- WDT が許可されたリフレッシュ ウィンドウの外でアンダーフローまたはリフレッシュされた場合、次のいずれかのイベントが生じる可能性があります。
  - デバイスのリセット
  - NMI の生成

- 内部ウォッチドッグタイマ周辺機能（IWDT）をサポートします。IWDT は独自のクロックソースを持ち、安全性が向上します。
- リセット後の自動ハードウェア構成をサポートします。

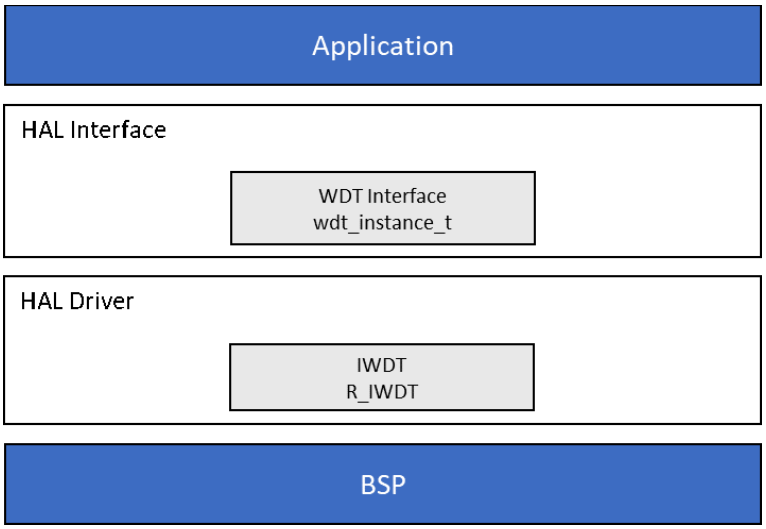


図 237:独立ウォッチドッグタイマ HAL モジュールのブロック図

4.2.27.2 独立ウォッチドッグタイマ HAL モジュールの API の概要

IWDT HAL モジュールでは、状態のオープン、リフレッシュ、リード、取得のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

独立ウォッチドッグタイマ HAL モジュールの API の要約

Function Name	Example API Call and Description
cfgGet	g_wdt0.p_api->cfgGet(g_wdt0.p_ctrl, g_wdt0.p_cfg); Initialize the iWDT in register start mode. In auto-start mode with NMI output it registers the NMI callback.

Function Name	Example API Call and Description
open	g_wdt0.p_api->open(g_wdt0.p_ctrl, g_wdt0.p_cfg); Initialize the iWDT in register start mode. In auto-start mode with NMI output it registers the NMI callback.
refresh	g_wdt0.p_api->refresh(g_wdt0.p_ctrl); Refresh the watchdog timer.
statusGet	g_wdt0.p_api->statusGet(g_wdt0.p_ctrl, &status); Read the status of the iWDT.
statusClear	g_wdt0.p_api->statusClear(g_wdt0.p_ctrl, clear); Clear the status flags of the iWDT.
counterGet	g_wdt0.p_api->counterGet(g_wdt0.p_ctrl, &counter); Read the current iWDT counter value.
timeoutGet	g_wdt0.p_api->timeoutGet(g_wdt0.p_ctrl, &timeout); Read the watchdog timeout values.
versionGet	g_wdt0.p_api->versionGet(&version); Retrieve the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successfully executed.
SSP_ERR_ASSERTION	Null Pointer(s).
SSP_ERR_INVALID_ARGUMENT	One or more configuration options is invalid.
SSP_ERR_INVALID_MODE	An attempt to open the WDT in register-start mode when the OFS0 register is configured for auto-start mode. Or to open the WDT in auto-start mode when the OSF0 is configured for register start mode.
SSP_ERR_ABORTED	Invalid clock divider for this watchdog

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.27.3 独立ウォッチドッグタイマ HAL モジュールの動作の概要

IWDT HAL モジュールは、IWDT インタフェースを構成します。IWDT が許可されたリフレッシュ ウィンドウの外でアンダーフローまたはリフレッシュされた場合、次のいずれかのイベントが生じる可能性があります。

- デバイスのリセット
- NMI の生成

次の図は、IWDT の動作の例を示しています。カウンターの有効なリフレッシュ周期でリフレッシュされた場合、タイマー カウントの値はリセットされます。カウントでアンダーフローまたは有効なリフレッシュ期間外のリフレッシュ発生が許可されている場合、IWDT はデバイスをリセットするか、NMI を生成します。

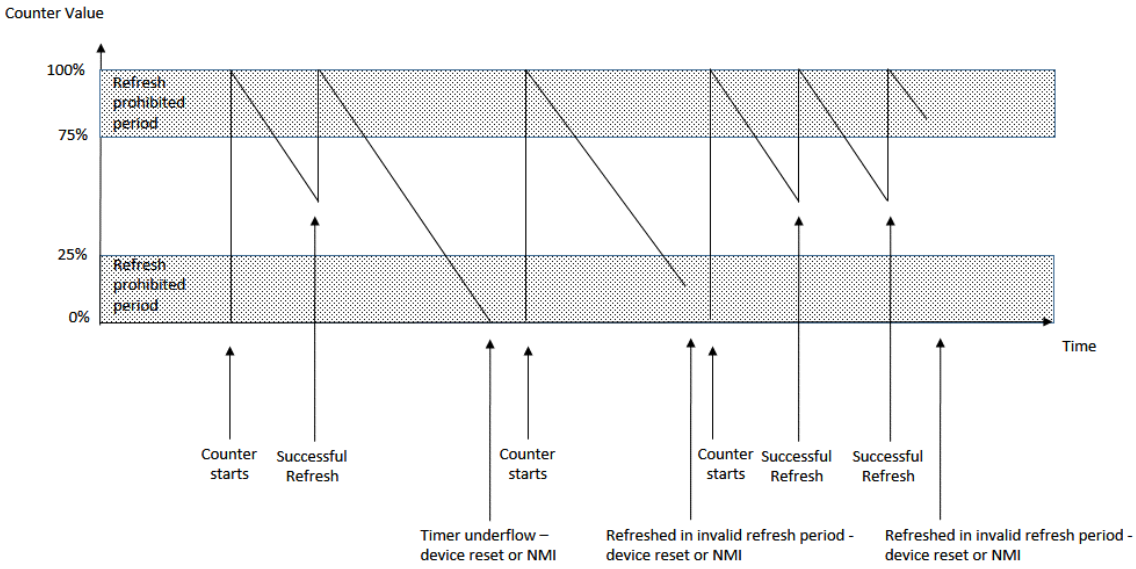


図 238:独立ウォッチドッグタイマ HAL モジュールの動作の図

すべての Synergy マイクロコントローラのシリーズは、オプション設定メモリを備えています。このメモリを使用して、リセット後の周辺機能の動作状態を設定できます。OFS は、IWDT、WDT、LVD、CGC HOCO の状態の設定に使用できます。

次の表では、OFS レジスタで構成できる IWDT のパラメータを示します。

注 :IWDT は、OFS レジスタを使用することによってのみ構成できます。IWDT は、レジスタスタートモードをサポートしていません。

Control	Description
IWDT Start Mode Select	Automatically starts the IWDT after a Reset, if enabled.
IWDT Timeout Period	Specifies the IWDT timeout (number of clock cycles) 128 cycles 512 cycles 1024 cycles 2048 cycles

Control	Description
IWDT-Dedicated Clock Frequency Division Ratio	1 1/16 1/32 1/64 1/128 1/256
IWDT Window End Position	25% 50% 75% 100% (no window end position set)
IWDT Window Start Position	25% 50% 75% 100% (no window start position set)
IWDT Reset Interrupt Request	The IWDT can either generate an Interrupt Signal or a Reset signal.
IWDT Stop Control	The IWDT can continue to count or Stop counting in Low Power Mode.

注 :OFS0 レジスタの内容の詳細については、Synergy MCU ハードウェアマニュアルを参照してください。

OFS レジスタの値は、次の図に示すように Synergy 構成エディタの [BSP] タブのプロパティダイアログで設定します。

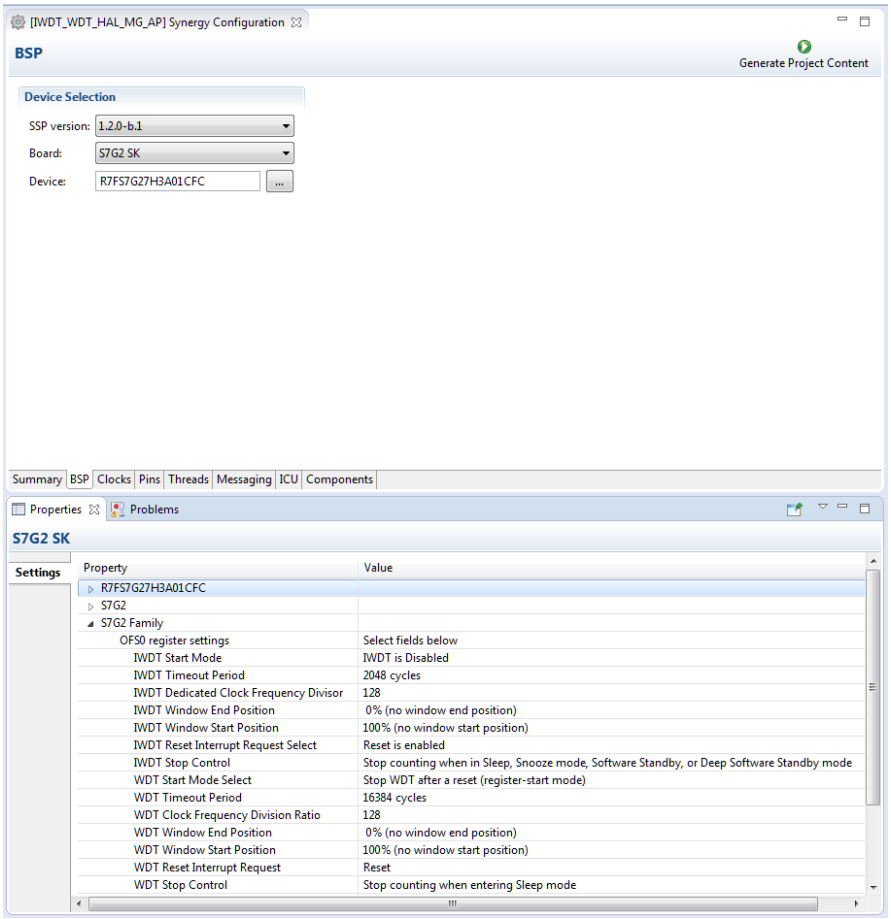


図 239:独立ウォッチドッグタイマ HAL モジュールの構成画面

独立ウォッチドッグタイマ HAL モジュールの動作に関する重要な注意事項と制限事項

IWDT HAL モジュールの期間の計算

IWDT は IWDTCLK から動作します。WDT で最大のパラメータがあり、IWDTCLK 周波数が 15kHz であると仮定した場合、最後のリフレッシュからデバイスのリセットまたは NMI の生成までの時間は、下記のように 35 秒をわずかに下回ります。

IWDTLCK=15kHz

クロック分割比 = IWTCLK/256

タイムアウト周期 = 2048 サイクル

IWDT クロック周波数 =15kHz/256=58.59Hz

サイクル時間 = 1 / 58.59 Hz = 17.07 マイクロ秒

タイムアウト = 17.07 マイクロ秒 x 2048 サイクル = 34.95 秒

IWDT HAL モジュールを使用した DMAC/DTC のトリガ

ウォッチドッグカウンタがアンダーフローした場合、または有効なリフレッシュ期間外にリフレッシュが試みられた場合に、DMAC　または　DTC　周辺機能を使用してデータの転送をトリガするには、activation_source　を ELC_EVENT_IWDT_UNDERFLOW. に設定して DMAC/DTC 転送を構成します。詳細については、適切なモジュールのガイドを参照してください。

IWDT HAL モジュールを使用した ELC イベントのトリガ

IWDT は、イベントリンクコントローラ（ELC）で使用可能な他の周辺機能の開始をトリガできます。詳細については、ELC HAL モジュールのガイドを参照してください。

- JLink　デバッガを使用していると、IWDT　カウンタはカウントされないため、デバイスのリセットや NMI の生成は行われません。
- 実行できるアクティブなタスクがない場合、ThreadX は MCU をスリープ モードにします。IWDT が低消費電力モードでカウンタを停止するように構成されている場合、ThreadX RTOS で使用されているときは、アプリケーションでタイマを再度開始する必要があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.27.4 アプリケーションへの独立ウォッチドッグタイマ HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに独立ウォッチドッグタイマ HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

独立ウォッチドッグタイマドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（独立ウォッチドッグタイマドライバーのデフォルト名は g_iwdt0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

独立ウォッチドッグタイマ HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_wdt0 IWDT HAL on r_iwdt	Threads	New Stack> Driver> Monitoring> IWDT HAL on r_iwdt

次の図に示すように、r_iwdt の独立ウォッチドッグタイマドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

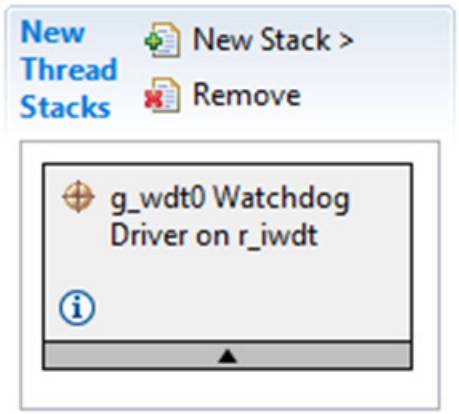


図 240:独立ウォッチドッグタイマ HAL モジュールのスタック

4.2.27.5 独立ウォッチドッグタイマ HAL モジュールの構成

ユーザーは必要な動作に合わせて独立ウォッチドッグタイマ HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_iwdt 上の独立ウォッチドッグタイマ HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_wdt0	Module name.

ISDE Property	Value	Description
NMI Callback	NULL	Callback. A user callback function can be registered in . If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the IRQn triggers. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

オプション機能選択レジスタ 0（OFS0）の構成

すべての Synergy マイクロコントローラのシリーズは、オプション設定メモリを備えています。このメモリを使用して、リセット後の周辺機能の動作状態を設定できます。OFS は、IWDT、WDT、LVD、CGC HOCO の状態の設定に使用できます。

IWDT HAL モジュールの割り込みの構成

IWDT の割り込みの構成は、ISDE を使用し、IWDT モジュールの他のオプションの構成と同じ方法で行います。アンダーフローまたは無効なリフレッシュで NMI 割り込みを生成するよう IWDT が構成されている場合、割り込みは BSP で有効になっている必要があります。

注：割り込みを有効にするには、[IWDT]>[IWDT NMIUNDF N] で優先順位を設定します。これにより、ssp_cfg/bsp/bsp_irq_cfg.h の BSP_IRQ_CFG_IWDT_UNDERFLOW が選択した優先順位レベルに設定されます。

IWDT NMIUNDF N 割り込みが BSP で有効になっている場合、対応する ISR が定義されます。ISR は、ユーザーコールバック関数が open API で登録されている場合、それをコールします。

独立ウォッチドッグタイマ HAL モジュールのクロック構成

IWDT の専用クロックは設定された周波数で動作しており、変更することはできません。

独立ウォッチドッグタイマ HAL モジュールのピン構成

IWDT は、動作のためにピンを必要としません。

4.2.27.6 アプリケーションでの独立ウォッチドッグタイマ HAL モジュールの使用

アプリケーションで独立ウォッチドッグタイマ HAL モジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して、IWDT NMI コールバックを登録します。
2) cfgGet API を使用して、IWDT HAL モジュールの構成を読み取ります。

3) refresh API を使用して、独立ウォッチドッグタイマをリフレッシュします。
4) statusGet API を使用して、IWDT のステータスフラグを読み取ります。
5) statusClear API を使用して、IWDT のステータスとエラーフラグをクリアします。
6) counterGet API を使用して、IWDT の現在のカウント値を読み取ります。
7) timeoutGet API を使用して、IWDT HAL モジュールのタイムアウト値を読み取ります。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

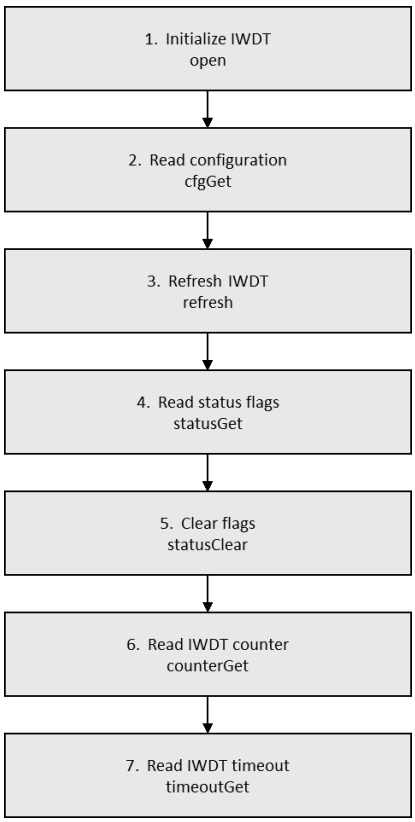


図 241:一般的な独立ウォッチドッグタイマ HAL モジュールアプリケーションのフロー図

4.2.28 JPEG デコード ドライバー

JPEG デコード HAL モジュールは、JPEG デコード画像処理用のハイレベル API を提供します。JPEG デコード HAL モジュールは、Synergy MCU の JPEG コーデック周辺機能を使用します。ユーザーコールバック関数を使用して、アプリケーションプログラムにキー処理イベントを通知できます。

4.2.28.1 JPEG デコード HAL モジュールの特長

- JPEG 伸長をサポート。
- JPEG　デコーダが完了するまでアプリケーションが待機できるようにするポーリングモードをサポート。
- ユーザーが指定したコールバック関数を使用した割り込みモードをサポート。
- 水平および垂直サブサンプル値、水平ストライド、デコードされたピクセル フォーマット、入力および出力データフォーマット、色空間などのパラメータを設定。
- 画像をデコードする前にそのサイズを取得。
- デコードされた画像フレームの保管のため、コード化されたデータを入力バッファおよび出力バッファに格納する操作をサポート。
- JPEG デコーダー モジュールへのコード化されたデータのストリーム転送をサポート。この機能により、アプリケーションは、ファイルやネットワークからのコード化された JPEG イメージの読み取りを、イメージ全体をバッファーリングすることなく実行できます。
- デコードするイメージ行数を設定。この機能により、アプリケーションは、デコードされたイメージの処理を、フレーム全体をバッファーリングすることなくオンザフライで実行可能。
- 入力値のデコードされたフォーマットとして YCbCr444、YCbCr422、YCbCr420、YCbCr411 をサポート。
- 出力フォーマットとして ARGB8888、RGB565 をサポート。
- JPEG イメージのサイズ、高さ、および幅が要件を満たさない場合にエラーを返すことが可能。

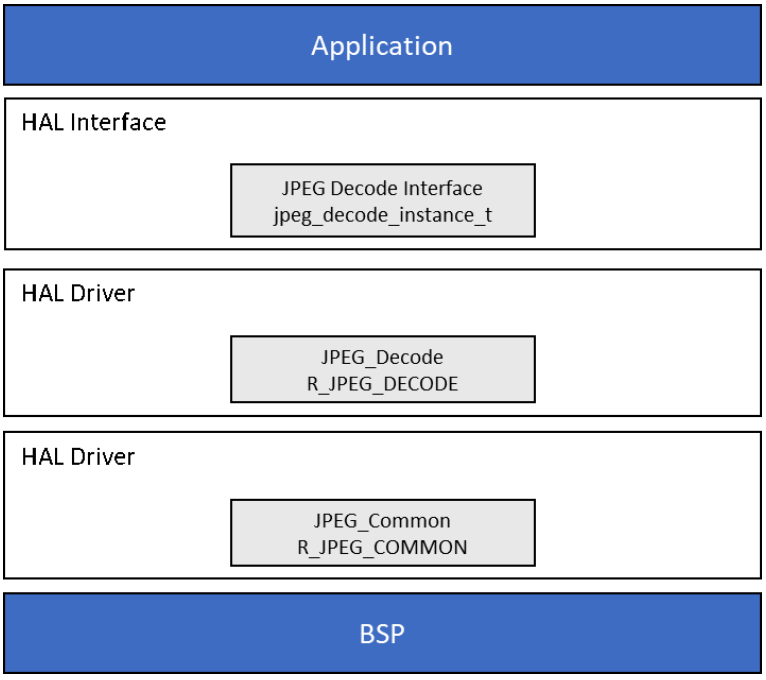


図 242:JPEG デコード HAL モジュールのブロック図

4.2.28.2 JPEG デコード HAL モジュールの API の概要

JPEG デコード HAL モジュールは、オープン、処理パラメータの設定、処理の開始、処理ステータスの取得、モジュールのクローズのための API を実装します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

JPEG デコード HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_jpeg_decode0.p_api->open(g_jpeg_decode0.p_ctrl, g_jpeg_decode0.p_cfg); Initial configuration.
outputBufferSet	g_jpeg_decode0.p_api->outputBufferSet(g_jpeg_decode0.p_ctrl, p_buffer, buffer_size); Assign output buffer to JPEG codec for storing output data.
horizontalStrideSet	g_jpeg_decode0.p_api->horizontalStrideSet(g_jpeg_decode0.p_ctrl, stride); Configure the horizontal stride value.
imageSubsampleSet	g_jpeg_decode0.p_api->imageSubsampleSet(g_jpeg_decode0.p_ctrl, horizontal, vertical); Configure the horizontal and vertical subsample settings.
inputBufferSet	g_jpeg_decode0.p_api->inputBufferSet(g_jpeg_decode0.p_ctrl, p_buffer, size); Assign input data buffer to JPEG codec.
linesDecodedGet	g_jpeg_decode0.p_api->linesDecodedGet g_jpeg_decode0.p_ctrl, p_lines); Return the number of lines decoded into the output buffer.

Function Name	Example API Call and Description
imageSizeGet	g_jpeg_decode0.p_api->imageSizeGet(g_jpeg_decode0.p_ctrl, p_horizontal, p_vertical); Retrieve image size during decoding operation.
statusGet	g_jpeg_decode0.p_api->statusGet(g_jpeg_decode0.p_ctrl, p_status); Retrieve current status of the JPEG codec module.
close	g_jpeg_decode0.p_api->close(g_jpeg_decode0.p_ctrl); Cancel an outstanding operation.
versionGet	g_jpeg_decode0.p_api->versionGet(&version); Get version and store it in provided pointer p_version.
pixelFormatGet	g_jpeg_decode0.p_api->pixelFormatGet(g_jpeg_decode0.p_ctrl, p_color_space); Get the input pixel format.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_INVALID_ALIGNMENT	Horizantal stride is not 8-byte aligned.

Name	Description
SSP_ERR_NOT_OPEN	Unit is not open
SSP_ERR_ASSERTION	An input pointer is NULL.
SSP_ERR_IN_USE	Peripheral is in use or hardware lock is taken.
SSP_ERR_HW_LOCKED	JPEG Codec resource is locked.
SSP_ERR_INVALID_CALL	An invalid call has been made, Codec output buffer address is attempted to change during codec operation. Or set output buffer first.
SSP_ERR_JPEG_IMAGE_SIZE_ERROR	Image size is not supported by JPEG codec.
SSP_ERR_JPEG_BUFFERSIZE_NOT_ENOUGH	Invalid buffer size
SSP_ERR_INVALID_MODE	JPEG Codec module is not decoding.
SSP_ERR_IMAGE_SIZE_UNKNOWN	The image size is unknown. More input data may be needed.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.28.3 JPEG デコード HAL モジュールの動作の概要

JPEG デコーダ HAL モジュールは、入力バッファストリーミングモードまたは JPEG 出力バッファストリーミングモードで使用できます。

入力バッファストリーミングモードの動作の説明

このシナリオでは、JPEG イメージデータはファイル上に存在し、ネットワークから受信されます。HAL レイヤドライバーは、最初に入力データをメモリに格納しなくても、このシナリオを処理できます。

出力バッファストリーミングモードの動作の説明

このシナリオでは、アプリケーションはデコードされたイメージデータをファイルまたはネットワークに書き込む必要があります。HAL レイヤドライバーは、アプリケーションがフレーム全体のメモリを割り当てることを必要としません。代わりに、アプリケーションは一度に 1 ライン以上のデコードを選択できます。この特長により、出力データに必要なメモリ量が大幅に削減されます。

MJPEG デコードの動作の説明

このシナリオでは、アプリケーションは JPEG 画像の連続的なストリームをネイティブディスプレイに表示する必要があります。JPEG 画像は、ファイルに存在するかネットワークから受信されます。HAL ドライバーは、JPEG デコードモジュールの入力バッファストリーミングモード機能を使用することによって、このシナリオを処理します。

これを行うための基本的なフローは次のとおりです。

1) JPEG デコードドライバーを開きます。
2) 画像パラメータ（水平ストライド、画像サブサンプル）を設定します。

3) JPEG 画像フレームを保持する入力バッファを設定します。
4) 未加工画像を保持する出力バッファを設定します。
5) デコードされた画像を表示します。
6) JPEG ドライバーを閉じます。
7) 必要に応じて手順 1 から処理を繰り返します。

JPEG デコード HAL モジュールの動作に関する重要な注意事項と制限事項

JPEG デコード コールバック

ユーザーコールバック関数を open API で登録できます。ユーザー コールバック関数が指定されている場合、割り込みが発生するたびに割り込みサービス ルーチン (ISR) からコールバック関数が呼び出されます。コールバック関数の引数 status は、デコーディングプロシージャにおいて発生したイベントをユーザーが識別できるように、以下に示す列挙値を受け取ることができます。

Event Name	Event Condition
JPEG_DECODE_STATUS_ERROR	JPEG Decode module encountered an error.
JPEG_DECODE_STATUS_IMAGE_SIZE_READY	JPEG Decode obtained the image size of data to be decoded, and paused.
JPEG_DECODE_STATUS_INPUT_PAUSE	JPEG Decode paused waiting for more input data.
JPEG_DECODE_STATUS_OUTPUT_PAUSE	JPEG Decode paused after decoded the number of lines specified by user.
JPEG_DECODE_STATUS_DONE	JPEG Decode operation has successfully completed.

注 : ユーザー コールバック関数は ISR から呼び出されるため、ブロッキング呼び出しを使用することや、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。

- JPEG デコード HAL モジュールは、JPEG デコード処理のみをサポートします。エンコーディングには JPEG エンコードドライバーを使用します。
- 両方のドライバーを使用している場合、JPEG デコードドライバーを使用するには JPEG エンコードドライバーを閉じる必要があります（逆も同様です。この 2 つのドライバーは同じ IP を共有します）。
- このモジュールの最新の制限事項については、SSP の最新のリリースノートを参照してください。

4.2.28.4 アプリケーションへの JPEG デコード HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに JPEG デコード HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

JPEG デコードドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(JPEG デコードドライバーのデフォルト名は g_jpeg_decode0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

JPEG デコード HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_jpeg_decode0 JPEG Decode Driver on r_jpeg_decode	Threads	New Stack> Driver> Graphics> JPEG Decode Driver

次の図に示すように、r_jpeg_decode の JPEG デコードドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

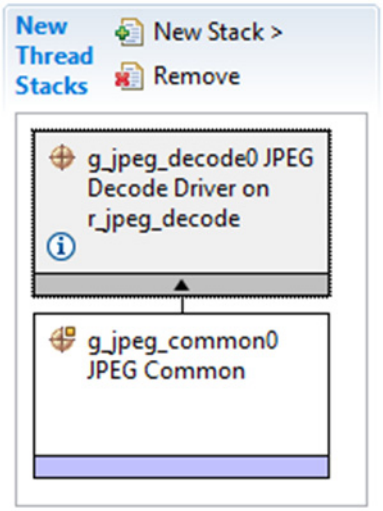


図 243:JPEG デコード HAL モジュールのスタック

4.2.28.5 JPEG デコード HAL モジュールの構成

ユーザーは必要な動作に合わせて JPEG デコード HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにあ

りますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_jpeg_decode 上の JPEG デコード HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_jpeg_decode0	The name to be used for a JPEG Decode module instance.
Byte Order for Input Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2) Default: Normal Byte order	Specify the byte order for input data. The order is swapped as specified in every 8-byte.

ISDE Property	Value	Description
Byte Order for Output Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2) Default: Normal Byte order	Specify the byte order for output data. The order is swapped as specified in every 8-byte.
Output Data Color Format	Pixel Data RGB565 format, Pixel Data ARGBB888 format Default: Pixel Data RGB565 format	Specify the output data format.
Alpha value to be applied to decoded pixel data (only valid for ARGB8888 format)	255	Specify the alpha value for the output data format (only valid for ARGB8888 format).
Name of user callback function	NULL	Specify the name of user callback function.
Decompression Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Decompression interrupt priority selection.
Data Transfer Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Data transfer interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

JPEG 共通モジュールの構成設定

ISDE Property	Value	Description
Name	g_jpeg_common0	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

JPEG デコード HAL モジュールのクロック構成

JPEG デコード HAL モジュールは PCLKA をクロックソースとして使用します。

実行時にクロック周波数を変更するには、CGC インタフェースを使用します。

JPEG デコード HAL モジュールのピン構成

JPEG デコード HAL モジュールには、構成可能な入力ピンも出力ピンもありません。

4.2.28.6 アプリケーションでの JPEG デコード HAL モジュールの使用

アプリケーションで JPEG デコード HAL モジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して、JPEG デコードを初期化します。
2) horizontalStrideSet API を使用して、水平ストライドを設定します。
3) imageSubsampleSet API を使用して、垂直および水平画像サブサンプルを設定します。
4) inputBufferSet API を使用して、(JPEG 画像が格納されている) 入力バッファのアドレスを設定します。
5) outputBufferSet API を使用して、出力バッファを設定します (未加工画像データを保持できる大きさが必要です)。
6) statusGet API を使用して、JPEG 操作を取得することができます。列挙値 (上記で説明) を返してユーザーに通知します。
   a) statusGet API から返されるステータス JPEG_DECODE_STATUS_DONE は、デコード操作が完了したことを示します。
   b) statusGet API から返されるステータス JPEG_DECODE_STATUS_INPUT_PAUSE または JPEG_DECODE_STATUS_OUTPUT_PAUSE は、デコード操作が完了していないことを示します。
7) 受信した未加工イメージデータをアプリケーションの必要に応じて操作します。
8) close API を使用して、モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

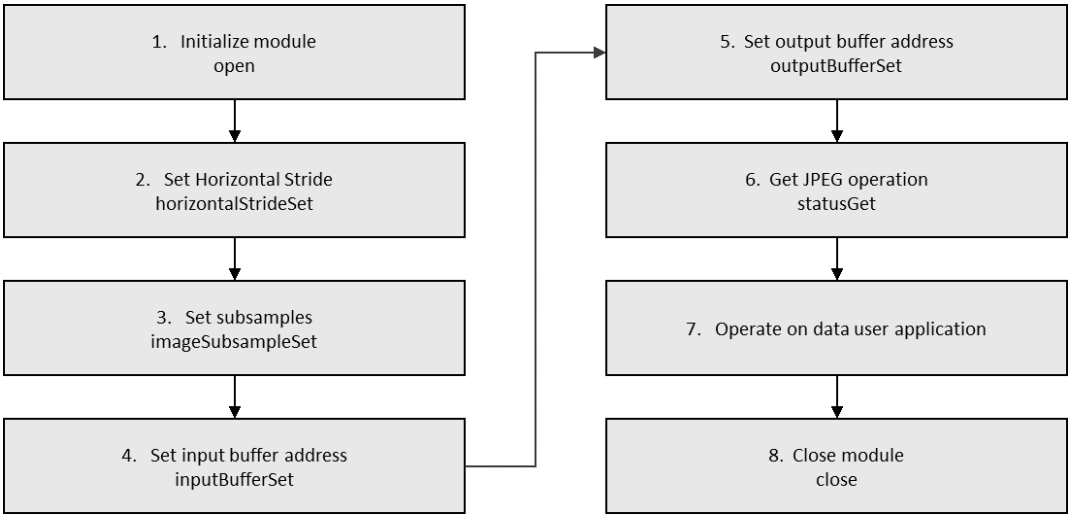


図 244:通常の JPEG デコード HAL モジュールアプリケーションのフロー図

4.2.29 JPEG エンコードドライバー

JPEG エンコード HAL モジュールは、業界標準の JPEG 画像エンコード処理（圧縮）用のハイレベル API を提供し、Synergy MCU の JPEG コーデックペリフェラルを使用します。ユーザーコールバック関数を使用して、アプリケーションプログラムにキー処理イベントを通知できます。

4.2.29.1 JPEG エンコード HAL モジュールの特長

- JPEG 圧縮をサポート。
- JPEG エンコーダが完了するまでアプリケーションが待機できるようにするポーリングモードをサポート。
- ユーザーが指定したコールバック関数を使用した割り込みモードをサポート。
- 水平および垂直解像度、水平ストライド、および品質係数などのパラメータを構成。
- エンコードおよび圧縮された JPEG 画像の保管のため、未加工の画像データを入力バッファおよび出力バッファに格納する操作をサポート。
- JPEG エンコーダモジュールへの未加工画像データのストリーム転送をサポート。この機能により、アプリケーションは、キャプチャデバイスやカメラまたはネットワークからのコード化された未加工画像の読み取りを、画像全体をバッファリングすることなく実行可能。
- 入力用に YCbCr422 色空間のみをサポート。
- RTP ストリーミングアプリケーション用に DRI Maker をサポート。
- 品質係数構成のサポート：品質係数値によって出力画像の品質を決定。

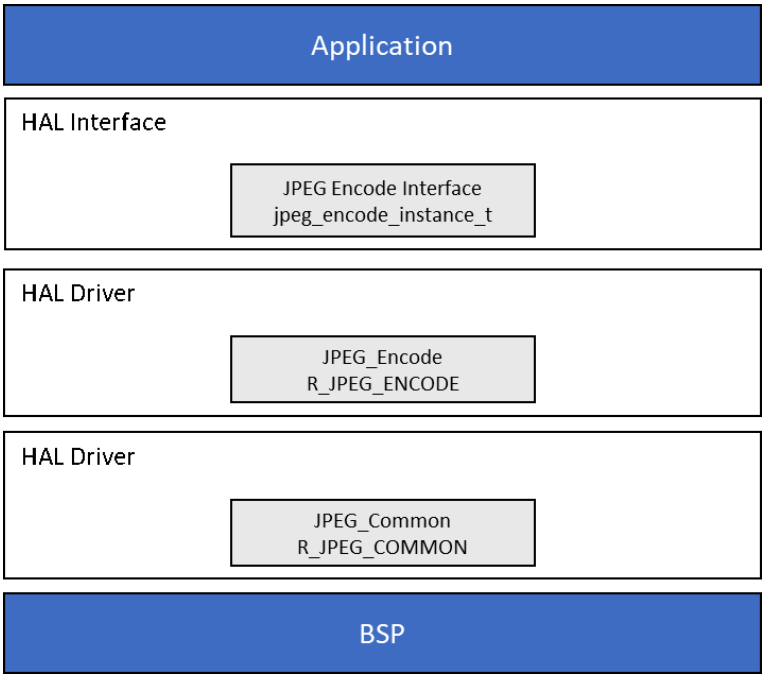


図 245:**JPEG エンコード HAL モジュールのブロック図**

4.2.29.2 JPEG エンコード HAL モジュールの API の概要

JPEG エンコード HAL モジュールでは、オープン、処理パラメータの設定、処理、モジュールからのステータスの取得、モジュールのクローズのための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は、HAL モジュールの API 要約の後にあります。

JPEG エンコード HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_jpeg_encode0.p_api->open(g_jpeg_encode0.p_ctrl, g_jpeg_encode0.p_cfg); Initial configuration.
imageParameterSet	g_jpeg_encode0.p_api->imageParameterSet(g_jpeg_encode0.p_ctrl, p_image_parameters); Set image parameters to JPEG Codec.

Function Name	Example API Call and Description
outputBufferSet	g_jpeg_encode0.p_api->outputBufferSet(g_jpeg_encode0.p_ctrl, p_buffer); Assign output buffer to JPEG Codec for storing output data.
inputBufferSet	g_jpeg_encode0.p_api->inputBufferSet(g_jpeg_encode0.p_ctrl, p_buffer, buffer_size); Assign input data buffer to JPEG Codec.
statusGet	g_jpeg_encode0.p_api->statusGet(g_jpeg_encode0.p_ctrl, p_status); Retrieve current status of the JPEG Codec module.
close	g_jpeg_encode0.p_api->close(g_jpeg_encode0.p_ctrl); Cancel an outstanding operation.
versionGet	g_jpeg_encode0.p_api->versionGet(&version); Get version and store it in the provided pointer version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_INVALID_ALIGNMENT	Horizontal stride is not 8-byte aligned.

Name	Description
SSP_ERR_NOT_OPEN	Unit is not open
SSP_ERR_ASSERTION	An input pointer is NULL.
SSP_ERR_IN_USE	Peripheral is in use or hardware lock is taken.
SSP_ERR_HW_LOCKED	JPEG Codec resource is locked.
SSP_ERR_INVALID_CALL	An invalid call has been made, Codec output buffer address is attempted to change during codec operation. Or set output buffer first.
SSP_ERR_JPEG_IMAGE_SIZE_ERROR	Image size is not supported by JPEG codec.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.29.3 JPEG エンコード HAL モジュールの動作の概要

JPEG エンコーダ HAL モジュールは、入力バッファストリーミングモードまたはノーマルモードで使用できます。

入力バッファストリーミングモードの動作の説明

このモードでは、未加工画像データがネットワーク、ファイル、またはキャプチャデバイスから、独立したデータ「チャンク」として届きます。HAL レイヤードライバーがこのモードを処理するとき、あらかじめ入力データをメモリに格納する必要はありません。

ノーマルモードの動作の説明

このモードでは、未加工画像データがネットワーク、ファイル、またはキャプチャデバイスから、完全なフレームとして届きます。HAL レイヤードライバーはこのモードを処理して、未加工画像のフレーム全体を圧縮できます。

JPEG エンコード HAL モジュールの動作に関する重要な注意事項と制限事項

モーション JPEG

モーション JPEG（MJPEG）に定義済みの標準はありませんが、基本的な考え方は、JPEG 画像を連続的にレンダリングデバイスに送り出すことです。アプリケーションで JPEG エンコーダ HAL モジュールを使用して MJPEG 機能を実装するための基本的な手順は次のとおりです。

1) 目的の品質係数値（デフォルトは 50）で JPEG エンコーダドライバーを開きます。
2) 画像の解像度を設定します（[Thread Stack] ウィンドウで構成済みの場合はオプション）。
3) YCbCr422 画像をキャプチャするためのキャプチャデバイスを初期化します。
4) outputBufferSet API 関数を使用して、JPEG 画像を保持する出力バッファを設定します。
5) 画像をキャプチャします。
6) inputBufferSet API 関数を使用して、キャプチャデバイスからキャプチャされた未加工の YCbCr 422 画像を保持する入力バッファを設定します。
7) エンコード操作のステータスを確認し、DONE であれば画像をレンダリングデバイスに送ります。

8) 必要に応じて手順 5 から処理を続行します。

JPEG エンコードコールバック

ユーザーコールバック関数を open API で登録できます。ユーザー コールバック関数が指定されている場合、割り込みが発生するたびに割り込みサービス ルーチン (ISR) からコールバック関数が呼び出されます。コールバック関数の引数 status は、エンコーディングプロシージャにおいて発生したイベントをユーザーが識別できるように、以下に示す列挙値を受け取ることができます。

Event Name	Event Condition
	JPEG Encode paused waiting for more input data.
	JPEG Encode operation has successfully completed.

注 : ユーザー コールバック関数は ISR から呼び出されるため、ブロッキング呼び出しを使用することや、長時間処理することはしないように注意してください。ISR の中で長時間費やすと、システムの応答性に影響を与えかねません。

- JPEG エンコード HAL モジュールは、JPEG エンコード処理のみをサポートします。デコードには JPEG デコード HAL モジュールを使用してください。
- エンコードモジュールとデコードモジュールの両方を使用するアプリケーションでは、JPEG エンコードドライバーを使用するためには JPEG デコードモジュールを閉じる必要があります (逆も同様です)。これは、両方のモジュールが同じ MCU ペリフェラルを共有しているためです。
- JPEG エンコード HAL モジュールでは、[Thread Stack] ウィンドウで [Normal byte order] のみがサポートされます。それ以外のオプションでは、無効な画像が得られる可能性があります。

4.2.29.4 アプリケーションへの JPEG エンコード HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに JPEG エンコード HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

JPEG エンコードドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(JPEG エンコードドライバーのデフォルト名は g_jpeg_encode0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

JPEG エンコード HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_jpeg_encode0 JPEG Encode Driver on r_jpeg_encode	Threads	New Stack> Driver> Graphics> JPEG Encode Driver

次の図に示すように、r_jpeg_encode の JPEG エンコードドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

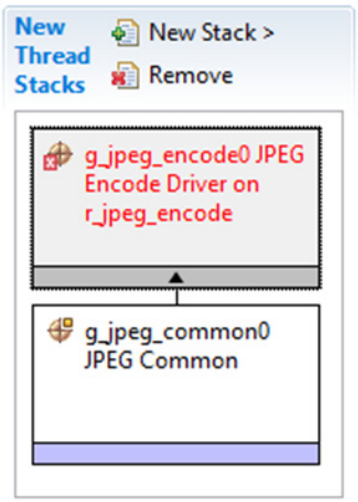


図 246:JPEG エンコード HAL モジュールのスタック

4.2.29.5 JPEG エンコード HAL モジュールの構成

ユーザーは必要な動作に合わせて JPEG エンコード HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_jpeg_encode 上の JPEG エンコード HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_jpeg_encode0	Module name.
RAW Image Vertical Resolution	800	RAW image vertical resolution selection
RAW Image Horizontal Resolution	480	RAW image horizontal resolution selection
Byte Order for Input Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2) Default: Normal Byte order	Byte order for input data format selection

ISDE Property	Value	Description
Byte Order for Output Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2) Default: Normal Byte order	Byte order for output data format selection
Define Restart Marker	512	Define restart marker selection
Quality Factor	50	Quality factor selection
Name of user callback function	NULL	Name of user callback function selection
Decompression Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Decompression interrupt priority selection.
Data Transfer Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Data transfer interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

JPEG 共通モジュールの構成設定

ISDE Property	Value	Description
Name	g_jpeg_common0	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

JPEG エンコード HAL モジュールのクロック構成

JPEG エンコード HAL モジュールは PCLKA をクロックソースとして使用します。

実行時にクロック周波数を変更するには、CGC インタフェースを使用します。

JPEG エンコード HAL モジュールのピン構成

JPEG エンコード HAL モジュールには、デバイス上に構成可能な入力ピンも出力ピンもありません。

4.2.29.6 アプリケーションでの JPEG エンコード HAL モジュールの使用

アプリケーションで JPEG エンコード HAL モジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して、JPEG エンコードを初期化します。
   a) 品質係数、水平ストライド、水平および垂直解像度を構成します。
2) outputBufferSet API を使用して、出力バッファアドレスを設定します（圧縮された JPEG 画像を保持できる大きさが必要です）。
3) inputBufferSet API を使用して、エンコーディング操作を開始する入力バッファ（未加工画像データのアドレスおよびサイズ）を設定します。
4) statusGet API を使用して JPEG 操作を取得でき、statusGet API は列挙値（上記で説明）を返してユーザーに通知します。
   a) statusGet API からのステータス JPEG_ENCODE_STATUS_DONE は、エンコード操作が完了したことを示します。
   b) statusGet API からのステータス JPEG_ENCODE_STATUS_INPUT_PAUSE は、ドライバーが追加の入力を待っていることを示します。手順 3 に移動し、残りのデータを使用して入力バッファを設定します。
5) 受信した JPEG 画像データをアプリケーションの必要に応じて操作します。
6) close API を使用して、モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

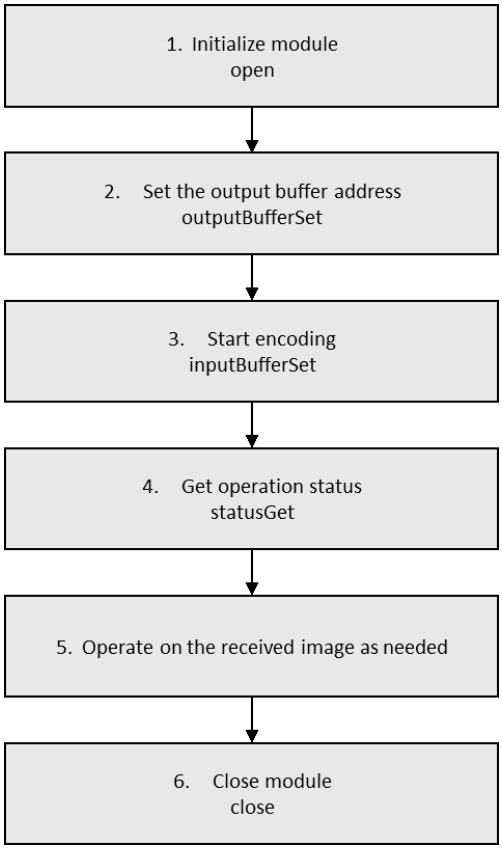


図 247:一般的な JPEG エンコード HAL モジュールアプリケーションのフロー図

4.2.30 Key Matrix ドライバー

Key Matrix HAL モジュールは、キー入力アプリケーション用のハイレベル API を提供し、Synergy MCU 上のキー割り込み機能ペリフェラルを使用します。ユーザー定義のコールバックを作成し、CPU にキー押下イベントを通知できます。

4.2.30.1 Key Matrix HAL モジュールの特長

この Key Matrix HAL モジュールは、キー割り込み（KINT）周辺機能を構成および制御します。次のキー機能を実装します :

- KINT チャネルでの立ち上がりエッジと立ち下がりエッジの両方をサポートします
- 割り込みベースのイベント通知をサポートします
- 複数のイベントを効率よくキャプチャするためのビットマスク機能をサポートします
- 任意の 2 つのチャネルにエッジのあるマトリックスキーパッドをサポートします

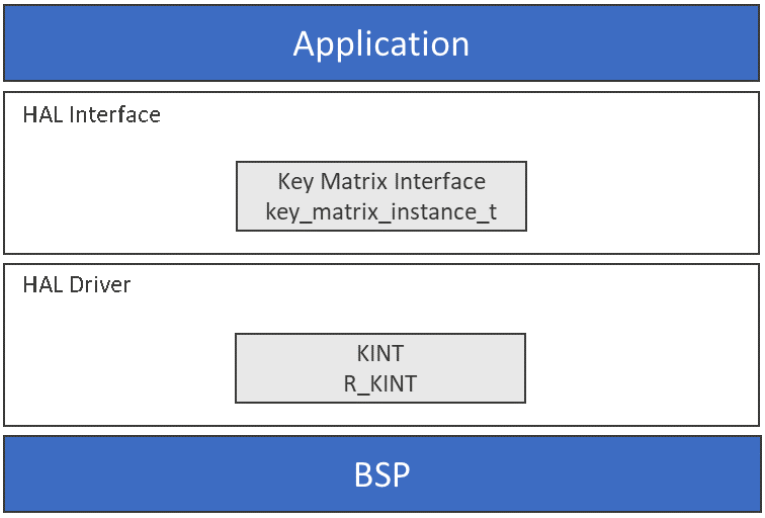


図 248:**Key Matrix HAL モジュールのブロック図**

4.2.30.2 Key Matrix HAL モジュールの API の概要

Key Matrix HAL モジュールでは、キー割り込み機能のオープン、クローズ、有効化、無効化のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

Key Matrix HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_kint.p_api->open(g_kint.p_ctrl, g_kint.p_cfg_cfg) Initial configuration.
enable	g_kint.p_api->enable(g_kint.p_ctrl) Enable Key interrupt.
disable	g_kint.p_api->disable(g_kint.p_ctrl) Disable Key interrupt.

Function Name	Example API Call and Description
triggerSet	g_kint.p_api->triggerSet() Set trigger for Key interrupt.
close	g_kint.p_api->close(&g_ keymatrix) Allow driver to be reconfigured. May reduce power consumption.
versionGet	g_kint.p_api->versionGet(&version) Get version and store it in provided pointer version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successfully completed.
SSP_ERR_ASSERTION	Parameter has invalid value.
SSP_ERR_INVALID_ARGUMENT	Argument is invalid.
SSP_ERR_HW_LOCKED	The API has already been opened. It must be closed before it can be opened again.
SSP_ERR_NOT_OPEN	The peripheral is not opened.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.30.3 Key Matrix HAL モジュールの動作の概要

Key Matrix HAL モジュールは、いずれかのキー割り込み（KINT）チャネルでの立ち上がりエッジまたは立ち下がりエッジを検出するように、KINT 周辺機能を構成します。設定されているいずれかのピンで該当するイベントが検出された場合、このモジュールは割り込みを生成します。その後、割り込みはユーザーコールバック (p_callback) を呼び出します。そのときに渡されるコールバック引数 keymatrix_callback_args_t では、ビットマスクを使用してエッジが検出された channel(s) が指定されています。

いずれか 1 つのチャネルでエッジが検出されると割り込みが生成されますが、その時点でトリガされたすべてのピンのビットマスク keymatrix_channels_t がコールバックで返されます（他のいずれかのピンでもエッジが検出された場合）。そのため、コールバックが呼び出される前に他のピンでもエッジが検出された場合、ピンごとのエッジ検出に対して必ずしも割り込みが生成されるとは限りません。コールバックが呼び出された後で新しいエッジが検出されると、割り込みが再度トリガされ、新たにコールバックが呼び出されます。割り込み要因チャネルを識別するために、ユーザーコールバックのビットマスクを確認する必要があります。

このモジュールを使用すると、押された実際のキーを示す 2 つのチャネル上のエッジを使用した、マトリックスキーパッドを実装できます。また、このモジュールは 1 つの入力ピン上のエッジを検出する単一入力としても使用できます。

Key Matrix HAL モジュールの動作に関する重要な注意事項と制限事項

- トリガエッジが検出されたときに、DMAC または DTC 周辺機能を使用するデータの転送をトリガするには、activation_source を ELC_EVENT_KEY_INT. に設定して DMAC/DTC 転送を構成します。
- KINT モジュールは、ELC に対して使用可能な他の周辺機能の開始をトリガできます。詳細については、『SSP ユーザーズマニュアル』の ELC に関するユーザーガイドを参照してください。
- このモジュールが動作するためには、open のコールでコールバックが使用されているかどうかにかかわらず、BSP で KINT（INTKR）割り込みを有効にする必要があります。
- このモジュールは、ポーリングモードの動作をサポートしていません。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.30.4 アプリケーションへの Key Matrix HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して Key Matrix HAL モジュールをアプリケーションに組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

Key Matrix ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（Key Matrix ドライバーのデフォルトの名前は g_kint0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

Key Matrix HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_kint0 Key Matrix Driver on r_kint	Threads	New Stack> Driver> Input> Key Matrix Driver on r_kint

次の図に示すように、r_kint の Key Matrix ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」という

テキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

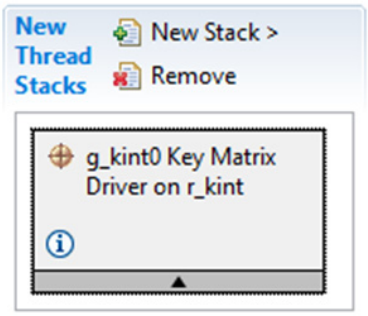


図 249:**Key Matrix HAL モジュールのスタック**

4.2.30.5 Key Matrix HAL モジュールの構成

ユーザーは必要な動作に合わせて、Key Matrix HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

変更が必須と識別されることが最も多いプロパティの 1 つは、割り込み優先順位です。この構成設定は、対応するモジュールの [Properties] ウィンドウで使用できます。示されたモジュールを選択するだけで、[Properties] ウィンドウが表示されます。割り込み設定は通常はプロパティリストの下の方にあるため、表示されるまでスクロールダウンします。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_kint での Key Matrix HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_kint0	Module name.

ISDE Property	Value	Description
Channel 0-7	Unused, Used Default: Unused	This is a bit-mask with each bit specifying if that channel is to be enabled or not. Select the channels to be used.
Trigger Type	Rising Edge, Falling Edge Default: Rising Edge	Specify if the enabled channels detect a rising edge or a falling edge. NOTE: either all channels detecting a rising edge or all channels detecting a falling edge.
Interrupt enabled after initialization	True, False Default: False	Specify if the module interrupts must be enabled as part of the open call.
Callback	NULL	A user callback function can be registered in . If this callback function is provided, it will be called from the interrupt service routine (ISR) each time a configured edge is detected on any of the channels. Note: Without callback, the application cannot determine whether an event has occurred. Warning: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.

ISDE Property	Value	Description
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

Key Matrix HAL モジュールのクロック構成

Key Matrix HAL モジュールには、固有のクロック構成は必要ありません。

Key Matrix HAL モジュールのピン構成

KINT 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は KINT ピンの選択例を示しています。

r_kint での Key Matrix HAL モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
KINT	Pins	Select Peripherals > Input:KINT> KINT0

注 : この選択シーケンスでは、KINT0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_kint での Key Matrix HAL モジュールのピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom Default: Disabled	Select Custom as the Operation Mode
KRM0:7	None, Pnn Default: None	Key Interrupt Pin selection

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

4.2.30.6 アプリケーションでの Key Matrix HAL モジュールの使用

アプリケーションで Key Matrix HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、Key Matrix HAL モジュールを初期化します。
2) オートスタートの構成設定が true の場合、モジュールはすぐに動作を開始します。
   a) オートスタートが設定されていない場合は、enable API を使用して動作を有効にします。
3) キー入力に応答します
4) disable API を使用して、動作を無効化します。
5) 初期化後にトリガエッジを変更するには、triggerSet API を使用します。
6) close API を使用して、モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

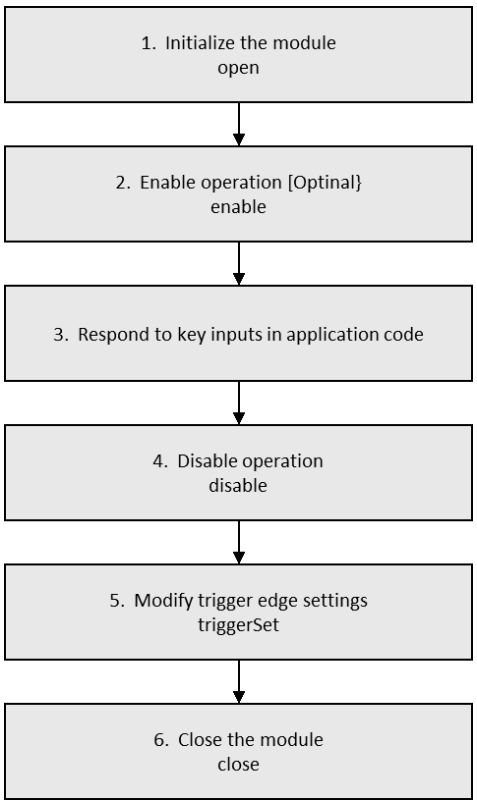


図 250:通常の Key Matrix HAL モジュールアプリケーションのフロー図

4.2.31 ローパワー モード ドライバー

低消費電力モード HAL モジュールは、低消費電力モードアプリケーション用のハイレベル API を提供し、Synergy MCU 上の低消費電力モードハードウェアペリフェラルを使用します。

4.2.31.1 LPM HAL モジュールの特長

LPM HAL モジュールは、低消費電力モードハードウェアペリフェラルを使用して、MCU 動作電力制御モードと MCU 低消費電力モードの構成をサポートしています。LPM ドライバーは、さまざまな電力モードで、内部の電力供給制御と IO ポートの状態の再設定を通じて消費電力の低減をサポートします。さらに、LPM ドライバーは、MCU 上のその他のハードウェアペリフェラルの無効化と有効化をサポートしています。

注 : すべての MCU で、すべての動作モードが利用可能とは限りません。すべての MCU で、すべての低消費電力モードが利用可能とは限りません。これは非推奨のモジュールであるため、新しいプロジェクトでは使用しないでください。代わりに lpm_v2 モジュールを使用してください。

LPM ドライバーは以下の動作電力制御モードをサポートしています。

- 低電圧モード
- 低速モード
- 中速モード
- 高速モード
- サブ発振器速度モード

この LPM ドライバーは、ローパワー モードの以下の機能をサポートしています。

- ディープ ソフトウェア スタンバイ モード
- ソフトウェア スタンバイ モード
- スリープ モード
- スヌーズ モード

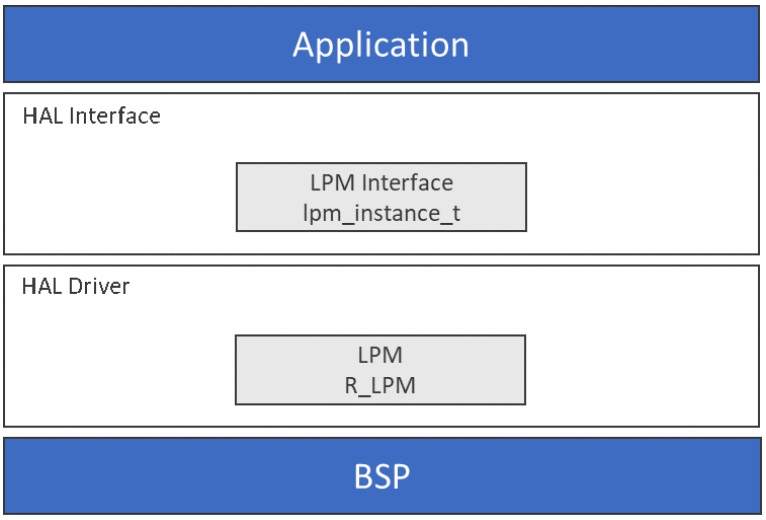


図 251:**LPM HAL モジュールのブロック図**

4.2.31.2 LPM HAL モジュールの API の概要

LPM HAL モジュールでは、初期化、値の設定と取得、モジュールの停止のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

LPM HAL モジュールの API の要約

Function Name	Example API Call and Description
init	g_lpm0.p_api->init(g_lpm0.p_cfg); Open the LPM driver module Initialized the LPM block according to the passed in config structure.
mstpcrSet	g_lpm0.p_api->mstpcrSet(value1, value2, value3, value4); Set the value of all the Module Stop Control Registers.
mstpcrGet	g_lpm0.p_api->mstpcrSet(&value1, &value2, &value3, &value4); Get the values of all the Module Stop Control Registers.
moduleStop	g_lpm0.p_api->moduleStop(module); Stop a module.
moduleStart	g_lpm0.p_api->moduleStart(module); Run the specified module.
operatingPowerModeSet	g_lpm0.p_api->operatingPowerModeSet(power, osc); Set the operating power mode and oscillator.

Function Name	Example API Call and Description
snoozeEnable	g_lpm0.p_api->snoozeEnable(rdx0_mode, dtc_mode, requests, triggers); Configure and enable snooze mode.
snoozeDisable	g_lpm0.p_api->snoozeDisable(); Disable snooze mode.
lowPowerCfg	g_lpm0.p_api->lowPowerCfg(power_mode, output_port_enable, power_supply, io_port_state); Configure a low power mode.
wupenSet	g_lpm0.p_api->wupenSet(wupen_value); Set the value of the Wake Up Interrupt Enable Register WUPEN.
wupenGet	g_lpm0.p_api->wupenGet(&wupen_value); Get the value of the Wake Up Interrupt Enable Register WUPEN.
deepStandbyCancelRequestEnable	g_lpm0.p_api-> deepStandbyCancelRequestEnable (pin_signal, rising_falling); Enable a Deep Standby Cancel Request.
deepStandbyCancelRequestDisable	g_lpm0.p_api-> deepStandbyCancelRequestDisable (pin_signal); Disable a Deep Standby Cancel Request.

Function Name	Example API Call and Description
lowPowerModeEnter	g_lpm0.p_api->lowPowerModeEnter (); Enter low power mode (sleep/standby/deep standby) using WFI macro. Function will return after waking from low power mode.
versionGet	g_lpm0.p_api->versionGet(&version); Retrieve the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_ASSERTION	Parameter has invalid value.
SSP_ERR_INVALID_PTR	p_version is NULL.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.31.3 LPM HAL モジュールの動作の概要

LPM 初期化

他の LPM 関数を呼び出す前に、LPM API 関数 init を呼び出す必要があります。init 関数は、内部変数とロックの初期化を処理します。

スリープ低消費電力モード

デフォルトでは、電源投入時に低消費電力モードとしてスリープモードが有効になっています。スリープモードは、通常のプログラム実行モードに戻すために（MCU をスリープから復帰させる適切な割り込みまたはイベントを構成して有効にすることを除き）特別な構成を必要としないため、使用できる最も便利な低消費電力モードです。スリープモードでは、SRAM、プロセッサレジスタ、およびハードウェア周辺機能の状態がすべて維持され、スリープへの移行とスリープからの復帰に必要な時間は最小限で済みます。ThreadX® スレッドスケジューラが使用する Systick 割り込みを含め、割り込みが発生すると MCU デバイスはスリープモードから復帰します。他の関数の前に、LPM API 関数 init を呼び出す必要があります。LPM API 関数 lowPowerCfg を使用すると、低消費電力モードとしてスリープを使用するように MCU を構成できます。スリープモードに直接移行するには、LPM API 関数 lowPowerModeEnter を使用する必要があります。

```
/* HAL-only entry function */

#include "hal_data.h"

void hal_entry(void)

{

/* Not shown: Enable any interrupt to wake from sleep mode */

/* Configure LPM peripheral for sleep mode */

g_lpm_on_hal.p_api->lowPowerCfg(LPM_LOW_POWER_MODE_SLEEP,

LPM_OUTPUT_PORT_ENABLE_RETAIN,

LPM_POWER_SUPPLY_DEEPCUT0,

LPM_IO_PORT_NO_CHANGE);

/* Enter sleep mode */

g_lpm_on_hal.p_api->lowPowerModeEnter();

}
```

「ソフトウェア スタンバイ モード」

ソフトウェア スタンバイ モードでは、CPU、オンチップ周辺機能の大部分、およびすべての内部発振器が停止します。ただし、CPU 内蔵レジスタと SRAM データの内容、およびオンチップ周辺機能と I/O ポートの状態は保持されます。ソフトウェア スタンバイ モードでは大半の発振器が停止するため、このモードを使用すると、消費電力を大幅に削減できます。スリープ モードと同様、スタンバイ モードの場合にも、スタンバイ モードからの復帰のために割り込みまたはイベントを設定して有効化する必要があります。MCU をスリープモードから復帰させることができるイベントと割り込みの完全な一覧については、『MCU Synergy ハードウェアユーザーズマニュアル』の表「Interrupt Sources To Transition To Normal Mode from Snooze and Software Standby Mode」を参照してください。ソフトウェア スタンバイ モードに入る前に、Wake Up 割り込み有効化レジスタ (WUPEN) を変更しておく必要があります。詳細は、「Wake Up 割り込み有効化レジスタ（WUPEN）」および『MCU Synergy ユーザーズマニュアル : マイクロコントローラ』のソフトウェアスタンバイモードのセクションをご覧ください。

注 :ThreadX アイドルスレッドと ThreadX 関数 tx_thread_sleep() を使用するには、MCU の低消費電力モードとしてスリープを構成する必要があります。詳細については、以下の使用上の注意を参照してください。

```
/* HAL-only entry function */
#include "hal_data.h"

#define WUPEN_AGT1_UNDERFLOW_MASK (1 << 28)
void hal_entry(void)
{
uint32_t wupen_value = 0;
/* Not shown: Configuration of the interrupt or event to wake from standby */

/* Set bit in WUPEN register to allow mcu to wake from standby through a specifi
```

```
c interrupt or event. AGT1 underflow is used in this example. */

/* Get current value of WUPEN register */
g_lpm.p_api->wupenGet(&wupen_value);

/* Set wake by AGT1 underflow bit in WUPEN register */
g_lpm.p_api->wupenSet(wupen_value | WUPEN_AGT1_UNDERFLOW_MASK);

/* Configure LPM peripheral for standby mode */
g_lpm.p_api->lowPowerCfg(LPM_LOW_POWER_MODE_STANDBY,
LPM_OUTPUT_PORT_ENABLE_RETAIN,
LPM_POWER_SUPPLY_DEEPCUT0,
LPM_IO_PORT_NO_CHANGE);

/* Enter standby mode */
g_lpm.p_api->lowPowerModeEnter();
   }
```

ソフトウェア スタンバイ モードを使用したスヌーズ モード

スヌーズ モードでは、MCU 周辺機器を使用して、MCU をローパワー ステータスで維持しながら、基本タスクを実行することができます。ADC や DTC、その他の周辺機器はスヌーズ モードで有効にできます。次のコードはスヌーズ モードの有効化処理とスヌーズ中の DTC の実行処理を示します。タイマ AGT1 の設定処理は記されていません。MCU をソフトウェア スタンバイ モードにすると、AGT1 の比較一致 A が検出されたときにスヌーズに移行します。DTC の転送が完了すると、MCU はスヌーズ モードではなくなります。DTC の詳細については、DTC の使用上の注意を参照してください。ISDE によって生成されるソースファイルに g_transfer_on_dtc ドライバーを組み込んでプロジェクトを生成すると、ISDE は以下のインスタンス構造を構成します。

注 :ThreadX アイドルスレッドと ThreadX 関数 tx_thread_sleep() を使用するには、MCU の低消費電力モードとしてスリープを構成する必要があります。詳細については、次の使用上の注意を参照してください。

```
#define WUPEN_AGT1_UNDERFLOW_MASK (1 << 28)
/* DTC settings */

g_transfer.p_cfg->p_info->p_src = &my_tx_data[0];
g_transfer.p_cfg->p_info->p_dest = &my_rx_data[0];
g_transfer.p_cfg->p_info->length = TRANSFER_SIZE;
g_transfer.p_api->open(g_transfer.p_ctrl, g_transfer.p_cfg);

/* snooze settings */
lpm_snooze_rxd0_t   rdx0_mode = LPM_SNOOZE_RXD0_FALLING_EDGE_IGNORE;
lpm_snooze_dtc_t        dtc_mode = LPM_SNOOZE_DTC_ENABLE;
lpm_snooze_request_t    requests = LPM_SNOOZE_REQUEST_AGT1_COMPARE_A;
lpm_snooze_end_t        triggers = LPM_SNOOZE_END_DTC_TRANS_COMPLETE;
g_lpm.p_api->snoozeEnable(rdx0_mode,
                          dtc_mode,
                          requests,
                          triggers);

/* standby settings */
lpm_low_power_mode_t        power_mode = LPM_LOW_POWER_MODE_STANDBY;
```

```
lpm_output_port_enable_t    output_port_enable = LPM_OUTPUT_PORT_ENABLE_RETAIN;
lpm_power_supply_t      power_supply = LPM_POWER_SUPPLY_DEEPCUT0;
lpm_io_port_t           io_port_state = LPM_IO_PORT_RESET;

g_lpm.p_api->lowPowerCfg(power_mode,
                         output_port_enable,
                         power_supply,
                         io_port_state);

/* Clear WUPEN */
g_lpm.p_api->wupenSet(0);

/* Wake from AGT1 interrupt underflow */
g_lpm.p_api->wupenSet(WUPEN_AGT1_UNDERFLOW_MASK);
```

ディープ ソフトウェア スタンバイ モード

ディープ ソフトウェア スタンバイ モードは S7G2 MCU でのみ利用可能であり、ソフトウェア スタンバイよりもローパワー モードです。リセットピンのネゲートまたは API typedef lpm_deep_standby_t で説明されるスリープ解除イベントのセットのいずれかによるリセットにより、MCU は必ずディープソフトウェアスタンバイモードからスリープ解除されます。以下のコードは、AGT1 アンダーフローを使用して MCU をディープ スタンバイ モードからスリープ解除し、ディープ ソフトウェア スタンバイ モードを有効にしてそのモードに移行する方法を示します。AGT1 が期限切れになり MCU のリセットを引き起こすと、MCU はリセットされます。

注 :ThreadX アイドルスレッドと ThreadX 関数 tx_thread_sleep() を使用するには、MCU の低消費電力モードとしてスリープを構成する必要があります。詳細については、以下の使用上の注意を参照してください。

```
#define WUPEN_AGT1_UNDERFLOW_MASK (1 << 28)

/* Deep standby wake signal (wake will cause reset) */
g_lpm.p_api->deepStandbyCancelRequestEnable(LPM_DEEP_STANDBY_AGT1,
                                  LPM_CANCEL_REQUEST_EDGE_RISING);

/* Deep standby settings */
lpm_low_power_mode_t      power_mode = LPM_LOW_POWER_MODE_DEEP;
lpm_output_port_enable_t  output_port_enable = LPM_OUTPUT_PORT_ENABLE_RETAIN;
lpm_power_supply_t        power_supply = LPM_POWER_SUPPLY_DEEPCUT0;
lpm_io_port_t             io_port_state = LPM_IO_PORT_RESET;

g_lpm.p_api->lowPowerCfg(power_mode,
                         output_port_enable,
                         power_supply,
                         io_port_state);

/* Clear WUPEN */
g_lpm.p_api->wupenSet(0);

/* Wake from AGT1 interrupt underflow */
g_lpm.p_api->wupenSet(WUPEN_AGT1_UNDERFLOW_MASK);
                      g_lpm.p_api->lowPowerModeEnter();
```

LPM HAL モジュールの動作に関する重要な注意事項と制限事項

このドライバーを使用して動作モードを変更するには、CGC モジュールを使用する必要があります。CGC ドライバーに関する使用上の注意も確認してください。MCU の動作モードを適切に変更するために必要なイベントのシーケンスの詳細については、『MCU Synergy ハードウェアユーザーズマニュアル』の「動作電力低減機能」セクションを参照してください。

このドライバーを使用し、割り込みに基づいて LPM 周辺機能を構成するには、BSP を使用して割り込みを構成および有効化する必要があります。

LPM API 関数 init の呼び出しによって設定されるのは、MCU の動作モードのみです。init 関数は、低電力モードの設定は行いません。

低消費電力モードを構成するには、LPM API 関数 lowPowerCfg を使用します。構成が完了すると、LPM API 関数 lowPowerModeEnter を使用して、いつでも低消費電力モードに移行することができます。

ディープスタンバイまたはスヌーズを使用するには、LPM API 関数 deepStandbyCancelRequestEnable および snoozeEnable を使用して、LPM 周辺機能を追加構成する必要があります。

ソフトウェア スタンバイ モードに入る前に、Wake Up 割り込み有効化レジスタ (WUPEN) を変更しておく必要があります。詳細は、「Wake Up 割り込み有効化レジスタ（WUPEN）」および『MCU Synergy ユーザーズマニュアル : マイクロコントローラ』のソフトウェアスタンバイモードのセクションを参照してください。

サブ発振器電力制御モードに移行するには、API 関数 operatingPowerModeSet を呼び出す前に、CGC モジュールを使用して MOCO クロックと HOCO クロックを停止する必要があります。

メイン発振器またはメイン発振器ソースを備えた PLL をシステム クロックに使用している場合、スタンバイ モードからの復帰時間は MOSCWTCR レジスタでのメイン発振器待ち時間により影響を受ける可能性があります。このレジスタ設定は、CGC ドライバー プロパティの [メイン発振器待ち時間設定] から変更可能です。詳細については、「電気的特性」の表「ウェイクアップのタイミングおよび時間」を参照してください。

注 : プロジェクトで ThreadX を使用する場合、アプリケーションは API 関数 lowPowerModeEnter をコールする直前にのみ LPM_LOW_POWER_MODE_STANDBY または LPM_LOW_POWER_MODE_DEEP に切り替える必要があります。LPM_LOW_POWER_MODE_STANDBY または LPM_LOW_POWER_MODE_DEEP が ThreadX で使用されている場合は、MCU が LPM_LOW_POWER_MODE_STANDBY. から復帰した直後に低消費電力モードを LPM_LOW_POWER_MODE_SLEEP に戻す必要があります。

- フラッシュの停止 (コード フラッシュの無効化) はサポートされていません。S124、S128、S3A3、または S3A7 の Synergy ハードウェアユーザーズマニュアルの「フラッシュ操作コントロールレジスタ (FLSTOP)」を参照してください。
- ソフトウェアスタンバイモードでの SRAM 保持領域の縮小はサポートされていません。S3A7 または S3A3 の Synergy ハードウェアユーザーズマニュアルの「省電力メモリコントロールレジスタ (PSMCR)」を参照してください。
- MCU はデバッガが付属した状態のソフトウェア スタンバイ モードやディープ ソフトウェア スタンバイ モードに移行したり、状態を維持したりすることはできません。代わりに、MCU はデバッガによって、ソフトウェアスタンバイモードやディープソフトウェアスタンバイモードから復帰することができます。ソフトウェアスタンバイモードやディープソフトウェアスタンバイモードを適切にテストして確認するには、デバッガをアタッチしないで行う必要があります。
- メイン発振器またはメイン発振器ソースを備えた PLL をシステム クロックに使用している場合、スタンバイ モードからの復帰時間は MOSCWTCR レジスタでのメイン発振器待ち時間により影響を受ける可能性があります。このレジスタ設定は、CGC HAL モジュールのプロパティの [Main Oscillator Wait Time] の設定で変更できます。詳細については、「電気的特性」の表「ウェイクアップのタイミングおよび時間」を参照してください。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.31.4 アプリケーションへの LPM HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに LPM HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

LPM ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(LPM ドライバーのデフォルト名は g_lpm0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

LPM HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_lpm0 Low Power Modes Driver on r_lpm	Threads	New Stack> Driver> Power> Low Power Modes on r_lpm

次の図に示すように、r_lpm の LPM ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

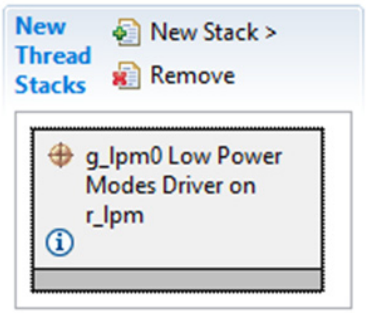


図 252:LPM HAL モジュールのスタック

4.2.31.5 LPM HAL モジュールの構成

ユーザーは必要な動作に合わせて LPM HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプ

ロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_lpm での LPM HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking
Name	g_lpm0	Module name
Operating power mode	High speed operating mode, Middle speed operating mode, Low speed operating mode, Low voltage operating mode Default: High speed operating mode	Operating power mode selection
Sub-oscillator mode	Sub oscillator mode enabled, Sub oscillator mode not enabled Default: Sub oscillator mode not enabled	Sub-oscillator mode selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

LPM HAL モジュールのクロック構成

LPM HAL モジュールには、構成可能なクロック入力はありません。

LPM HAL モジュールのピン構成

LPM HAL モジュールには、構成する必要のある特定の入力ピンと出力ピンはありません。

4.2.31.6 アプリケーションでの LPM HAL モジュールの使用

アプリケーションで LPM HAL モジュールを使用する一般的な手順は次のとおりです。

1) init API を使用して、低消費電力モード HAL モジュールを初期化します。
2) lowPowerCfg API を使用して、低消費電力モードを構成します。
3) lowPowerModeEnter API を使用して、低消費電力モードに移行します。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

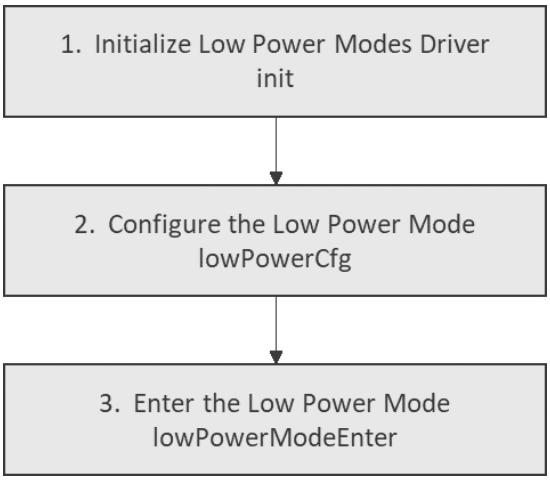


図 253:一般的な LPM HAL モジュールアプリケーションのフロー図

4.2.32 低消費電力モード V2 ドライバー

低消費電力モード V2 HAL モジュールは、低消費電力モードアプリケーション用のハイレベル API を提供し、Synergy MCU 上の低消費電力モードハードウェアペリフェラルを使用します。

4.2.32.1 LPM V2 HAL モジュールの特長

- MCU 動作電力制御モードと MCU 低消費電力モードの構成をサポートします。
- 以下の低消費電力モードをサポートしています。
  - ディープ ソフトウェア スタンバイ モード
  - ソフトウェア スタンバイ モード
  - スリープ モード
  - スヌーズ モード
- ディープスタンバイモード時に、内部の電力供給制御と I/O ポートの状態の再設定を通じて消費電力の低減をサポートします。

- MCU の他のハードウェアペリフェラルの無効化と有効化をサポートしています。

注：すべての MCU グループで、すべての低消費電力 V2 モードを利用できるとは限りません。

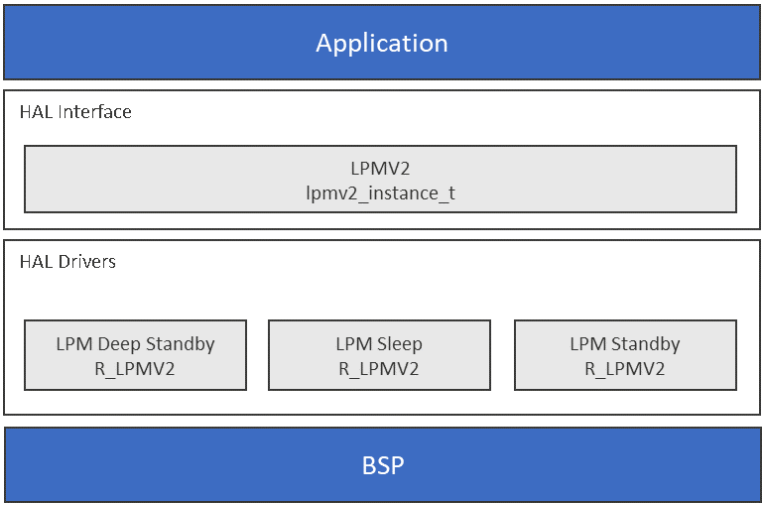


図 254:LPM V2 HAL モジュールのブロック図

4.2.32.2 LPM V2 HAL モジュールの API の概要

低消費電力モード V2 HAL モジュールでは、動作の構成および低消費電力動作の有効化と無効化のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

注：低消費電力モード V2 HAL モジュールは、MCU の動作電力制御モードを処理しなくなります。この機能は、CGC HAL モジュールによって処理されます。

次の API の例では、スリープモードの使用を示します。「"deep_standby"」と「standby」を API の例の「sleep」に置き換えて、これらのモードの例を作成できます。

LPM V2 HAL モジュールの API の要約

Function Name	Example API Call and Description
init	g_lpmv2_sleep0.p_api->init(g_lpmv2_sleep0.p_cfg); Open the LPM driver module Initialized the LPM block according to the passed in config structure.

Function Name	Example API Call and Description
lowPowerCfg	g_lpmv2_sleep0.p_api->lowPowerCfg(power_mode, output_port_enable, power_supply, io_port_state); Configure a low power mode.
lowPowerModeEnter	g_lpmv2_sleep0.p_api->lowPowerModeEnter(void); Enter low power mode (sleep/standby/deep standby) using WFI macro. Function will return after waking from low power mode.
versionGet	g_lpmv2_sleep0.p_api->versionGet(&version); Get the driver version and place it at the pointer version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_POINTER	Pointer is NULL
SSP_ERR_INVALID_MODE	Invalid settings for specified mode
SSP_ERR_INVALID_HW_CONDITION	OPCMTSF and SOPCMTSF flags are not cleared within internally set timeout.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.32.3 LPM V2 HAL モジュールの動作の概要

LPM V2 初期化

他の LPM V2 関数を呼び出す前に、LPM V2 API 関数 init を呼び出す必要があります。init 関数は、内部変数とロックの初期化を処理します。

スリープ低消費電力モード

デフォルトでは、電源投入時に低消費電力モードとしてスリープモードが有効になっています。スリープモードは、通常のプログラム実行モードに戻すために（MCU をスリープから復帰させる適切な割り込みまたはイベントを構成して有効にすることを除き）特別な構成を必要としないため、使用できる最も便利な低消費電力モードです。任意の割り込みが発生すると、MCU デバイスはスリープ低消費電力モードから復帰します。スリープモードでは、SRAM、プロセッサレジスタ、およびハードウェア周辺機能の状態がすべて維持され、スリープへの移行とスリープからの復帰に必要な時間は最小限で済みます。ThreadX® スレッドスケジューラが使用する Systick 割り込みを含め、割り込みが発生すると MCU デバイスはスリープモードから復帰します。他の関数の前に、LPM API 関数 init を呼び出す必要があります。LPM API 関数 lowPowerCfg を使用すると、低消費電力モードとしてスリープを使用するように MCU を構成できます。スリープモードに直接移行するには、LPM API 関数 lowPowerModeEnter を使用する必要があります。

次のコード例では、低消費電力モードとしてスリープを構成し、低消費電力スリープモードに切り替えます。この例の LPM V2 スリープモジュールは、g_lpmv2_sleep0: という名前を使用しています。

```
/* HAL-only entry function */
#include "hal_data.h"

void hal_entry(void)

{

    ssp_err_t error = SSP_SUCCESS;

    /* Initialize the LPM V2 Driver */
    error = g_lpmv2_sleep0.p_api->init();

    /* Handle error if any */

    /* Configure LPM peripheral for sleep mode */
    error = g_lpmv2_sleep0.p_api->lowPowerCfg(g_lpmv2_sleep0.p_cfg);

    /* Handle error if any */

    /* Entry sleep mode */
    error = g_lpmv2_sleep0.p_api->lowPowerModeEnter();

    /* Handle error if any */

}
```

LPM V2 のソフトウェアスタンバイモード

ソフトウェアスタンバイモードでは、CPU、オンチップ周辺機能の大部分、およびすべての内部発振器が停止します。保持されるのは、CPU 内蔵レジスタと SRAM データの内容、オンチップ周辺機能の状態、I/O ポートです。ソフトウェアスタンバイモードでは大半の発振器が停止するため、消費電力を大幅に削減できます。スリープモードと同様、

スタンバイモードの場合にも、スタンバイモードからの復帰のために割り込みまたはイベントを構成して有効化する必要があります。

便宜上、スタンバイモードから復帰するために使用できるトリガが、[Properties] ウィンドウに列挙されています。複数のトリガを有効にできます。

次のコード例では、低消費電力モードとしてスタンバイを構成し、低消費電力スタンバイモードに切り替えます。この例では、nameg_lpmv2_standby0 という名前の LPM V2 スタンバイモジュールを使用しています。スヌーズを有効にしてスタンバイモジュールを使用する例も同じです。

```
/* HAL-only entry function */
#include "hal_data.h"

void hal_entry(void)
{

    ssp_err_t error = SSP_SUCCESS;

    /* Initialize the LPM V2 Driver */
    error = g_lpmv2_standby0.p_api->init();

    /* Handle error if any */

    /* Configure LPM peripheral for standby mode */
    error = g_lpmv2_standby0.p_api->lowPowerCfg(g_lpmv2_standby0.p_cfg);

    /* Handle error if any */

    /* Entry standby mode */
    error = g_lpmv2_standby0.p_api->lowPowerModeEnter();

    /* Handle error if any */
}
```

LPM V2 のソフトウェアスタンバイモードを使用したスヌーズモード

スヌーズモードは、スタンバイモードの LPM V2 インスタンスで使用できます。[Properties] ウィンドウの [Choose the low power mode] で [Standby with Snooze Enabled] を選択します。スヌーズモードを MCU 周辺機器で使用すると、MCU を低消費電力状態に維持しながら、基本タスクを実行することができます。スヌーズ設定は、[Properties] ウィンドウのスタンバイ設定の下にあります。ADC、DTC、他の周辺機能を、スヌーズモードで有効にできます。レジスタ SELSR0 および IELSRn のイベントリンクコントローラ設定を除き、スヌーズのすべての設定は、スタンバイインスタンスの構成プロパティで使用できます。スヌーズは高度な機能とみなされています。

次の画面キャプチャで示すように、[Snooze Mode Settings] は、低消費電力モードの選択が [Standby with Snooze Enabled] である場合にのみ使用されます。

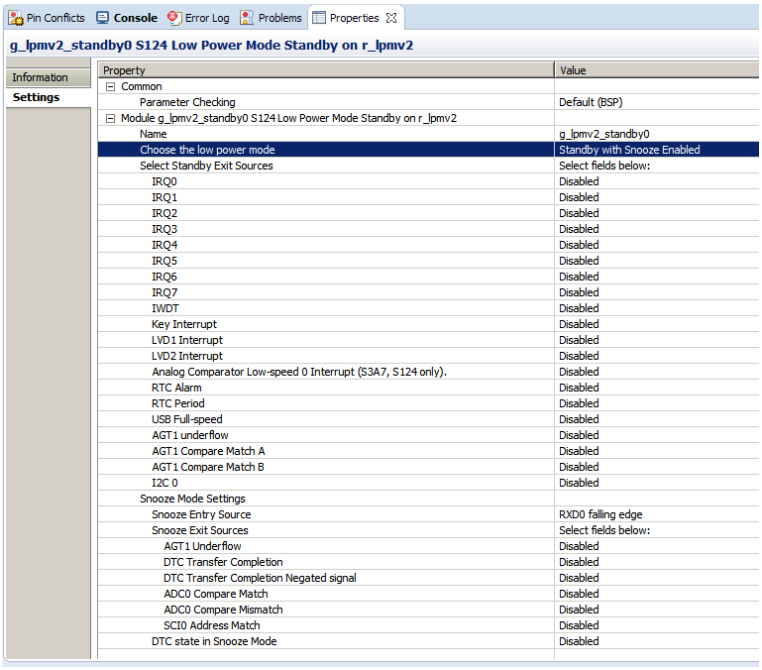


図 255:LPM V2 HAL モジュールのスヌーズの設定が有効になっているスタンバイモード

スヌーズはスタンバイモードの機能の 1 つです。これを使用すると、MCU コアが命令を実行していなくても、一部のペリフェラルを動作させることができます。スヌーズモードに関連する低消費電力モード周辺機能のオプションを、次の図に示します。有効にできるスヌーズ開始ソースは 1 つだけです。複数のスヌーズ開始ソースを有効にすることはできません。DTC ペリフェラルもスヌーズモードで有効にできます。

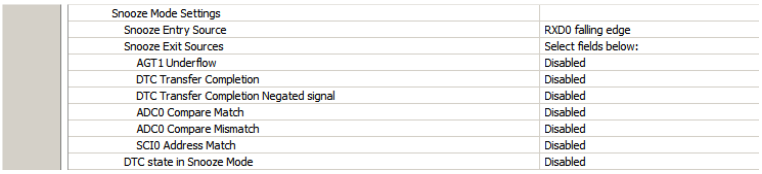


図 256:LPM V2 HAL モジュールのスヌーズモードの設定

LPM V2 のディープソフトウェアスタンバイモード

ディープソフトウェアスタンバイモードは、一部の MCU デバイスでのみ利用できます。リセットピンのネゲート、またはLPMディープスタンバイインスタンスの構成 [Properties] ウィンドウに表示されるスリープ解除イベントのセットのいずれかにより、MCU デバイスは常にリセットを経てディープソフトウェアスタンバイモードから復帰します。

便宜上、ディープスタンバイモードから復帰するために使用できるトリガが [Properties] ウィンドウ内に列挙されています。複数のトリガを有効にすることができます。一部のトリガには、関連するエッジタイプ（立ち上がりまたは立ち下がり）があります。これらのオプションについても、上および下のウィンドウに列挙されています。

次の例では、低消費電力モードとしてディープスタンバイを構成し、低消費電力ディープスタンバイモードに切り替えます。この例では、nameg_lpmv2_deep_standby0 という名前の LPM V2 ディープスタンバイモジュールを使用しています。

```
/* HAL-only entry function */
#include "hal_data.h"
```

```
void hal_entry(void)
{

    ssp_err_t error = SSP_SUCCESS;

    /* Initialize the LPM V2 Driver */
    error = g_lpmv2_deep_standby0.p_api->init();

    /* Handle error if any */

    /* Configure LPM peripheral for deep sleep mode */
    error = g_lpmv2_deep_standby0.p_api->lowPowerCfg(g_lpmv2_deep_standby0.p_cfg
);

    /* Handle error if any */

    /* Entry deep sleep mode */
    error = g_lpmv2_deep_standby0.p_api->lowPowerModeEnter();

    /* Handle error if any */
```

LPM V2 HAL モジュールの動作に関する重要な注意事項と制限事項

このドライバーを使用して LPM ペリフェラルを構成し、割り込みによってスタンバイモードから MCU を復帰させるには、割り込みを構成し、その割り込みを使用する周辺機能ドライバーまたはフレームワークごとに割り込みを有効にする必要があります。たとえば、AGT1 アンダーフローによってスタンバイから復帰させるには、AGT タイマモジュールの構成で割り込みを有効にする必要があります。

メイン発振器またはメイン発振器ソースを備えた PLL をシステムクロックに使用している場合、スタンバイ モードからの復帰時間は、MOSCWTCR レジスタでのメイン発振器待ち時間の設定により影響を受ける可能性があります。このレジスタ設定は、CGC HAL モジュールのプロパティの [Main Oscillator Wait Time] の設定で変更できます。詳細については、「電気的特性」の「ウェイクアップのタイミングおよび時間」表を参照してください。

注 : プロジェクトで ThreadX と低消費電力モードのスタンバイ、ディープスタンバイ、またはスヌーズを有効にしたスタンバイを使用する場合、lowPowerModeEnter API 関数をコールする直前に、lowPowerCfg API 関数をコールする必要があります。これが必要なのは、ThreadX もアイドルループと tx_thread_sleep 関数で低消費電力モードを使用するためです。ThreadX では、MCU デバイスが低消費電力モードスリープ用に構成されていることが想定されています。

注 : プロジェクトで ThreadX と低消費電力モードのスタンバイまたはスヌーズを有効にしたスタンバイを使用する場合、lowPowerModeEnter 関数から戻った後、MCU デバイスがスタンバイから復帰してから低消費電力モードをスリープに戻す必要があります。これが必要なのは、ThreadX もアイドルループと tx_thread_sleep 関数で低消費電力モードを使用するためです。ThreadX では、MCU デバイスが低消費電力モードスリープ用に構成されていることが想定されています。tx_thread_sleep() 関数がプロジェクトで使用されているか、実行待ち状態のスレッドが常にあるとは限らない場合は、低消費電力モードをスリープに再構成するために、lowPowerModeEnter の前に API 関数 lowPowerCfg を再度コールする必要があります。

動作状態および低消費電力モード V2 の MCU デバイスの予想消費電力の詳細については、『MCU Synergy ハードウェアユーザーズマニュアル』の「電気的特性」セクションの「動作電流とスタンバイ電流」を参照してください。

- フラッシュの停止 (コード フラッシュの無効化) はサポートされていません。S1/S3 ファミリの Synergy ハードウェアユーザーズマニュアルの「フラッシュ操作コントロールレジスタ（FLSTOP）」セクションを参照してください。

- ソフトウェアスタンバイモードでの SRAM 保持領域の縮小はサポートされていません。『S3 Synergy ハードウェアユーザーズマニュアル』の「省電力メモリコントロールレジスタ（PSMCR）」セクションを参照してください。
- MCU はデバッガが付属した状態のソフトウェア スタンバイ モードやディープ ソフトウェア スタンバイ モードに移行したり、状態を維持したりすることはできません。代わりに、MCU はデバッガによって、ソフトウェアスタンバイモードやディープソフトウェアスタンバイモードから復帰することができます。ソフトウェアスタンバイモードやディープソフトウェアスタンバイモードを適切にテストして確認するには、デバッガをアタッチしないで行う必要があります。
- メイン発振器またはメイン発振器ソースを備えた PLL をシステム クロックに使用している場合、スタンバイ モードからの復帰時間は MOSCWTCR レジスタでのメイン発振器待ち時間により影響を受ける可能性があります。このレジスタ設定は、CGC HAL モジュールのプロパティの [Main Oscillator Wait Time] の設定で変更できます。詳細については、「電気的特性」の表「ウェイクアップのタイミングおよび時間」を参照してください。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.32.4 アプリケーションへの LPM V2 HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに LPM V2 HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

LPM V2 ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(LPM V2 ドライバーのデフォルト名は g_lpm2_< モード >0 です。この名前は、対応する [Properties] ウィンドウで変更できます。)

LPM V2 HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_lpmv2_deep_standby0 S7G2 Low Power Mode Sleep on r_lpmv2	Threads	New Stack> Driver> Power> Low Power Mode Deep Standby on r_lpmv2
g_lpmv2_sleep0 S7G2 Low Power Mode Sleep on r_lpmv2	Threads	New Stack> Driver> Power> Low Power Mode Sleep on r_lpmv2
g_lpmv2_standby0 S7G2 Low Power Mode Sleep on r_lpmv2	Threads	New Stack> Driver> Power> Low Power Mode Standby on r_lpmv2

次の図に示すように、r_lpm2 の LPM V2 ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキスト

が赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

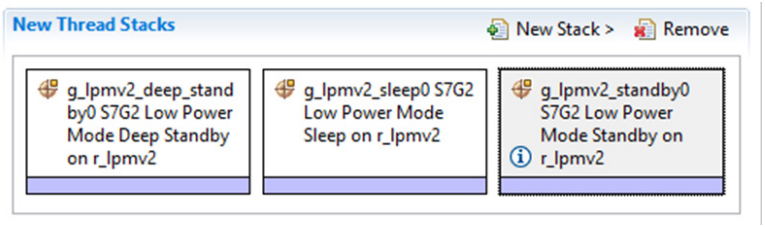


図 257:LPM V2 HAL モジュールのスタック

4.2.32.5 LPM V2 HAL モジュールの構成

ユーザーは必要な動作に合わせて LPM V2 HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_lpm2 での LPM ディープスタンバイモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking
Name	g_lpmv2_deep_standby	Module name
Output port state in standby and deep standby, applies to address output, data output, and other bus control output pins	High impedance state, No change Default: No change	Output port state selection

ISDE Property	Value	Description
Maintain or reset the IO port states on exit from deep standby mode	Maintain the IO port states, Reset the IO port states Default: Maintain the IO port states	Maintain/reset I/O port states selection
Internal power supply control in deep standby mode	Maintain the internal power supply, Cut the power supply to standby RAM, low-speed on-chip oscillator, AGTn, and USPFS/HS resume detecting unit, Cut the power supply to LVDn, standby RAM, low-speed on-chip oscillator, AGTn, and USBFS/HS resume detecting unit Default: Maintain the internal power supply	Internal power supply control selection
IRQ0-15	Enabled, Disabled Default: Disabled	IRQ0-15 selection
IRQ0-15 Edge	Disabled, Rising Edge, Falling Edge Default: Disabled	IRQ0-15 Edge selection
LVD1	Enabled, Disabled Default: Disabled	LVD1 selection
LVD1 Edge	Disabled, Rising Edge, Falling Edge Default: Disabled	LVD1 Edge selection
LVD2	Enabled, Disabled Default: Disabled	LVD2 selection

ISDE Property	Value	Description
LVD2 Edge	Disabled, Rising Edge, Falling Edge Default: Disabled	LVD2 Edge selection
RTC Interval	Enabled, Disabled Default: Disabled	RTC Interval selection
RTC Alarm	Enabled, Disabled Default: Disabled	RTC Alarm selection
NMI	Enabled, Disabled Default: Disabled	NMI selection
NMI Edge	Disabled, Rising Edge, Falling Edge Default: Disabled	NMI Edge selection
USBFS	Enabled, Disabled Default: Disabled	USBFS selection
UBSHS	Enabled, Disabled Default: Disabled	UBSHS selection
AGT11	Enabled, Disabled Default: Disabled	AGT11 selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_lpm2 上の LPM スリープモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Enables or disables the parameter checking
Name	g_lpmv2_sleep0	Module name

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_lpm2 上の LPM スタンバイモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking
Name	g_lpmv2_standby0	Module name
Choose the low power mode	Standby, Standby with snooze Enabled Default: Standby	Low power mode selection
Output port state in standby and deep standby, applies to address output, data output, and other bus control output pins	High impedance state, No change Default: No change	Output port state selection
IRQ1-15	Enabled, Disabled Default: Disabled	IRQ1-15 selection
IWDT	Enabled, Disabled Default: Disabled	IWDT selection

ISDE Property	Value	Description
Key Interrupt	Enabled, Disabled Default: Disabled	Key Interrupt selection
LVD1 Interrupt	Enabled, Disabled Default: Disabled	LVD1 Interrupt selection
LVD2 Interrupt	Enabled, Disabled Default: Disabled	LVD2 Interrupt selection
Analog Comparator High-speed 0 Interrupt	Enabled, Disabled Default: Disabled	Analog Comparator High-speed 0 Interrupt selection
RTC Alarm	Enabled, Disabled Default: Disabled	RTC Alarm selection
RTC Period	Enabled, Disabled Default: Disabled	RTC Period selection
USB High-speed	Enabled, Disabled Default: Disabled	USB High-speed selection
USB Full-speed	Enabled, Disabled Default: Disabled	USB Full-speed selection
AGT1 underflow	Enabled, Disabled Default: Disabled	AGT1 underflow selection

ISDE Property	Value	Description
AGT1 Compare Match A	Enabled, Disabled Default: Disabled	AGT1 Compare Match A selection
AGT1 Compare Match B	Enabled, Disabled Default: Disabled	AGT1 Compare Match B selection
12C 0	Enabled, Disabled Default: Disabled	12C 0 selection
Snooze Entry Source	RXD0 falling edge, IRQ0-IRQ15, KINT, ACMPHS0, RTC Alarm, RTC Period, AGT1 Underflow, AGT1 Compare Match A, AGT1 Compare Match B Default: RXD0 falling edge	Snooze Entry Source selection
AGT1 Underflow	Enabled, Disabled Default: Disabled	AGT1 Underflow selection
DTC Transfer Completion	Enabled, Disabled Default: Disabled	DTC Transfer Completion selection
DTC Transfer Completion Negated Signal	Enabled, Disabled Default: Disabled	DTC Transfer Completion Negated Signal selection
ADC0 Compare Match	Enabled, Disabled Default: Disabled	ADC0 Compare Match selection

ISDE Property	Value	Description
ADC0 Compare Mismatch	Enabled, Disabled Default: Disabled	ADC0 Compare Mismatch selection
ADC1 Compare Match	Enabled, Disabled Default: Disabled	ADC1 Compare Match selection
ADC1 Compare Mismatch	Enabled, Disabled Default: Disabled	ADC1 Compare Mismatch selection
SCI0 Address Match	Enabled, Disabled Default: Disabled	SCI0 Address Match selection
DTC state in Snoooze Mode	Enabled, Disabled Default: Disabled	DTC state in Snoooze Mode selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

デフォルト以外の設定が望ましい場合もあります。たとえば、低消費電力状態の開始または終了に異なる状態を選択する場合に役立つことがあります。

LPM V2 HAL モジュールのクロック構成

LPM V2 ペリフェラルモジュールには、選択可能なクロックソースはありません。

LPM V2 HAL モジュールのピン構成

LPM V2 ペリフェラルモジュールでは、ピン配置は必要ありません。ピン機能の選択は、プロパティ構成ウィンドウで行います。

4.2.32.6 アプリケーションでの LPM V2 HAL モジュールの使用

アプリケーションで LPM V2 HAL モジュールを使用する一般的な手順は次のとおりです。

1) init API を使用して、低消費電力モード V2 HAL モジュールを初期化します。
2) lowPowerCfg API を使用して、低消費電力モードを構成します。
3) lowPowerModeEnter API を使用して、低消費電力モードに移行します。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

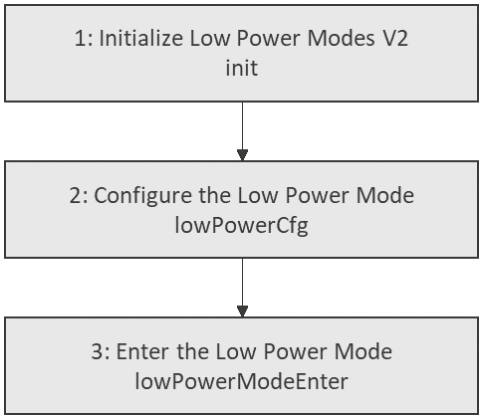


図 258:一般的な LPM V2 HAL モジュールアプリケーションのフロー図

4.2.33 低電圧検出ドライバー

低電圧検出（LVD）HAL モジュールは、電圧検出アプリケーション用のハイレベル API を提供し、Synergy MCU 上の LVD ペリフェラルを使用します。ユーザー定義のコールバックを作成し、電圧検出イベントがトリガされたときに CPU に通知できます。VCC は、すべての電圧検出機能のソースです。

4.2.33.1 LVD HAL モジュールの特長

LVD HAL モジュールは、次の機能をサポートしています。

- 電圧検出入力としての VCC
- 1 つのビルド時に構成可能な低電圧検出（OFS1 レジスタを使用）
- 2 つの実行時に構成可能な低電圧検出
- 2 つの結果フラグ。1 つはしきい値チェック用、もう 1 つは現在の状態用
- 割り込みとポーリングイベントチェックの両方をサポート

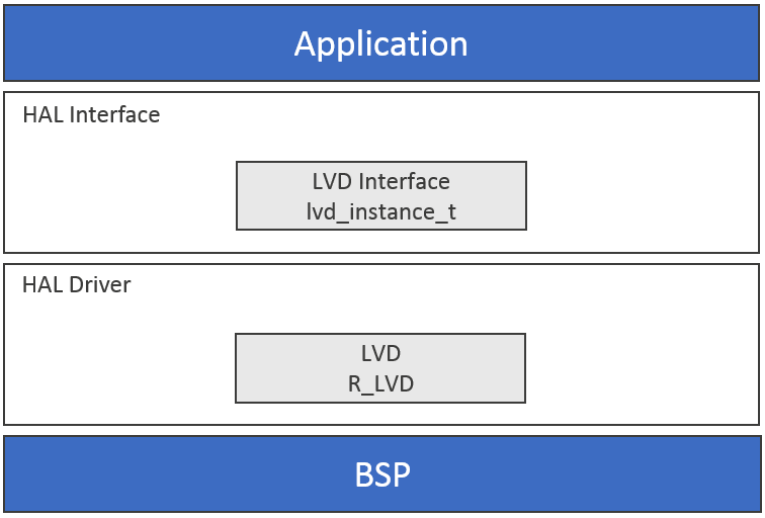


図 259:**LVD HAL モジュールのブロック図**

4.2.33.2 LVD HAL モジュールの API の概要

LVD HAL モジュールでは、オープン、クローズ、statusGet、statusClear のための API が定義されています。次の表に、使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明を示します。ステータス戻り値の表は API 要約表の後にあります。

LVD HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_lvd.p_api->open(g_lvd.p_ctrl, g_lvd.p_cfg; Initializes a low voltage detection driver according to the passed in configuration structure. Enables an LVD peripheral based on configuration structure.

Function Name	Example API Call and Description
statusGet	g_lvd.p_api->statusGet(g_lvd.p_ctrl, &monitor_status); Get the current state of the monitor, (threshold crossing detected, voltage currently within range) Can be used to poll the state of the LVD monitor at any time. Must be used if the peripheral was initialized with the lvd_response_t set to LVD_RESPONSE_NONE.
statusClear	g_lvd.p_api->statusClear(g_lvd.p_ctrl); Clears the latched status of the monitor. Must be used if the peripheral was initialized with lvd_response_t set to LVD_RESPONSE_NONE.
close	g_lvd.p_api->close(g_lvd.p_ctrl, g_lvd.p_cfg); Disables the LVD peripheral. Closes the driver instance.
versionGet	g_lvd.p_api->versionGet(&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_IN_USE	Driver already open or unable to acquire hardware lock
SSP_ERR_NOT_OPEN	Unit is not open

Name	Description
SSP_ERR_ASSERTION	Invalid configuration value
SSP_ERR_INVALID_MODE	If the attempted mode is invalid for this configuration

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.33.3 LVD HAL モジュールの動作の概要

LVD HAL モジュールは、Synergy MCU での LVD 監視の構成と操作をサポートします。LVD HAL モジュールは、単一の LVD 監視の完全な構成に必要なすべての情報を提供する、構成構造体を提供します。LVD0 の監視を除き、LVD 監視の各インスタンスに LVD HAL モジュールのインスタンスが 1 つ必要です。これは実行時の構成が不可能であるため、OFS1 レジスタを介してコンパイル時に構成する必要があります。

LVD1 と LVD2 の監視は、どちらも実行時に構成可能で、構成はこのモジュールを使用して行います。open 関数を使用すると、1 回の関数コールで LVD 監視を構成して有効にすることができます。close 関数は LVD 監視を無効化します。statusGet 関数は、LVD 監視の現在のステータスを返します。モジュールがポーリングモードの場合は（つまり、LVD 監視割り込みが有効になっていない場合）、statusGet 関数を使用する必要があります 。

監視のステータスは 2 つのフラグで構成されます。最初のフラグは crossing_detected と呼ばれるラッチフラグで、監視対象の電圧が電圧しきい値を超えたかどうかを示します。ポーリングモードでは、このフラグは statusClear のコールを通じてクリアする必要があります。LVD 割り込みが使用中である場合を除き、このフラグを明示的にクリアする必要はありません。LVD 割り込みでは、このフラグはユーザーコールバック関数のコール後にドライバーコードによりクリアされます。2 番目のフラグである current_state は監視対象の電圧の電圧しきい値に対する瞬間的な状態です。このフラグはラッチされず、その状態は監視対象の電圧が変化すると変わります。

LVD HAL モジュールは、1 つまたは複数の LVD 周辺機能割り込みを有効にするように構成できます。割り込みを使用する場合、ユーザーはその監視に対するコールバック関数を提供する必要があります。LVD 監視ごとに異なるコールバックルーチンを提供する必要があります。

LVD HAL モジュールには、BSP によって提供される機能が必要です。このドライバーは、BSP によって提供されるハードウェアロックを利用することで、レジスタのロックを行い、また、割り込みの有効化とクリアを有効にします。

LVD HAL モジュールの動作に関する重要な注意事項と制限事項

- これらの LVD 設定に対して適切な値を選択した後、ユーザーは LVD HAL モジュールの open API 関数をコールするコードをプロジェクトに追加する必要があります。この関数は、アプリケーションの早い段階で 1 回呼び出す必要があります。
- LVD 監視の構成を変更する必要があるときは常に、モジュールを閉じて再度開きます。LVD open API 関数をコールすると、指定された LVD 監視の LVD ハードウェアペリフェラルが構成されて有効になります。
- close 関数は、LVD 監視を無効にして、ドライバーを閉じます。
- このモジュールを使用して LVD ペリフェラルを構成し、割り込みを作成するには、モジュールのプロパティタブで対応する割り込みが有効にされている必要があります。
- LVD 割り込みを使用する際、コールバック関数は必須ではありませんが、使用が推奨されます。
- 各 LVD 割り込みに対する固有のコールバック関数は必須ではありませんが推奨されます。
- クロックシステムの初期化、構成、実行時の変更は、このモジュールの外で処理されます。このドライバーは、ユーザーが選択したサンプル クロックの除算に基づくデジタル フィルター サンプル クロッ

クのみを変更します。このデジタル フィルター サンプル クロックは、LOCO システム クロックから得られたものです。

- すべての MCU ですべての電圧しきい値が利用可能とは限りません。
- すべての MCU で LVD 監視に対する VCC 入力のデジタルフィルタリングが利用可能とは限りません。
- LVD ドライバーには、BSP によって提供される機能が必要です。このドライバーは、BSP によって提供されるハードウェアロックを使用して、レジスタのロックおよび割り込みの有効化とクリアを行います。
- 低電圧検出監視を構成 / 有効化するプロセスは、具体的な時間的制約とレジスタの書き込み順序に拘束されます。このような制約があるため、電圧監視を設定 / 有効化するプロセスは、全体を単一の関数で処理するのが最も効率的です。open API 関数は、タイミングとレジスタのライト順序の制約が正しく適用されるように、構成を実行し、監視を有効にします。
- すべての Synergy マイクロコントローラのシリーズは、オプション設定メモリを備えています。このメモリを使用して、リセット後の周辺機能の動作状態を設定できます。OFS は、IWDT、WDT、LVD、CGC HOCO の状態の設定に使用できます。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.33.4 アプリケーションへの LVD HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに LVD HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

LVD ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(LVD ドライバーのデフォルト名は g_lvd0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

LVD HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_lvd Low Voltage Detection Driver on r_lvd	Threads	New Stack> Driver> Power> Low Voltage Detection Driver on r_lvd

次の図に示すように、r_lvd の LVD ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。

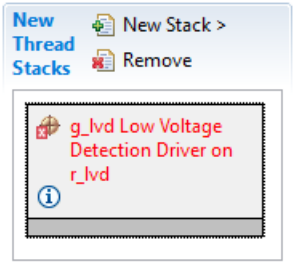


図 260:**LVD HAL モジュールのスタック**

4.2.33.5 LVD HAL モジュールの構成

ユーザーは必要な動作に合わせて LVD HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_lvd での LVD HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking
Name	g_lvd	Module name
Monitor number	1	Monitor number selection

ISDE Property	Value	Description
Digital filter, enable by selecting a valid sample clock rate (S7G2 only).	Digital filter is disabled, Digital filer is enabled (sampling clock is LOCO/2), Digital filer is enabled (sampling clock is LOCO/4), Digital filer is enabled (sampling clock is LOCO/8), Digital filer is enabled (sampling clock is LOCO/16) Default: Digital filter is disabled	Digital filter selection
Voltage Threshold	Default: 2.85V (Vdet1_13)(S7G2 only).	Voltage threshold selection
Detection Response, either reset, interrupt, non-maskable interrupt, or no response (polling mode).	Maskable interrupt triggered when voltage crosses the detection threshold, Non-maskable interrupt triggered when voltage crosses the detection threshold, Reset MCU when voltage falls below the detection threshold, No response driver will be in polled mode (using statusGet and statusClear functions). Default: Maskable interrupt triggered when voltage crosses the detection threshold	Detection response selection
Voltage slope, rising or falling or both	Threshold crossing detected with decreasing voltage, Threshold crossing with increasing voltage, Threshold crossing detected with increasing or decreasing voltage Default: Threshold crossing detected with decreasing voltage	Indicates the direction of voltage change detection in relation to the threshold

ISDE Property	Value	Description
Negation of the monitor signal can be either be delayed from the reset event or from voltage returning to normal range	Negation of reset signal is based on delay from reset, Negation of reset signal is based on delay from voltage returning to normal range Default: Negation of reset signal is based on delay from reset	Negation of the monitor signal selection
Monitor Interrupt Callback	NULL	Monitor interrupt callback selection
LVD Monitor Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	LVD monitor interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

LVD HAL モジュールのクロック構成

クロック システムのクロックの初期化、構成、実行時変更は、このモジュール外で処理されます。このモジュールは、ユーザーが選択したサンプルクロックの除数に基づいてデジタルフィルタサンプルクロックのみを変更します。このデジタル フィルター サンプル クロックは、LOCO システム クロックから得られたものです。

LVD HAL モジュールのピン構成

LVD HAL モジュールは、VCC ピンのみで電圧を測定し、構成する必要はありません。

4.2.33.6 アプリケーションでの LVD HAL モジュールの使用

アプリケーションで LVD HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、LVD HAL モジュールを初期化します。
2) ソフトウェアポーリングを使用する場合は、statusGet API で LVD の状態フラグを監視し、適切に処理します。割り込みモードを使用する場合は、監視番号と状態の両方を返すコールバック関数の中で適切に処理します。
3) 必要に応じて処理します
4) close API を使用して、LVD インスタンスを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

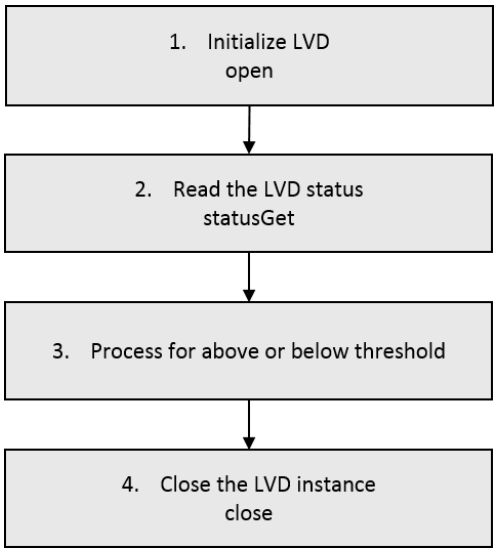


図 261:一般的な LVD HAL モジュールアプリケーションのフロー図

4.2.34 OPAMP ドライバー

OPAMP HAL モジュールは、信号増幅アプリケーション用のハイレベル API を提供し、Synergy MCU で利用可能な OPAMP ペリフェラルをサポートします。

4.2.34.1 OPAMP HAL モジュールの特長

- 低消費電力モードまたは高速モード
- ソフトウェアまたは AGT 比較一致による開始
- ソフトウェアまたは ADC 変換終了による停止（ADC 変換終了による停止は、AGT 比較一致による開始が構成されたオペアンプチャネルでのみサポート）
- 一部の MCU でトリミングが利用可能（ハードウェアマニュアルを参照）

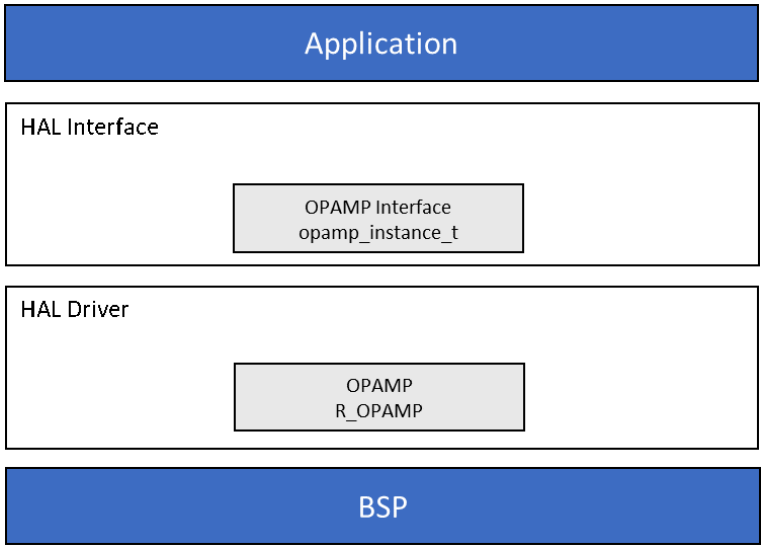


図 262:**OPAMP HAL モジュールのブロック図**

4.2.34.2 OPAMP HAL モジュールの API の概要

OPAMP HAL モジュールでは、モジュールのオープン、開始、停止、ステータスのリード、トリム、クローズの API 関数が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

OPAMP HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_opamp0.p_api->open(g_opamp0.p_ctrl, g_opamp0.p_cfg); Applies power to the OPAMP and initializes the hardware based on the user configuration.
start	g_opamp0.p_api->start(g_opamp0.p_ctrl, channel_mask); If the OPAMP is configured for hardware triggers, enables hardware triggers. Otherwise, starts the op-amp.

Function Name	Example API Call and Description
stop	g_opamp0.p_api->stop(g_opamp0.p_ctrl, channel_mask); Stops the op-amp. If the OPAMP is configured for hardware triggers, disables hardware triggers.
trim	g_opamp0.p_api->trim(g_opamp0.p_ctrl, cmd, p_args); On MCUs that support trimming, the op-amp trim register is set to the factory default after open(). This function allows the application to trim the operational amplifier to a user setting, which overwrites the factory default factory trim values. See Operational Notes for important details.
infoGet	g_opamp0.p_api->infoGet(g_opamp0.p_ctrl, p_info); Provides the minimum stabilization wait time in microseconds.
statusGet	g_opamp0.p_api->statusGet(g_opamp0.p_ctrl, p_status); Provides the operating status for each op-amp in a bitmask. This bit is set when operation begins, before the stabilization wait time has elapsed.
close	g_opamp0.p_api-> close(g_opamp0.p_ctrl); Stops the op-amps.
versionGet	g_opamp0.p_api->versionGet(&version); Retrieve the module version using the version pointer

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_NOT_OPEN	Unit is not open
SSP_ERR_ASSERTION	An input pointer is NULL.
SSP_ERR_IN_USE	Peripheral is in use or hardware lock is taken.
SSP_ERR_INVALID_POINTER	The parameter p_data is NULL.
SSP_ERR_INVALID_STATE	The command is not valid for the current state of the trim function.
SSP_ERR_INVALID_MODE	Trim is not allowed in low power mode.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.34.3 OPAMP HAL モジュールの動作の概要

OPAMP HAL モジュールは、Synergy マイクロコントローラ上の OPAMP ペリフェラルを制御します。RTOS 要素を使用しないで OPAMP ハードウェアを直接制御し、開発が簡単になる便利な API を提供します。

OPAMP HAL モジュールの動作に関する重要な注意事項と制限事項

OPAMP のトリミング

トリミングがサポートされる MCU では、open API がコールされた後、オペアンプトリムレジスタが出荷時のデフォルトに設定されます。

この機能により、アプリケーションはオペアンプを、出荷時のデフォルトトリム値を上書きするユーザー設定にトリムすることができます。

一部の MCU ではサポートされません。詳細については、ハードウェアのマニュアルを参照してください。低消費電力モード（OPAMP_MODE_LOW_POWER）. 用に構成されている場合はサポートされません。

この関数は再入可能ではありません。同時にトリムできるのはオペアンプの片側だけです。1 つのチャネルの片側について手順を完了してから、コマンド OPAMP_TRIM_CMD_START で trim API を再度コールします。

トリムの手順は、次のように実行します。

- Pch（+）側入力について、コマンド OPAMP_TRIM_CMD_START. で trim API をコールします。
- 一定電圧を Pch（+）入力に接続します。
- Nch（-）入力をオペアンプの出力に接続して、電圧フォロワを作成します。

- オペアンプが動作し、安定していることを確認します。
- Pch（+）側入力について、コマンド OPAMP_TRIM_CMD_START. で trim API をコールします。
- SAR ADC を使用して Pch（+）側入力に接続されている一定電圧を測定し、値（この手順でこの後、A として参照します）を保存します。
- 以下のループ処理を 5 回繰り返します。
  - Pch（+）側入力について、コマンド OPAMP_TRIM_CMD_NEXT_STEP. で trim API をコールします。
  - SAR ADC を使用して、オペアンプの出力を測定します（次のステップで B として参照します）。
  - A<=B の場合は、Pch（+）側入力について、コマンド OPAMP_TRIM_CMD_CLEAR_BIT. で trim API をコールします。
- Nch（-）側入力について、コマンド OPAMP_TRIM_CMD_START. で trim API をコールします。
- SAR ADC を使用して Pch（+）側入力に接続されている一定電圧を測定し、値（この手順でこの後、A として参照します）を保存します。
- 以下のループ処理を 5 回繰り返します。
  - Nch（-）側入力について、コマンド OPAMP_TRIM_CMD_NEXT_STEP. で trim API をコールします。
  - SAR ADC を使用して、オペアンプの出力を測定します（次のステップで B として参照します）。
  - A<=B の場合は、Nch（-）側入力について、コマンド OPAMP_TRIM_CMD_CLEAR_BIT. で trim API をコールします。

以下のステータス戻り値が trim API に関連付けられています。

ステータス戻り値

Status Return Value	Meaning
SSP_SUCCESS	Conversion result in p_data.
SSP_ERR_UNSUPPORTED	Trimming is not supported on this MCU.
SSP_ERR_INVALID_STATE	The command is not valid in the current state of the trim state machine.
SSP_ERR_INVALID_ARGUMENT	The requested channel is not operating or the trim procedure is not in progress for this channel/input combination.
SSP_ERR_INVALID_MODE	Trim is not allowed in low power mode.
SSP_ERR_ASSERTION	An input pointer was NULL.
SSP_ERR_NOT_OPEN	Instance control block is not open. Trimming is not supported on all MCUs.

このモジュールは、一部の Synergy MCU でのみ使用できます。現在の SSP リリースのリリースノートを参照して、このモジュールでサポートされている MCU を確認してください。また、使用可能なペリフェラルが MCU ハードウェアマニュアルに示されています。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.34.4 アプリケーションへの OPAMP HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに OPAMP HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

OPAMP ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(OPAMP ドライバーのデフォルト名は g_opamp0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

OPAMP HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_opamp0 OPAMP Driver on r_opamp	Threads	New Stack> Driver> Analog> OPAMP Driver on r_opamp

次の図に示すように、r_opamp の OPAMP ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

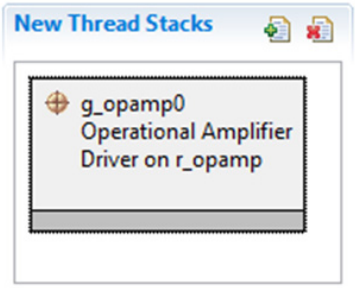


図 263:OPAMP HAL モジュールのスタック

4.2.34.5 OPAMP HAL モジュールの構成

ユーザーは必要な動作に合わせて OPAMP HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_opamp での OPAMP HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_opamp	Module name
AGT Start Trigger Configuration (N/A unless AGT Start Trigger is Selected for the Channel)	AGT1 Compare Match Starts OPAMPs 0 and 2 if configured for AGT Start/AGT0 Compare Match Starts OMPAMPs 1 and 3 if configured for AGT Start, AGT1 Compare Match Starts OPAMPs 0 and 1 if configured for AGT Start/AGT0 Compare Match Starts OPAMPs 2 and 3 if configured for AGT Start, AGT1 Compare Match Starts all OPAMPs configured for AGT Start Default: AGT1 Compare Match Starts all OPAMPs configured for AGT Start	Configure which AGT channel event triggers op-amp channel. The AGT compare match event only starts the op-amp channel if the AGT Start trigger is selected in the Trigger configuration for the channel.

ISDE Property	Value	Description
Power Mode	Low Power, Middle Speed, High Speed Default: High Speed	Configure the op-amp based on power or speed requirements. This setting affects the minimum required stabilization time. Middle speed is not available for all MCUs.
Trigger Channel 0	Software Start Software Stop, AGT Start Software Stop, AGT Start ADC Stop Default: Software Start Software Stop	Select the event triggers to start or stop op-amp channel 0. If the event trigger is selected for start, the API enables the event trigger for this channel. If the event trigger is selected for stop, the API disables the event trigger for this channel.
Trigger Channel 1	Software Start Software Stop, AGT Start Software Stop, AGT Start ADC Stop Default: Software Start Software Stop	Select the event triggers to start or stop op-amp channel 1. If the event trigger is selected for start, the API enables the event trigger for this channel. If the event trigger is selected for stop, the API disables the event trigger for this channel.
Trigger Channel 2	Software Start Software Stop, AGT Start Software Stop, AGT Start ADC Stop Default: Software Start Software Stop	Select the event triggers to start or stop op-amp channel 2. If the event trigger is selected for start, the API enables the event trigger for this channel. If the event trigger is selected for stop, the API disables the event trigger for this channel.
Trigger Channel 3	Software Start Software Stop, AGT Start Software Stop, AGT Start ADC Stop Default: Software Start Software Stop	Select the event triggers to start or stop op-amp channel 3. If the event trigger is selected for start, the API enables the event trigger for this channel. If the event trigger is selected for stop, the API disables the event trigger for this channel.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

OPAMP HAL モジュールのクロック構成

OPAMP HAL モジュールには、特定のクロック構成は必要ありません。

OPAMP HAL モジュールのピン構成

OPAMP HAL モジュールを使用するには、アナログ入力を受信するチャネルのポートピンを、ISDE のピンコンフィギュレータで入力ピンとして設定する必要があります。次の表では、ISDE [Configuration] ウィンドウでのピンの選択方法を示します。

r_opamp の OPAMP HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
OPAMP	Pins	Select Peripherals > Analog: OPAMP0/1/2

4.2.34.6 アプリケーションでの OPAMP HAL モジュールの使用

アプリケーションで OPAMP HAL モジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して、OPAMP モジュールを初期化します。
   a) 注：オペアンプを開始する際は、必ず事前にハードウェアマニュアルを参照して、使用する MCU の OPAMP ペリフェラルに内部接続スイッチがあるかどうかを確認してください。OPAMP ペリフェラルに内部接続スイッチがある場合は、AMPnMS、AMPnPS、AMP0OS の各レジスタを直接設定することによって内部接続を構成します。
2) start API で必要なトリガを使用して OPAMP チャネルを開始します。
   a) 注：AGT 比較一致による開始を使用する場合は、このコールにより、OPAMP を AGT 比較一致によってトリガできるようになります。ソフトウェアトリガを使用する場合、このコールにより OPAMP チャネルが開始されます。
3) stop API をコールすることによって OPAMP チャネルを停止します。（オプション）
   a) 注：OPAMP を停止する ADC 変換終了トリガが有効化されているかどうかに関係なく、これにより OPAMP は停止します。
4) close API を使用し、モジュールを閉じてペリフェラルの電源をオフにします。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

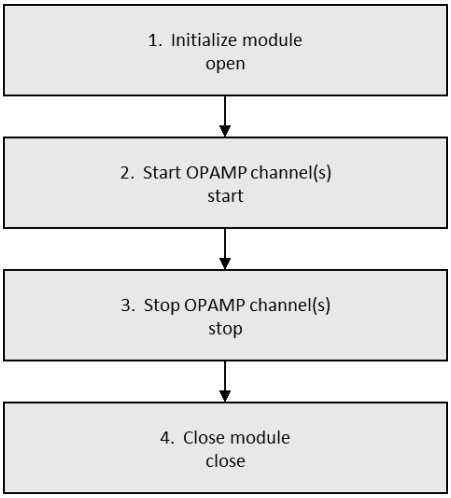


図 264:一般的な OPAMP HAL モジュールアプリケーションのフロー図

4.2.35 PDC ドライバー

パラレルデータキャプチャユニット（PDC）HAL モジュールは、カメラアプリケーションから画像をキャプチャするためのハイレベル API を提供し、Synergy MCU 上の PDC ペリフェラルを使用します。ユーザー定義のコールバックを作成し、キャプチャ完了時に CPU に通知できます。

4.2.35.1 PDC HAL モジュールの特長

- 接続された構成済みのカメラからのキャプチャをサポートします。
- キャプチャの完了を CPU に通知するコールバックをサポートします
- キャプチャバッファへのポインタを提供します
- コールバックをトリガしたイベントを示す値を提供します

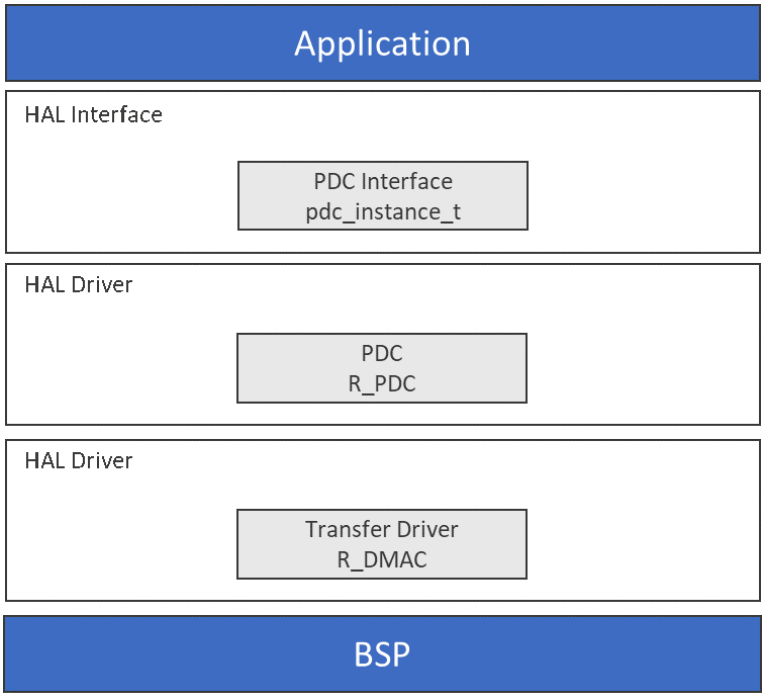


図 265:**PDC HAL モジュールのブロック図**

4.2.35.2 PDC HAL モジュールの API の概要

PDC HAL モジュールでは、データキャプチャをオープン、クローズ、開始するための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

PDC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_pdc.p_api->open(g_pdc.p_ctrl, g_pdc.p_cfg) Initial configuration.
close	g_pdc.p_api->close(g_pdc.p_ctrl)) Closes the driver and allows reconfiguration. May reduce power consumption.

Function Name	Example API Call and Description
captureStart	g_pdc.p_api->captureStart(g_pdc.p_ctrl, NULL) Start a capture.
stateGet	g_pdc.p_api->stateGet(g_pdc.p_ctrl, &state_data) Get the state of VSYNC and HSYNC pins.
versionGet	g_pdc.p_api->versionGet(&version) Return the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_ASSERTION	The parameter p_ctrl or p_sample is NULL.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_NOT_OPEN	Unit is not open.
SSP_ERR_ALREADY_OPEN	Unit is already open.
SSP_ERR_HW_LOCKED	Unable to reserve BSP hardware lock.
SSP_ERR_TIMEOUT	Reset Operation timed out.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.35.3 PDC HAL モジュールの動作の概要

キャプチャ動作には、Synergy マイクロコントローラに接続された構成済みの外部カメラが必要です。キャプチャを実行する前に、カメラを構成し、マイクロコントローラへの PIXCLK クロック入力がカメラによって生成される必要があります。一部のインスタンスでは、カメラの構成には実行中のクロック入力が必要となる場合があります。

カメラを初期化する前に、open API をコールします。この API が PDC からカメラへの PCKO クロック出力を構成して開始します。カメラの構成が済んだら、captureStart をコールしてイメージをキャプチャできます。カメラモジュールの構成には、I^2C または SPI インタフェースの使用が必要となる場合があります。

PDC HAL モジュールの動作に関する重要な注意事項と制限事項

ほとんどの場合、カメラまたは PDC ペリフェラルからのデータレートは、割り込みサービスルーチン（ISR）内の CPU による処理に対して速すぎます。そのため、このモジュールで PDC 周辺機能およびメモリからの高速転送を実行するには、DMAC に転送ドライバーの実装が必要です。

以下の状況で割り込みを生成するには、PDC フレーム終了割り込みと PDC エラー割り込みの両方を使用する必要があります。

- イメージがキャプチャされた場合の割り込み (フレーム終了)
- エラーが発生した場合の割り込み

データ バッファの設定

p_buffer が NULL 以外に設定されている場合、イメージデータを格納するために 1 つまたは複数のデータバッファが作成されます。各バッファのサイズは、以下の式を使って計算されます。

バッファサイズ（バイト）=x_capture_pixels x y_capture_pixels x bytes_per_pixel

解像度の高いカメラの場合、キャプチャしたイメージのデータが膨大な量になることがあります。このため、buffer(s) を外部メモリ（SDRAM など）に置くことが必要になる場合があります。外部メモリを使用する場合は、バス帯域を考慮してください。

たとえば、PDC を使用して高フレームレートカメラで SDRAM へのイメージキャプチャを行い、高リフレッシュレートの LCD ディスプレイ用にディスプレイバッファを保持するのに SDRAM を使用すると、PDC からメモリへのデータボトルネックが発生し、オーバーランエラー状態になります。

注 :p_buffer を NULL に設定した場合、キャプチャされたイメージデータの格納用にメモリは割り当てられません。アプリケーションは、PDC に十分なサイズの適切なメモリを確保する必要があります。PDC は接続された LCD パネルのディスプレイバッファに直接キャプチャを行うことができます。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.35.4 アプリケーションへの PDC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに PDC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

PDC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(PDC ドライバーのデフォルト名は g_pdc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

PDC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_pdc0 PDCDriver on r_pdc	Threads	New Stack> Driver> Graphics> PDC Driver on r_pdc

次の図に示すように、r_pdc の PDC ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

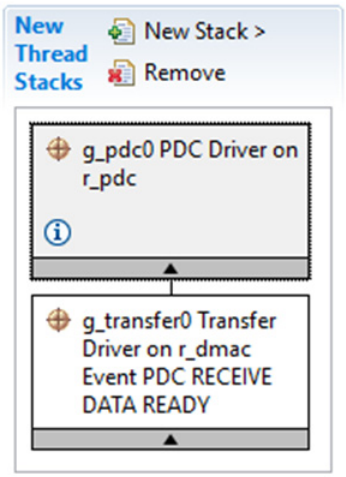


図 266:PDC HAL モジュールのスタック

4.2.35.5 PDC HAL モジュールの構成

ユーザーは必要な動作に合わせて PDC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_pdc での PDC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_pdc0	The name of the PDC module instance. Specify arbitrary C symbol.
Name of the data buffer to store image data	g_user_buffer	Specify the name of the data buffer to create or set to NULL if it is to be created by the user external to the PDC driver.
Section where data buffer is allocated	sdram	Specify the RAM section for the image data buffer. Typically bss (internal RAM) or sdram.
Number of bytes per pixel	2	Specify the number of bytes per pixel of the captured image data.
Number of image data buffers	1	Specify the number of buffers to create.
Clock Divider	CLK/2, CLK/4, CLK/6, CLK/8, CLK10, CLK12, CLK14, CLK16 Default: CLK/2	Specify the clock divider of the clock input to the PDC peripheral.
Endian of image data	Little, Big Default: Little	Specify the endian of the captured image data.

ISDE Property	Value	Description
HYSNC signal polarity	High, Low Default: High	Specify the active polarity of the HSYNC signal.
VSYNC signal polarity	High, Low Default: High	Specify the active polarity of the VSYNC signal.
Number of pixels to capture horizontally	640	Number of horizontal pixels to capture.
Number of pixels to capture vertically	480	Number of vertical lines to capture.
Horizontal pixel to start capture from	0	Horizontal pixel to start capturing image data from. Allows an image smaller than the native resolution of a camera to be captured.
Line to start capture from	0	Vertical line to start capturing image data from. Allows an image smaller than the native resolution of a camera to be captured.
Callback	g_pdc_user_callback	A user callback function can be registered in open. If this callback function is provided, it is called from the interrupt service routine (ISR) each time a frame is captured and ready to be processed. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.

ISDE Property	Value	Description
Frame End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	The driver needs a valid interrupt priority setting. It won't function if disabled.
PDC Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	The driver needs a valid interrupt priority setting. It won't function if disabled.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

PDC HAL モジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_dmac Event PDC RECEIVE DATA READY 時の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter selection.
Name	g_transfer0	Driver name.
Mode	Block	Mode selection.
Transfer Size	4 Bytes	Transfer size selection.
Destination Address Mode	Incremented	Destination address mode selection.
Source Address Mode	Fixed	Source address mode selection.
Repeat Area (Unused in Normal Mode	Source	Repeat area selection.

ISDE Property	Value	Description
Destination Pointer	NULL	Destination pointer selection.
Source Pointer	NULL	Source pointer selection.
Number of Transfers	8	Number of transfers selection.
Number of Blocks (Valid only in Block Mode)	1	Number of blocks selection.
Activation Source (Must enable IRQ)	Event PDC RECEIVE DATA READY	Activation source selection.
Auto Enable	FALSE	Auto enable selection.
Callback (Only valid with Software start)	NULL	Callback selection.
Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

PDC HAL モジュールのクロック構成

PDC は PCLKB をそのクロックソースとして使用します。このクロックの構成に関する唯一の制限事項は、PPCLKB 周波数がこれに応じて設定されるよう、PIXCLCK が 0.6 x PCLKB 未満でなければならない点です。

PDC HAL モジュールのピン構成

PDC 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では PDC ピンの選択例を示します。

r_pdc の PDC HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
PDC	Pins	Select Peripherals > Graphics: PDC > PDC0

注 : この選択シーケンスでは、KINT0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_pdc での PDC HAL モジュールのピン構成設定

Property	Value	Description
Pin Group Selection	Mixed, _A Only Default: Mixed	Pin group selection
Operation Mode	Disabled, Custom, Enabled Default: Disabled	Select Enabled as the Operation Mode for PDC
HSYNC	None, P704 Default: None	HSYNC Pin
PCKO	None, P511 Default: P511	PCKO Pin
PIXCLK	None, P705 Default: None	PIXCLK Pin
VSYNC	None, P512 Default: P512	VSNC Pin
PIXD0:7	None, Pnnn Default: None	PIX Data0:7 Pins

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

4.2.35.6 アプリケーションでの PDC HAL モジュールの使用

アプリケーションで PDC HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、PDC HAL モジュールを初期化します
2) 必要に応じてカメラを構成します

3) captureStart API を使用して、イメージのキャプチャを開始します
4) イメージがキャプチャされると、コールバックが呼び出されます
5) stateGet API を使用して、HSYNC と VSTNC の状態を読み取ります
6) 必要に応じてデータを処理します
7) close API を使用して閉じます

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

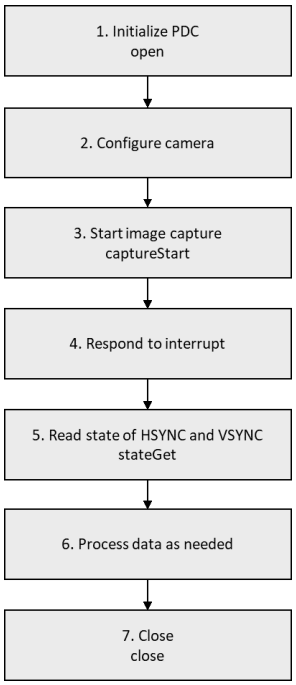


図 267:一般的な PDC HAL モジュールアプリケーションのフロー図

4.2.36 QSPI ドライバー

クワッド SPI（QSPI）HAL モジュールは、マイクロコントローラに接続された QSPI フラッシュデバイスの内容を消去およびプログラミングするためのハイレベル API を提供します。他のモジュールの多くと異なり、QSPI 用のコールバック関数はありません。

4.2.36.1 QSPI HAL モジュールの特長

QSPI HAL モジュールは、クワッド SPI インタフェースを通じてマイクロコントローラに接続された QSPI フラッシュデバイスの内容を消去およびプログラミングできる QSPI 周辺機能を初期化するために使用されます。主な特長は次のとおりです。

- 直接通信モードを使用したクワッド SPI フラッシュデバイスへのアクセス

- QSPI フラッシュデバイスのデータのリード
- QSPI フラッシュデバイスのページのプログラミング
- QSPI フラッシュデバイスのセクターの消去
- QSPI フラッシュデバイスへのアクセスを制御するバンクの選択

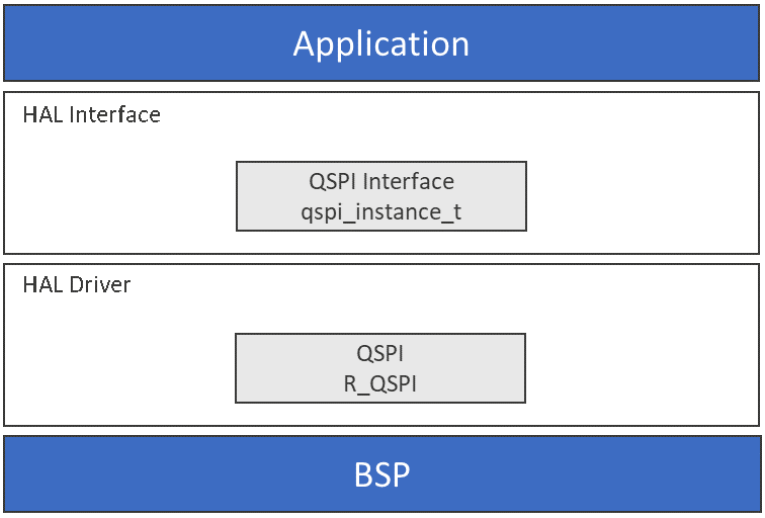


図 268:**QSPI HAL モジュールのブロック図**

4.2.36.2 QSPI HAL モジュールの API の概要

QSPI インタフェースでは、QSPI HAL モジュールを使用したオープン、クローズ、リード、ライト、消去、デバイスバンク選択のための API 関数が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

QSPI HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_qspi0.p_api->open(g_qspi0.p_ctrl, g_qspi0.p_cfg); Open the QSPI HAL module.
close	g_qspi0.p_api->close(g_qspi0.p_ctrl); Close the QSPI HAL module.

Function Name	Example API Call and Description
read	g_qspi0.p_api->read(g_qspi0.p_ctrl, (uint8_t *) QSPI_DEVICE_ADDRESS, readBuffer, BUFFER_LENGTH); Read data from the flash.
pageProgram	g_qspi0.p_api->pageProgram(g_qspi0.p_ctrl, (uint8_t *) QSPI_DEVICE_ADDRESS, writeBuffer, BUFFER_LENGTH); Program a page of data to the flash.
sectorErase	g_qspi0.p_api->sectorErase(g_qspi0.p_ctrl, (uint8_t *) QSPI_DEVICE_ADDRESS); Erase a sector on the flash.
erase	g_qspi0.p_api->erase(g_qspi0.p_ctrl, (uint8_t *) QSPI_DEVICE_ADDRESS, BYTE_COUNT); Erase a block of memory depending on the input argument "byte_count"
statusGet	g_qspi0.p_api->statusGet(g_qspi0.p_ctrl, &in_progress); Get the write or erase status of the flash.
bankSelect	g_qspi0.p_api->bankSelect(0); Select the bank to access.
infoGet	g_qspi0.p_api->infoGet(g_qspi0.p_ctrl, &qspi_info); Provides information about the underlying QSPI flash, as specified in bsp_qspi.c

Function Name	Example API Call and Description
versionGet	g_qspi0.p_api->versionGet(& ssp_version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter is passed.
SSP_ERR_ASSERTION	p_cfg was NULL.
SSP_ERR_NOT_OPEN	Driver is not opened.
SSP_ERR_UNSUPPORTED	Driver not able to query the following information from the flash manufacturer id, memory capacity and memory type.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.36.3 QSPI HAL モジュールの動作の概要

QSPI HAL モジュールは、Synergy デバイスが QSPI シリアルフラッシュデバイスと通信（データのリード、ライト、消去）できるように、QSPI ペリフェラルを初期化するために使用されます。

このドライバーは、ページプログラム（ライト）、リード、消去の 3 つの動作モードをサポートします。

- ページプログラム操作は、単一のデータページをフラッシュデバイスにプログラムします。ページサイズはフラッシュメモリに固有であり、ベンダによって異なる場合があります。フラッシュの一般的なページサイズは、128、256、または 512 バイトです。基になっているフラッシュデバイスでサポートされているページサイズを取得するには、infoGet API を使用します。
- リード動作は、フラッシュからデータを読み取って、ユーザーが提供するバッファに格納します。
- 消去動作は、データのブロックをフラッシュから消去します。基になっているフラッシュデバイスでサポートされている消去サイズを取得するには、infoGet API を使用します。

  注 : すべての消去 / ライト動作の後、次の動作を開始する前に、statusGet API を使用して動作の状態をポーリングすることをお勧めします。これを行わないと、ユーザーデータが壊れる可能性があります。

QSPI HAL モジュールの動作に関する重要な注意事項と制限事項

SSP および BSP ベースのプロジェクト（たとえば SK-S7G2 や DK-S7G2）によってサポートされているボード、および QSPI メモリデバイスがプレインストールされているボードを使用する場合、BSP は初期化を行って、QSPI ペリフェラルを XIP（インプレース実行）が有効な ROM アクセスモードにします。このプロセスにより、標準メモリのようにメモリを読み取ることができるようになります。つまり、QSPI HAL モジュールは、QSPI フラッシュデバイスの消去とプログラミングのときにのみ必要です。

デフォルトのページプログラミングは、1 つのライン（D0）のみを使用します。ページプログラムで複数のラインを使用するには、bsp_qspi.h: QSPI_COMMAND_PAGE_PROGRAM_QUAD,、QSPI_COMMAND_4BYTE_PAGE_PROGRAM_QUAD,、QSPI_COMMAND_PAGE_PROGRAM_DUAL,、または QSPI_COMMAND_4BYTE_PAGE_PROGRAM_DUAL. で以下のマクロの 1 つとしてコマンドを定義します。また、デュアル入力ページプログラムおよびクワッド入力ページプログラムのコマンドのときのみアドレスを D0 で送る必要がある場合は、QSPI_PAGE_PROGRAM_ADDRESS_ONE_LINE を 1 に定義します。これの例は、SSP バージョン 1.5.0 以降の S7G2-DK 用の bsp_qspi.h にあります。

このモジュールガイドに付属するコードでは、QSPI HAL モジュール経由のアクセスと XIP が有効な ROM アクセスモードという両方の動作モードの例が示されています。

一般的な QSPI アプリケーションは、QSPI フラッシュ デバイスでデータをプログラムまたは消去します。このドライバーが開かれていない場合、QSPI フラッシュデバイスの内容は 0x60000000 にマップされ、通常のメモリであるかのように読み取ることができます。

このドライバーは、Micron N25Q256A QSPI フラッシュデバイスでテストされました。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.36.4 アプリケーションへの QSPI HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに QSPI HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

QSPI ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（QSPI ドライバーのデフォルト名は g_qspi0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

QSPI HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_qspi0 QSPI Driver on r_qspi	Threads	New > Driver > Storage > QSPI Driver on r_qspi

次の図に示すように、r_qspi の QSPI ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキ

ストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

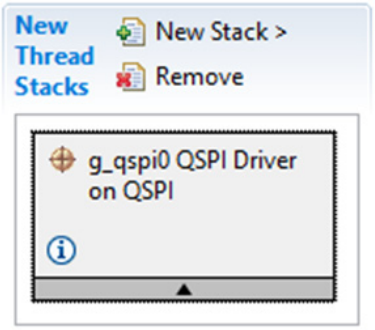


図 269:**QSPI HAL モジュールのスタック**

4.2.36.5 QSPI HAL モジュールの構成

ユーザーは必要な動作に合わせて QSPI HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_qspi での QSPI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Name	g_qspi0	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

QSPI HAL モジュールのピン構成

QSPI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表では [SSP configuration] ウィンドウでのピンの選択方法を示し、その次の表では QSPI ピンの選択例を示します。

r_qspi での QSPI HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
QSPI	Pins	Select Peripherals > Storage:QSPI > QSPI0

注 : この選択シーケンスでは、QSPI0 がドライバーに必要なハードウェアターゲットであることを想定しています。

QSPI HAL モジュールのピン構成設定

Property	Value	Description
Pin Group Selection	- Mixed - A only	Pin group selection.
Operation Mode	- Disabled - Custom - Single or Dual - Quad	Operation mode.
QSPCLK	None, P500	QSPI clock output pin.
QSSL	None, P501	QSPI slave select pin.
QIO0	None, P502	Data 0 input/output.
QIO1	None, P503	Data 1 input/output.
QIO2	None, P504	Data 2 input/output.
QIO3	None, P505	Data 3 input/output.

注 : 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトからのものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.2.36.6 アプリケーションでの QSPI HAL モジュールの使用

アプリケーションで QSPI HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API をコールして、QSPI HAL モジュールを初期化します。
2) read API をコールして、データのブロックを読み取ります。
3) sectorErase API をコールして、データのセクターを消去します。
4) erase API をコールして、データの n バイトを消去します。
   a) infoGet を使用すると、基になっているフラッシュでサポートされている消去サイズを取得できます。
   b) statusGet API を使用すると、sectorErase API にも適用される消去動作の状態をポーリングできます。
5) pageProgram API をコールして、データのページをプログラミングします。
   a) infoGet API を使用して、基になっているフラッシュでサポートされているページサイズを取得します。
   b) statusGet API を使用すると、ライト動作の状態をポーリングできます。
6) close API をコールして、QSPI HAL モジュールを閉じます。

注 : セクターはフラッシュデバイスの標準サイズではなく、サイズはベンダによって異なるので、sectorErase API ではなく erase API を使用することをお勧めします。

これらの一般的な手順を、次の図の一般的な動作フロー図に示します。

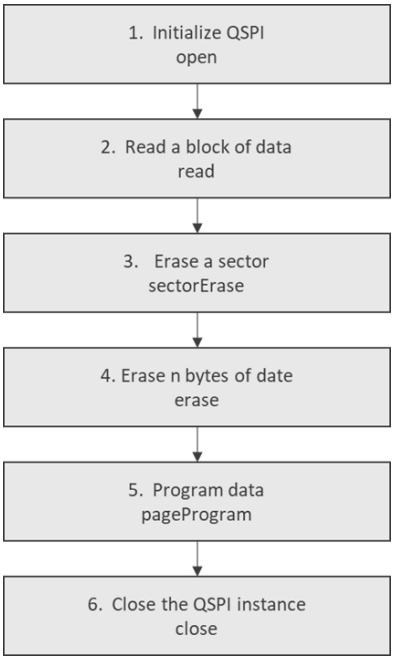


図 270:一般的な QSPI HAL モジュールアプリケーションのフロー図

4.2.37 RTC ドライバー

リアルタイムクロック（RTC）HAL モジュールは、リアルタイムタイミングアプリケーション用のハイレベル API を実装し、Synergy MCU 上のリアルタイムクロックモジュールを使用します。RTC HAL モジュールは、RTC モジュールを構成し、クロック、カレンダー、およびアラーム機能を制御します。コールバックを使用することにより、サポートされる 3 つの割り込みタイプ（アラーム、周期、キャリー）のどれにも応答できます。

4.2.37.1 RTC HAL モジュールの特長

- RTC ペリフェラルの設定。
- RTC の時刻と日付の取得と設定。
- RTC の時刻と日付のアラームの取得と設定。
- RTC の時間カウンタの開始と停止。
- RTC のアラームイベント、周期イベント、キャリーイベントの通知。
- RTC のイベントタイプの有効化と無効化。
- RTC のイベントレートの構成。
- RTC のクロックソースの設定と取得。
- RTC サブクロックのエラー調整。
- RTC ステータスの取得。

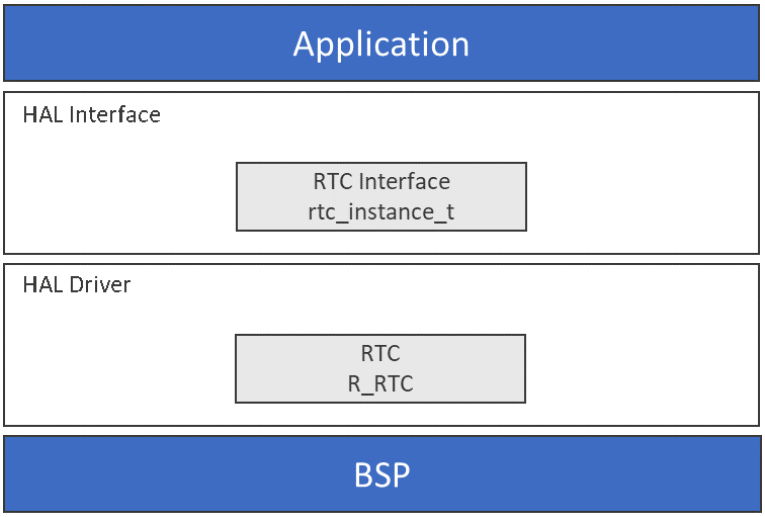


図 271:RTC HAL モジュールのブロック図

4.2.37.2 RTC HAL モジュールの API の概要

RTC HAL モジュールでは、RTC 動作のオープン、クローズ、アラーム設定、開始、停止のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

RTC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_rtc0.p_api->open(g_rtc0.p_ctrl, g_rtc0.p_cfg); Open the RTC HAL.
close	g_rtc0.p_api->close(g_rtc0.p_ctrl); Close the RTC HAL.
configure	g_rtc0.p_api->configure(g_rtc0.p_ctrl, p_extend);
calendarTimeSet	g_rtc0.p_api->calendarTimeSet(g_rtc0.p_ctrl, &start_time_struct_in, true); Set the calendar time.
calendarTimeGet	g_rtc0.p_api->calendarTimeGet(g_rtc0.p_ctrl, &current_time_struct_out); Get the calendar time.
calendarAlarmSet	g_rtc0.p_api->calendarAlarmSet(g_rtc0.p_ctrl, &in_alarm_time_struct_in, true); Set the calendar alarm time.
calendarAlarmGet	g_rtc0.p_api->calendarAlarmGet(g_rtc0.p_ctrl, &get_alarm_time_struct_out); Get the calendar alarm time.
calendarCounterStart	g_rtc0.p_api->calendarCounterStart(g_rtc0.p_ctrl); Start the calendar counter.

Function Name	Example API Call and Description
calendarCounterStop	g_rtc0.p_api->calendarCounterStop(g_rtc0.p_ctrl); Stop the calendar counter.
irqEnable	g_rtc0.p_api->irqEnable(g_rtc0.p_ctrl, CALLBACK); Enable the alarm irq.
irqDisable	g_rtc0.p_api->irqDisable(g_rtc0.p_ctrl, CALLBACK); Disable the alarm irq.
periodicIrqRateSet	g_rtc0.p_api->periodicIrqRateSet(g_rtc0.p_ctrl, Rate); Set the periodic irq rate.
infoGet	g_rtc0.p_api->infoGet(g_rtc0.p_ctrl, clk_src); Return the currently configure clock source for the RTC.
errorAdjustmentModeSet	g_rtc0.p_api->errorAdjustmentModeSet(g_rtc0.p_ctrl, p_error_adjust_mode); Set time error adjustment mode.
errorAdjustmentSet	g_rtc0.p_api->errorAdjustmentSet(g_rtc0.p_ctrl, p_error_adjust_config); Set time error adjustment.
versionGet	g_rtc0.p_api->versionGet(&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function executed successfully
SSP_ERR_ASSERTION	API dependent error
SSP_ERR_INVALID MODE	Invalid mode.
SSP_ERR_INVALID_PTR	Invalid parameter.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.37.3 RTC HAL モジュールの動作の概要

RTC HAL モジュールは、Synergy MCU のリアルタイムクロックモジュールの操作を制御します。一般的な RTC アプリケーションは、ユーザーによって指示されたシステム設定に基づき、リアルタイム クロック コントローラーを定期的に設定します。共通の動作としては、時刻の設定、アラームの設定、周期割り込みの構成、動作の開始または停止などがあります。RTC アプリケーションは通常、RTC HAL モジュールのコールと、ISR ハンドラからのオプションのコールバックで構成されます。

- RTC HAL モジュールは、2 つのメインクロックソースを使用できます
  - 低速オンチップ発振器（LOCO）は低消費電力ですが、精度は高くありません
  - サブクロック発振器は消費電力が多く、精度が高く、高コストです（外部水晶発振器が必要）
- RTC HAL モジュールは 3 種類の割り込みタイプをサポートしています
  - アラーム割り込みは、年、月、日、曜日、時、分、秒の任意の組み合わせに一致した場合に生成されます
  - 周期割り込みは、2、1、1/2、1/4、1/8、1/16、1/32、1/64、1/128、または 1/256 秒ごとに生成されます
  - キャリー割り込みは、第 2 のカウンタへのキャリーが発生したとき、または 64Hz カウンタへのリードアクセス中に R64CNT カウンタへのキャリーが発生したときに、生成されます

ユーザー定義のコールバック関数を（open API のコールで）登録することができ、このコールバック関数はサポートされているいずれかの割り込みタイプの割り込みサービスルーチン（ISR）からコールされます。呼び出されたコールバック関数には、ユーザー定義のコンテキストポインタと発生した割り込みの種類を示す情報が格納された構造体（rtc_callback_args_t）へのポインタが渡されます。

注 : ロールオーバー時に calendarTimeGet API から誤った時刻が返されるのを避けるために、キャリー割り込み優先順位を設定する必要があります。

RTC HAL モジュールの動作に関する重要な注意事項と制限事項

他の RTC モジュール API を呼び出す前に、RTC HAL モジュールを開く必要があります。open のコールに渡す構成構造体では、クロックソース、ISR ハンドラからのユーザーコールバックの名前、コールバックに対するユーザー指定のコンテキストを指定します。設定構造体は、手動で定義するか、設定手順の中でのユーザー入力に基づいて ISDE によって生成します。

ドライバーの関数にアクセスするには、HAL レイヤーを直接呼び出すか、RTC インターフェイス構造体を使用します。このインタフェース構造体の名前は、モジュールの構成で入力した名前の設定に基づきます。たとえば、名前が g_rtc, の場合、インタフェース構造体の名前は g_rtc_api.. になります。

- calendarTimeGet API は内部で処理のためにキャリー割り込みを使用するため、グローバルに割り込みを無効化した状態でこの API をコールすることはできません。これを行うと、API から誤った時刻が返される可能性があります。
- このモジュールでは、次の機能はサポートされていません。
  - バイナリカウントモード
  - バイナリアラームの設定と取得
  - バイナリ時刻の取得と設定
  - LOCO クロックエラー訂正
  - 1-Hz/64-Hz クロック出力
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.37.4 アプリケーションへの RTC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに RTC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

RTC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(RTC ドライバーのデフォルト名は g_rtc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

RTC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_rtc0 RTC HAL on r_rtc	Threads	New Stack> Driver> Timers> RTC HAL on r_rtc

次の図に示すように、r_rtc の RTC ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

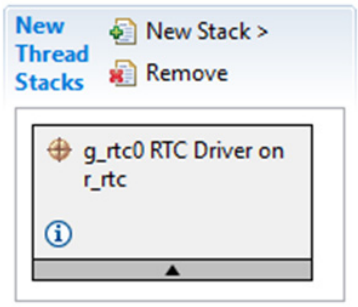


図 272:RTC HAL モジュールのスタック

4.2.37.5 RTC HAL モジュールの構成

ユーザーは必要な動作に合わせて RTC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_rtc での RTC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.
Name	g_rtc0	The name to be used for the RTC module control block instance. This name is also used as the prefix of the other variable instances. See the example code below

ISDE Property	Value	Description
Clock Source	LOCO, Sub-clock Default: LOCO	Clock source for the RTC block.
Configure RTC hardware in open() call	Yes, No Default: Yes	If enabled, the RTC registers and clock source will be initialized in the open() call. If disabled, the user call must call the configure() api to initialize the hardware.
Error Adjustment Value	0	Warning: Deprecated configuration field. Must be 0.
Error Adjustment Type	None	Warning: Deprecated configuration field. Must be 0.
Callback	NULL	The name of the ISR that is called when one of the three interrupts fire. The argument passed into this ISR has an indication of which interrupt caused it to be called. See the example code below.
Alarm Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Alarm interrupt priority selection
Period Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Period interrupt priority selection
Carry Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Default: Priority 12	Carry interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

RTC HAL モジュールのクロック構成

RTC HAL モジュールは、次のクロックソースを使用できます。

- LOCO (低速オンチップ オシレーター)
  - 低い電力消費
  - 低い精度
- サブクロック オシレーター
  - 高い電力消費
  - 高い精度
  - 高コスト (水晶が必要)

LOCO が構成の間のデフォルトの選択です。

RTC HAL モジュールのピン構成

現在、RTC は出力をサポートしていないので、出力ピンの選択はありません。

4.2.37.6 アプリケーションでの RTC HAL モジュールの使用

一般的な使用方法

一般的な RTC アプリケーションは、ユーザーによって指示されたシステム設定に基づき、リアルタイム クロック コントローラーを定期的に設定します。たとえば、時刻の設定、アラームの設定、周期割り込みの構成などです。RTC アプリケーションは、RTC モジュールのコールと、オプションのコールバックで構成されます。

他の API を呼び出す前に、RTC モジュールを開く必要があります。open のコールに渡す構成構造体では、クロック ソース、コールバックの名前、ハンドラのユーザー指定のコンテキストを指定します。設定構造体は、手動で定義するか、設定手順の中でのユーザー入力に基づいて ISDE によって生成します。モジュールの機能には、RTC インタフェース構造体を使用してアクセスできます。このインタフェース構造体の名前は、モジュールの構成で入力した名前の設定に基づきます。

リセット後のドリフト問題の回避

RTC の時刻が MCU リセットの前後でドリフトするのを避けるために、アプリケーションは以下の手順を実行する必要があります。

以下の変更をモジュールの構成設定に加えます。

1) ISDE コンフィギュレータで CGC スタック要素の [Configure Subclock Drive On Reset] オプションを無効化します。
2) ISDE コンフィギュレータで RTC スタック要素の [Configure RTC hardware in open() call] オプションを無効化します。

アプリケーションコードで以下のコールを実行します。

1) open API を通常どおりにコールします。
2) RTC configure API は、コールドスタートの場合にのみコールします。これにより、RTC はコールドスタート状態のときにのみ初期化されます。

この 2 つの手順は、アプリケーションで以下の初期化シーケンスを使用することで実行できます。

```
g_rtc.p_api->open(g_rtc.p_ctrl,g_rtc.p_cfg);

g_rtc.p_api->infoGet(g_rtc.p_ctrl,&info1);
```

```
/* initialize RTC if its status is stopped state, that is, on cold start */

if(RTC_STATUS_STOPPED == info1.status)

{

       /* if the RTC clock source is sub-clock, stop it so that the sub-clock dr
ive capacity is set correctly in the configure API */

       g_cgc.p_api->clockStop(CGC_CLOCK_SUBCLOCK);

       g_rtc.p_api->configure(g_rtc.p_ctrl, NULL);

       g_rtc.p_api->calendarTimeSet(g_rtc.p_ctrl,&rt_time,true);

             }
```

コールドスタート状態を判断するもう 1 つの（ステータスを使用しない）方法は、リセットステータスレジスタ（RSTSRx）から情報を取得することです。

日付と時刻の検証

calendarTimeSet および calendarAlarmSet API に関して日付と時刻の検証が必要な場合は、ISDE コンフィギュレータで [Parameter Checking] 設定を有効にする必要があります。[Parameter Checking] を有効にすると、指定された日付に対して [day of the week] フィールドがドライバーによって自動的に計算されて更新されます。calendarAlarmSet API を使用すると、対応する一致フラグが設定されているフィールドのみがレジスタに書き込まれ、それ以外のレジスタフィールドはデフォルト値にリセットされます。

サブクロックのエラー調整

errorAdjustmentModeSet および errorAdjustmentSet API を使用すると、RTC サブクロックソースのエラーを修正することができます。これらの API は、RTC が構成され、時刻が設定された後でのみ使用できます。

このエラー調整は、RTC が再構成されて時刻が設定されるたびにリセットされます。

RTC HAL モジュールの一般的な用途は 2 つあります。1 つ目は、RTC を使用することによりアプリケーションの必要に合わせて単に現在時刻を提供するものです。2 つ目は、RTC HAL モジュールの周期的な割り込み機能を使用して、定期的に処理を開始するというものです。以下に両方の例を示します。

注 :ISDE コンフィギュレータの RTC スタックにある構成プロパティ [Configure RTC hardware in open() call] は、open API の動作を制御します。有効にした場合、RTC ペリフェラルは open API で構成されます。無効にした場合、RTC の使用前に configure API を使用して構成することはアプリケーションの責任になります。

タイミングアプリケーションで RTC HAL モジュールを使用する一般的な手順は、次のとおりです。

1) open API を使用して、RTC を初期化します
2) calendarTimeSet API を使用して、時刻を設定します
3) calendarAlarmSet API を使用して、アラームを設定します（必要な場合）
4) calendarCounterStart API を使用して、カレンダーカウンタを開始します
5) calendarTimeGet API を使用して、現在の日時を取得します（必要な場合）

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

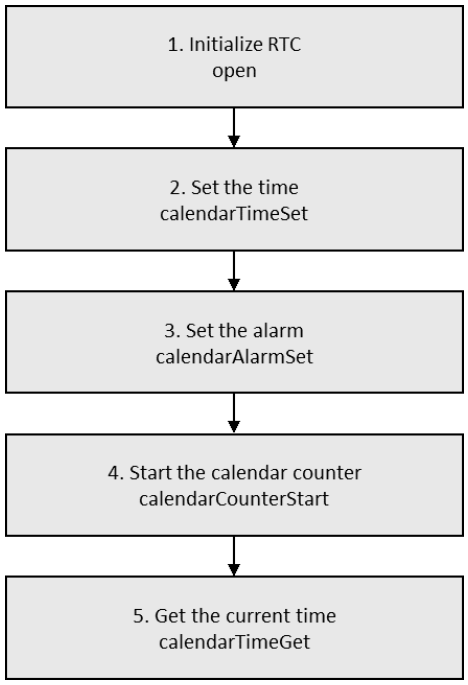


図 273:一般的な RTC HAL モジュールタイミング使用アプリケーションのフロー図

アプリケーションで RTC の周期 IRQ を使用する一般的な手順は次のとおりです。

1) open API を使用して、RTC を初期化します。I
2) periodicIrqRateSet API を使用して、周期 IRQ のレートを設定します
3) calendarCounterStart API を使用して、カレンダーカウンタを開始します
4) irqEnable API を使用して、割り込みを有効にします

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

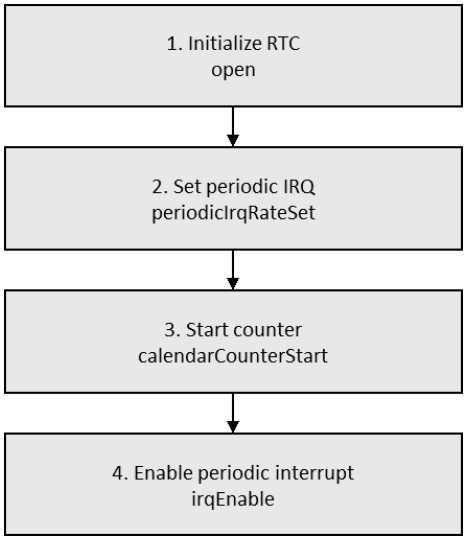


図 274:一般的な RTC HAL モジュールアプリケーションのフロー図

4.2.38 SCE 暗号化ドライバー

セキュア暗号化エンジン（SCE）HAL モジュールは、乱数生成、ダイジェスト計算（ハッシュ）、データ暗号化および復号化、デジタル署名および検証、キー生成（RSA、AES、および ECC アルゴリズムを使用）、キーインストール（RSA、AES、および ECC キー用）のためのハイレベル API 関数を提供します。SCE は専用ハードウェアブロックであり、SCE が提供する機能はサポートされている MCU によって異なります。

4.2.38.1 SCE HAL モジュールの特長

SCE HAL モジュールによって構成される暗号モジュールを使用すると、以下の暗号プリミティブを備えたセキュリティ用の暗号プロトコルを作成できます。

- 乱数生成
- AES、Triple DES（3DES）、または ARC4 アルゴリズムを使用したデータ暗号化および復号化
- ECC、RSA、または DSA アルゴリズムを使用した署名の生成と検証
- HASH アルゴリズム MD5、SHA1、SHA224、または SHA256 を使用したメッセージダイジェストの計算
- キー生成 - AES ラップキー、RSA プレーンテキストおよびラップキー、ECC プレーンテキストおよびラップキー
- 暗号化されたユーザーキーの Synergy プラットフォームへのインストール

SSP のコンテキストでは、「キーのラップ」と「キーのインストール」という用語は次のように定義されます。

キーのラップ：Synergy プラットフォーム上で対称キーまたは非対称キーのペアを生成する API で、秘密キーがラップキー（暗号化キー）になります。

キーのインストール：ユーザーが生成した PC（Synergy プラットフォーム以外のシステム）上の秘密キーが Synergy プラットフォームにインストールされ（保管はなし）、ラップされた秘密キーがユーザーに返されます。

ラップキーには以下のメリットがあります。

- ラップキーは、それが生成された Synergy プラットフォーム（MCU）でのみ使用可能です。
- 別の Synergy プラットフォーム（MCU）に移動することはできません。
- ラップキーから元のキーを復元することはできません。

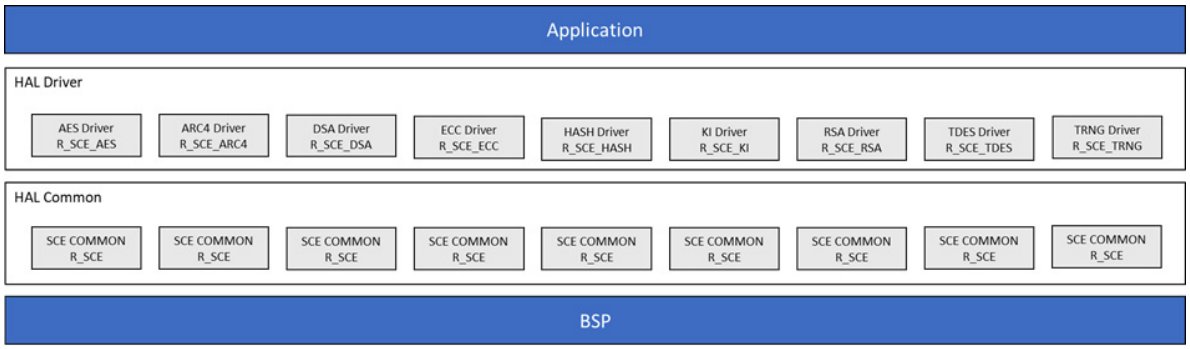


図 275:SCE HAL モジュールのブロック図

注 :KI は Key Installation の略語です。

4.2.38.2 SCE HAL モジュールの API の概要

SCE インタフェースは、SCE HAL モジュール用の共通 API を提供します。SCE インタフェースは、選択されたモジュール（AES、ARC4、RSA、DSA、HASH、TDES、TRNG）に応じて、複数の動作をサポートします。

AES インタフェースでは、AES アルゴリズムの使用に関するオープン、クローズ、ラップキーの生成、データの暗号化、データの復号化のための API が定義されています。128 ビット、192 ビット、256 ビットのキーと、ECB、CBC、CTR、GCM、または XTS のチェーンモードオプションを使用します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。戻りステータスの値については、『SSP ユーザーズマニュアル』の「SCE API リファレンス」セクションを参照してください。

SCE 共通インスタンスの API の要約

Function Name	Example API Call and Description
open	g_sce.p_api->open(g_sce.p_ctrl, g_sce.p_cfg); SCE Common module open function. Must be called before performing any other crypto operations.

Function Name	Example API Call and Description
close	g_sce.p_api->close(g_sce.p_ctrl); Close the SCE Common module.
interfaceGet	g_sce.p_api->interfaceGet(g_sce_aes.p_ctrl , p_interface_info, p_interface); Get the interface structure for the interface info provided.
statusGet	g_sce.p_api->statusGet (g_sce.p_ctrl, p_status); Get status of SCE initialization.
versionGet	g_sce.p_api->versionGet(&version); Gets the module code and API version and stores it in provided version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

AES HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_sce_aes.p_api->open(g_sce_aes.p_ctrl, g_sce_aes.p_cfg); AES module open function. Must be called before performing any encrypt/decrypt operations.
createKey	g_sce_aes.p_api->createKey(g_sce_aes.p_ctrl, num_words, p_key); Generate an AES key for encrypt/decrypt operations.

Function Name	Example API Call and Description
encrypt	g_sce_aes.p_api->encrypt(g_sce_aes.p_ctrl , p_key, p_vi, num_words, p_source, p_dest); AES encryption using the chaining mode and padding mode specified in the open() function call.
addAdditionalAuthenticationData	g_sce_aes.p_api->addAdditionalAuthenticationData (g_sce_aes.p_ctrl, p_key, p_vi, num_words, p_source); Add additional authentication data (called before starting an encryption or decryption operation).
encryptFinal	g_sce_aes.p_api->encryptFinal(g_sce_aes.p_ctrl, p_key, p_iv, input_num_words, p_source, output_num_words, p_dest); AES final encryption using the chaining mode and padding mode specified in the open() function call.
decrypt	g_sce_aes.p_api->decrypt(g_sce_aes.p_ctrl , p_key, p_iv, num_words, p_source, p_dest); AES decryption.
setGcmTag	g_sce_aes.p_api-> setGcmTag(g_sce_aes.p_ctrl,num_words, p_source); Set authentication tag data.
getGcmTag	g_sce_aes.p_api->getGcmTag(g_sce_aes.p_ctrl,num_words, p_dest); Get authentication tag data.

Function Name	Example API Call and Description
zeroPaddingEncrypt	g_sce_aes.p_api->zeroPaddingEncrypt(g_sce_aes.p_ctrl, p_key, p_iv, num_bytes, p_source, p_dest) AES zero padding encryption using the chaining mode and padding mode specified. Implementation for GCM mode only API usage - 1. To provide any Add Authentication Data (AAD): set p_dest = NULL 2. Encryption: set p_source to input data and p_dest will return encrypted data 3. Get/Compute Tag: set p_source = NULL
zeroPaddingDecrypt	g_sce_aes.p_api->zeroPaddingDecrypt(g_sce_aes.p_ctrl, p_key, p_iv, num_words, p_source, p_dest); AES zero padding decryption using the chaining mode and padding mode specified. Implementation for GCM mode only API usage - 1. Set expected tag value using the setGcmTag() function 2. To provide any Add Authentication Data (AAD), invoke this API using p_dest = NULL 3. Decryption: set p_source to input encrypted data, decrypted data will be returned in p_dest 4. To verify the tag, invoke this API using p_source = NULL and p_dest = NULL, the return value indicates authentication tag verification status.
versionGet	g_sce_aes.p_api->versionGet(&version); Gets the module code and API version and stores it in provided version pointer.
close	g_sce_aes.p_api->close(g_sce_aes.p_ctrl); Close the AES module.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ARC4 インタフェースでは、オープン、クローズ、キーの設定、データの処理のための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。

ARC4 HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_sce_arc4.p_api->open(g_sce_arc4.p_ctrl, g_sce_trng.p_cfg); Open the ARC4 module.
keySet	g_sce_arc4.p_api->keySet(g_sce_arc4.p_ctrl, &rngbuf, nbytes); Set the key to be used by the ARC4 module.
arc4Process	g_sce_arc4.p_api->arc4Process(g_sce_arc4.p_ctrl, nbytes, &source, &destination); Encrypt or decrypt data using the ARC4 module.
close	g_sce_arc4.p_api->close(g_sce_arc4.p_ctrl); Close the ARC4 module.
versionGet	g_sce_arc4.p_api->versionGet (&version); Retrieve the version using the provided version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

DSA インタフェースでは、オープン、クローズ、デジタル署名、検証のための API が定義されています。使用できるオプションは、1024 ビット公開キーと 160 ビット秘密キー、2048 ビット公開キーと 224 ビット秘密キー、2048

ビット公開キーと 256 ビット秘密キーです。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。

DSA HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_sce_dsa.p_api->open(g_sce_dsa.p_ctrl, g_sce_dsa.p_cfg); DSA module open function. Must be called before performing any sign/verify operations.
verify	g_sce_dsa.p_api->verify(p_key, p_domain, num_words, p_signature, p_paddedHash); DSA signature verification using given DSA public key. This function is deprecated. The function hashVerify should be used instead.
hashVerify	g_sce_dsa.p_api->hashVerify(g_sce_dsa.p_ctrl, p_key, p_domain, num_words, p_signature, p_paddedHash); DSA signature verification using given DSA public key.
sign	g_sce_dsa.p_api->sign(p_key, p_domain, num_words, p_padded_hash, p_dest); DSA Signature generation using DSA private key. This function is deprecated. The function hashSign should be used instead.
hashSign	g_sce_dsa.p_api->hashSign(g_sce_rsa.p_ctrl, p_key, p_domain, num_words, p_padded_hash, p_dest); DSA Signature generation using DSA private key.
close	g_sce_dsa.p_api->close(g_sce_dsa.p_ctrl); Close the DSA module.

Function Name	Example API Call and Description
versionGet	g_sce_dsa.p_api->versionGet(p_version); Gets version and stores it in provided pointer p_version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ECC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_sce_ecc.p_api->open(g_sce_ecc.p_ctrl, g_sce_ecc.p_cfg); Open the ECC driver. This API must be called before performing any ECC operations.
close	g_sce_ecc.p_api->close(g_sce_ecc.p_ctrl); Close the ECC module.
scalarMultiplication	g_sce_ecc.p_api-> scalarMultiplication(g_sce_ecc.p_ctrl, p_domain, p_k, p_p, p_r); This API calculates R=kP.
keyCreate	g_sce_ecc.p_api->keyCreate(g_sce_ecc.p_trl, p_domain, p_generator_point, p_key_private, p_key_public); This API generates key pair for ECC.

Function Name	Example API Call and Description
sign	g_sce_ecc.p_api->sign(g_sce_ecc.p_ctrl, p_domain, p_generator_point, p_key_private, msg_digest, signature_r, signature_s); This API generates signature of ECDSA.
verify	g_sce_ecc.p_api->verify(g_sce_ecc.p_ctrl, p_domian, p_generator_point, p_key_public, msg_digest, signature_r, signature_s); This is a procedure for signature verification of ECDSA.
versionGet	g_sce_ecc.p_api->versionGet(&version); Gets version and stores it in provided version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

HASH インタフェースでは、特定のデータセットのハッシュ値を計算するための API が定義されています。使用できるオプションは、SHA1 アルゴリズムと SHA256 アルゴリズムです。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。

HASH HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_sce_hash.p_api->open(g_sce_hash.p_ctrl, g_sce_hash.p_cfg); HASH module open function. Must be called before performing any sign/verify operations.

Function Name	Example API Call and Description
updateHash	g_sce_hash.p_api->updateHash(p_source, num_words, p_dest); Update hash for the num_words words from source buffer p_source. This function is deprecated. The function hashUpdate should be used instead.
hashUpdate	g_sce_hash.p_api->hashUpdate(g_sce_hash.p_ctrl, p_source, num_words, p_dest); Update hash for the num_words words from source buffer p_source.
close	g_sce_hash.p_api->close(g_sce_hash.p_ctrl); HASH module close function.
versionGet	g_sce_hash.p_api->versionGet(p_version); Gets version and stores it in provided pointer p_version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

Key Installation HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_sce_key_installation.p_api-> open(g_sce_key_installation.p_ctrl, p_cfg); Open the Crypto Key Installation framework for subsequent call/Key installation.

Function Name	Example API Call and Description
close	g_sce_key_installation.p_api->close(g_sce_key_installation.p_ctrl); Close the Crypto Key Installation framework.
keyInstall2	g_sce_key_installation.p_api-> keyInstall2 (g_sce_key_installation.p_ctrl, p_user_key_input, p_user_key_rsa_modulus, p_install_key_input, p_key_data_out); Install a key version 2. This function takes the RSA modulus of the user's RSA private key as one of the input * Parameters to return the RSA wrapped key in a format that is compatible with other Crypto APIs. * For all other key types the functionality remains the same as the earlier keyInstall API.
versionGet	g_sce_key_installation.p_api-> versionGet(&version); Get version of the Crypto Key Installation framework and stores it in the provided version pointer.
keyInstall	g_sce_key_installation.p_api-> keyInstanll(g_sce_key_installation.p_ctrl, p_user_key_input, p_install_key_input, p_key_data_out); Install a key. **L :NOTE:** * This API is deprecated. keyInstall2() must be used instead. * For RSA key installation this function returns a wrapped key that is not compatible with other *Crypto* APIs. * The standard format RSA wrapped key must have the modulus appended to the wrapped private exponent.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

RSA インタフェースでは、オープン、クローズ、RSA アルゴリズムを使用したデータの暗号化と復号化、デジタル署名、アルゴリズムの検証のための API が定義されています。RSA インタフェースでは、1024 ビットまたは 2048 ビットのキーが使用されます。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。

RSA HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_sce_rsa.p_api->open(g_sce_rsa.p_ctrl, g_sce_rsa.p_cfg); RSA module open function. Must be called before performing any encrypt/decrypt or sign/verify operations.
encrypt	g_sce_rsa.p_api->encrypt(g_sce_rsa.p_ctrl, p_key, p_domain, num_words, p_source, p_dest); Encrypt source data from p_source using an RSA public key from p_key and write the results to destination buffer p_dest.
decrypt	g_sce_rsa.p_api->decrypt (g_sce_rsa.p_ctrl, p_key, p_domain, num_words, p_source, p_dest); Decrypt source data from p_source using an RSA private key from p_key and write the results to destination buffer p_dest.
decryptCrt	g_sce_rsa.p_api->decryptCrt(g_sce_rsa.p_ctrl,p_key, p_domain, num_words, p_source, p_dest); Decrypt source data from p_source using an RSA private key from p_key and write the results to destination buffer p_dest. RSA private key data is specified in CRT format.

Function Name	Example API Call and Description
verify	g_sce_rsa.p_api->verify(g_sce_rsa.p_ctrl, p_key, p_domain, num_words, p_signature, p_padded_hash); Verify signature given in buffer p_signature using the RSA public key p_key for the given padded message hash from buffer p_padded_hash.
sign	g_sce_rsa.p_api->sign(g_sce_rsa.p_ctrl, p_key, p_domain, num_words, p_padded_hash, p_dest); Generate signature for the given padded hash buffer p_padded_hash using the RSA private key p_key. Write the results to the buffer p_dest.
signCrt	g_sce_rsa.p_api->signCrt(g_sce_rsa.p_ctrl, p_key, p_domain, num_words, p_padded_hash, p_dest); Generate signature for the given padded hash buffer p_padded_hash using the RSA private key p_key. RSA private key p_key is assumed to be in CRT format. Write the results to the buffer p_dest.
close	g_sce_rsa.p_api->close(g_sce_rsa.p_ctrl); Close the RSA module.
keyCreate	g_sce_rsa.p_api->keyCreate(g_sce_rsa.p_ctrl, p_private_key, p_public_key); Generates an RSA key pair.
versionGet	g_sce_rsa.p_api->versionGet(p_version); Gets version and stores it in provided pointer p_version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

TDES インタフェースでは、TDES 標準に従ってデータを暗号化および復号化するための API が定義されています。使用可能なすべての API のリスト、API コールの例、各 API の簡単な説明が、次の表に含まれます。

TDES HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_sce_tdes.p_api->open(g_sce_tdes.p_ctrl, g_sce_tdes.p_cfg); Open the TDES module.
encrypt	g_sce_tdes.p_api->encrypt(g_sce_tdes.p_ctrl, &key, &iv, nwords, &source, &destination); Encrypt the data.
decrypt	g_sce_tdes.p_api->decrypt(g_sce_tdes.p_ctrl, &key, &iv, nwords, &source, &destination); Decrypt the data.
close	g_sce_tdes.p_api->close(g_sce_tdes.p_ctrl); Close the TDES module.
versionGet	g_sce_tdes.p_api->versionGet(p_version); Gets version and stores it in provided pointer p_version.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

TRNG インタフェースでは、乱数ジェネレータを計算するための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。

TRNG HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_sce_trng.p_api->open(g_sce_trng.p_ctrl, g_sce_trng.p_cfg); Open the TRNG driver for reading random data from the hardware TRNG module.
read	g_sce_trng.p_api->read(g_sce_trng.p_ctrl, p_rngbuf, nbytes); Generate nbytes of random number bytes and store them in p_rngbuf buffer.
close	g_sce_trng.p_api->close(g_sce_trng.p_ctrl); Close the TRNG interface driver.
versionGet	g_sce_trng.p_api->versionGet(&version); Gets version and stores it in provided version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.38.3 SCE HAL モジュールの動作の概要

使用できる暗号化関数はターゲット MCU によって異なります。次の表に、各 MCU シリーズで使用できる機能を示します。

Function	S7G2, S5D9, S5D5, S5D3	S3A1, S3A3, S3A7, S3A6	S124, S128, S1JA	Notes
TRNG	Generate and read random number	Generate and read random number	Generate and read random number	Generate and read random number

Function	S7G2, S5D9, S5D5, S5D3	S3A1, S3A3, S3A7, S3A6	S124, S128, S1JA	Notes
AES	Encryption, decryption, Key Generation - wrapped keys	Encryption, decryption, Key Generation - wrapped keys	Encryption, decryption	Symmetric Key Encryption based on AES standard
AES Key Size	128-bit, 192-bit, 256-bit	128-bit, 256-bit	128-bit, 256-bit	
AES Key Type	Plain text/raw key, Wrapped key	Plain text/raw key, Wrapped key	Plain text/raw key	
AES Chaining Modes	ECB, CBC, CTR, GCM, XTS **L :NOTE:** XTS is supported for 128-bit and 256-bit keys only	ECB, CBC, CTR, GCM, XTS	ECB, CBC, CTR	
ARC4	Encryption, decryption	NA	NA	
TDES	Encryption, decryption	NA	NA	
TDES Key Size	192-bit	NA	NA	
TDES Chaining Modes	ECB, CBC, CTR	NA	NA	
RSA	Signature Generation, Signature Verification, Public-key Encryption, Private-key Decryption, Key Generation - plain text and wrapped keys	NA	NA	Supports CRT keys and standard keys for private key operations
RSA Key Size	1024-bit, 2048-bit	NA	NA	

Function	S7G2, S5D9, S5D5, S5D3	S3A1, S3A3, S3A7, S3A6	S124, S128, S1JA	Notes
RSA Key Type	Plain text/raw key, Wrapped key	NA	NA	
DSA	Signature Generation, Signature Verification	NA	NA	
DSA Key Size	(1024, 128)-bit, (2048, 224)-bit, (2048, 256)-bit	NA	NA	
HASH	MD5, SHA1, SHA224, SHA256	NA	NA	Message digest algorithms
ECC	Key Generation – plain text and wrapped keys, Scalar Multiplication, ECDSA-Signature Generation, ECDSA -Signature Verification,	NA	NA	
ECC Key Size	192-bit, 224-bit, 256-bit and 384-bit	NA	NA	
ECC Key Type	Plain Text/Raw Key, Wrapped Key			
Key Installation	AES, ECC, RSA keys (Plain text keys to be prepared as specified in the Appendix)	AES keys (Plain text keys to be prepared as specified in the Appendix)	NA	

R_SCE モジュールの構成設定

SCE のエンディアンは、デフォルトでビッグエンディアンに設定されます。これをリトルエンディアンモードに設定することができます。

エンディアン構成パラメータの使用方法については、動作に関する注意事項を参照してください。

TRNG モジュールの構成設定

乱数生成では、基礎となるハードウェアが、前回生成した乱数とは異なる固有の 16 バイトの乱数を生成する最大試行回数を設定できます。最大試行回数に達した場合、読み取り API は呼び出し側に対しエラーを返します。それ以外の場合は正常終了コードが返され、呼び出し側が供給するデータバッファに生成された乱数が転送されます。

AES モジュールの構成設定

AES モジュールは、ユーザーが指定したキーの長さ、キーの種類（プレーンテキストまたはラップキー）、チェーンモードを使用するように構成できます。

RSA モジュールの構成設定

RSA モジュールは、ユーザーが指定したキーの長さとキーの種類（プレーンテキストまたはラップキー）を使用するように構成できます。

DSA モジュールの構成設定

DSA モジュールは、ユーザーが指定したキーの長さを使用するように構成できます。

HASH モジュールの構成設定

HASH モジュールは､ユーザーが指定した HASH アルゴリズム(対象の MCU に依存)を使用するように構成できます。

TDES モジュールの構成設定

TDES モジュールは、ユーザーが指定するチェーンモードを使用するように構成できます。

Key Installation モジュールの構成設定

Key Installation モジュールを構成することによってユーザーの暗号化キーをインストールできます。

ECC モジュールの構成設定

ECC モジュールは、ユーザーが指定したキーの長さとキーの種類（プレーンテキストまたはラップキー）を使用するように構成できます。

SCE HAL モジュールの動作に関する重要な注意事項と制限事項

- Synergy S7 および S5 デバイスは SCE7 を備えているので、AES、TRNG、RSA、HASH、DSA、ECC、および Key Installation をサポートします。
- Synergy S3 デバイスは SCE5 を備えているので、AES、TRNG、GHASH をサポートします。GHASH は、AES GCM モードの一部としてサポートされます。Key Installation は、AES GCM、ECB、CTR、XTS、CBC チェーンモードの一部としてサポートされます。Synergy S3 デバイスは、MD5、SHA1/SHA256 HASH の機能をサポートしません。
- Synergy S1 デバイスは、AES および TRNG のみをサポートします。
- サポートされていないモジュールがプロジェクトに追加された場合、そのことを示すコンパイラの警告が生成されます。
- すべてのモジュールは versionGet API をサポートし、この API はモジュールが開かれる前であっても呼び出すことができます。
- フレームワークレイヤーの SF CRYPTO モジュールが使用するために R_SCE モジュールの interfaceGet API が提供されています。InterfaceGet API は、暗号化 HAL インタフェースを要求するために使用されます。次の例にこの API の使用方法を示します。

```
crypto_instance_t          * p_crypto;  /* R_SCE  instance */
void * p_interface = NULL;    /* Declare a pointer to hold the output interface
structure object */
crypto_interface_get_param_t     param;
param.hash_type = CRYPTO_TYPE_HASH_256; /* Requesting SHA 256 interface*/
/* It is mandatory for the address of the p_interface pointer be passed to the A
PI */
```

- p_crypto->p_api->interfaceGet(&param, &p_interface);
- すべての crypto API は、入力パラメータが null ポインタまたは無効の場合に、SSP_ERR_ASSERTION を返す可能性があります。すべての API は、sf_crypto_err_t または ssp_err_t に記載されている、uint32_t. 型に収まるエラーコードを返します。
- 暗号化ハードウェアエンジンは、再入可能性をサポートしていません。暗号化ハードウェアエンジンがタスクの実行でビジー状態の場合、新しい要求には状態エラーコード SSP_ERR_CRYPTO_SCE_RESOURCE_CONFLICT. が返されます。

エンディアン構成パラメータの使用方法：

- デフォルトモードはビッグエンディアンモードで、入力および出力パラメータ（keys、payload、IV など）は uint32_t データ型である必要があります。
- リトルエンディアンモードでは、uint8_t/byte の配列を入力および出力パラメータ（keys、payload、IV など）に使用できますが、これらは（uint32_t*）にキャストする必要があります。
- エンディアンの構成は、SCE モジュールの初期化時に設定され、モジュールが閉じられるまで有効です。したがって、すべてのデータをそれに基づいてフォーマットする必要があります。例を以下に示します。
  - データが uint32_t でビッグエンディアンフォーマットの場合は、ビッグエンディアンモードを選択します。

```
uint32_t test_data[5] = {0x84983E44, 0x1C3BD26E, 0xBAAE4AA1, 0xF95129E5,
0xE54670F1};
```

  - データが同じでバイト配列フォーマットの場合は、リトルエンディアンモードを選択します。

```
uint8_t test_data_byte_array[20] =
{0x84, 0x98, 0x3E, 0x44, 0x1C, 0x3B, 0xD2, 0x6E, 0xBA, 0xAE, 0x4A,
0xA1, 0xF9, 0x51, 0x29, 0xE5, 0xE5, 0x46, 0x70, 0xF1};
```

AES キー

AES ラップキーのサイズは次のとおりです。

```
/** Return Wrapped AES secret key size in bytes for a 128-bit AES Key */
#define AES128_WRAPPPED_SECRET_KEY_SIZE_BYTES  (36U)

/** Return Wrapped AES secret key size in bytes for a 192-bit AES Key */
#define AES192_WRAPPPED_SECRET_KEY_SIZE_BYTES  (52U)

/** Return Wrapped AES secret key size in bytes for a 256-bit AES Key */
#define AES256_WRAPPPED_SECRET_KEY_SIZE_BYTES  (52U)

/** Return Wrapped AES-XTS secret key size in bytes for a 128-bit AES XTS Mode K
ey */
#define AES_XTS_128_WRAPPPED_SECRET_KEY_SIZE_BYTES  (52U)

/** Return AES-XTS secret key size in bytes for a 256-bit AES XTS Mode Key */
#define AES_XTS_256_WRAPPPED_SECRET_KEY_SIZE_BYTES  (84U)
```

keyCreate API によって生成される RSA キーのフォーマットは次のとおりです。

注：エンディアンの設定は、SCE の初期化の際の設定と同じです。

RSA キーのフォーマット：

- RSA 公開キーのフォーマット：
  - ワード 0：公開キー指数
  - ワード 1：RSA モジュールの開始
  - （RSA 1024 ビットキーの場合は 128 バイト、RSA 2048 ビットキーの場合は 256 バイト）
- RSA 秘密キーのプレーンテキスト標準フォーマット：
  - ワード 0：秘密キー指数（RSA 1024 ビットキーの場合は 128 バイト、RSA 2048 ビットキーの場合は 256 バイト）
  - RSA 係数が続きます。（RSA 1024 ビットキーの場合は 128 バイト、RSA 2048 ビットキーの場合は 256 バイト）
- RSA 秘密キーのプレーンテキスト CRT フォーマット：
  - コンポーネントはバイト 0 を exponent2 として次の順序で並べられます。
  - exponent2 // 第 2 係数の CRT 指数、正の整数
  - prime2 // 第 2 係数、正の整数
  - exponent1 // 第 1 係数の CRT 指数、正の整数
  - prime1 // 第 1 係数、正の整数
  - coefficient //（第 1）CRT 係数、正の整数

keyCreate API によって生成される RSA ラップキーのフォーマットは次のとおりです。

- RSA 公開キーは常にプレーンテキストです。
  - バイト 0 ～バイト 3: 公開キー指数
  - バイト 4：RSA モジュールの開始
  - （RSA 1024 ビットキーの場合は 128 バイト、RSA 2048 ビットキーの場合は 256 バイト）
- 標準フォーマットの RSA 秘密キー
  - バイト 0：秘密キー指数はラップされます（長さは、RSA 1024 ビットキーの場合は 128 バイト、RSA 2048 ビットキーの場合は 256 バイト）。
  - プレーンテキストの RSA モジュールが続きます（長さは、RSA 1024 ビットキーでは 128 バイト、RSA 2048 ビットキーでは 256 バイト）。
- AES の encrypt() 関数と decrypt() 関数は、データのパディングをサポートしません。これらの関数は、16 バイトの倍数であるデータ長で動作します。（データパディングは、ユーザーアプリケーションで処理する必要があります）。AES GCM モードでは、16 バイトの倍数ではない認証データへのサポートが必要となる場合があります。これをサポートするために、AES GCM モードでのみ zeroPaddingEncrypt() および zeroPaddingDecrypt() 関数 API が提供されています。
  - AES GCM で uint32_t 配列をビッグエンディアンモードで使用しているときに、データのバイト数が 4 の倍数（ワード長）でない場合は、ゼロでパディングする必要があります。
- TDES の encrypt() 関数と decrypt() 関数は、データのパディングをサポートしません。これらの関数は、8 バイトの倍数であるデータ長で動作します。（データパディングは、ユーザーアプリケーションで処理する必要があります）。

HASH モジュール - MD5

- MD5 では、最後のメッセージダイジェストの出力をバイトスワップする必要があります。中間の更新（部分的更新）は、バイトスワップする必要はありません。
- また、hashUpdate API をコールする前に、フォーマットされた最終ブロック内の長さフィールドをビッグエンディアンフォーマットにすることも必要です。

このモジュールの最新の制限事項については、SSP の最新のリリースノートを参照してください。

4.2.38.4 アプリケーションへの SCE HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに SCE HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

暗号化ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

SCE HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sce_aes_0 AES Driver on r_sce_aes	Threads	New Stack> Driver> Crypto> AES Driver on r_sce_aes
g_sce_arc4_0 ARC4 Driver on r_sce_arc4	Threads	New Stack> Driver> Crypto> ARC4 Driver on r_sce_arc4
g_sce_dsa_0 DSA Driver on r_sce_dsa	Threads	New Stack> Driver> Crypto> DSA Driver on r_sce_dsa
g_sce_ecc_0 ECC Driver on r_sce_ecc	Threads	New Stack> Driver> Crypto> ECC Driver on r_sce_ecc
g_sce_hash_0 HASH Driver on r_sce_hash	Threads	New Stack> Driver> Crypto> HASH Driver on r_sce_hash
g_sce_key_initialization_0 Key Initialization Driver on r_sce_key_initialization	Threads	New Stack> Driver> Crypto> Key Initialization Driver on r_sce_key_initialization
g_sce_rsa_0 RSA Driver on r_sce_rsa	Threads	New Stack> Driver> Crypto> RSA Driver on r_sce_rsa

Resource	ISDE Tab	Stacks Selection Sequence
g_sce_tdes TDES Driver on r_sce_tdes	Threads	New Stack> Driver> Crypto> TDES Driver on r_sce_tdes
g_sce_trng TRNG Driver on r_sce_trng	Threads	New Stack> Driver> Crypto> TRNG Driver on r_sce_trng

次の図に示すように、暗号化 HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

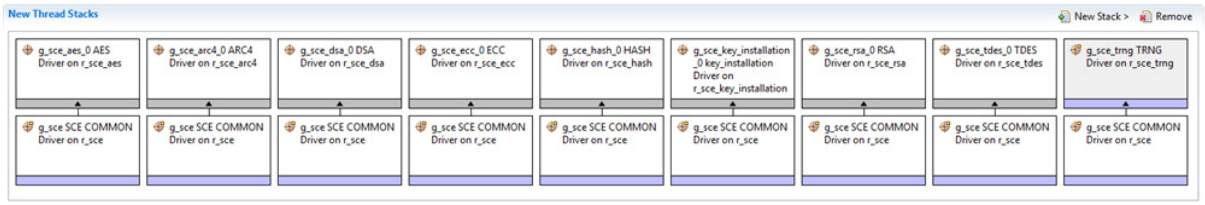


図 276:SCE HAL モジュールのスタック

4.2.38.5 SCE HAL モジュールの構成

ユーザーは必要な動作に合わせて SCE HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_sce_aes 上の AES HAL モジュールの構成設定

ISDE Property	Value	Description
Name	g_sce_aes_0	Module name.

ISDE Property	Value	Description
Key Length	128, 192, 256 Default: 128	Key length used for encryption/decryption operations by this instance of the driver.
Chaining Mode	ECB, CBC, CTR, GCM, XTS Default: CBC	Block cipher chaining mode used for encryption/decryption operations by this instance of the driver.
Key Format	Plain Text Key, Wrapped Key (Not available for S1 MCU series) Default: Plain Text Key	Key format selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sce_arc4 上の ARC4 HAL モジュールの構成設定

ISDE Property	Value	Description
Name (for S7G2, S5D9, S5D5 devices only)	g_sce_arc40	Module name.
Key Length in number of bytes	0	Key length selection.
Key Name, this symbol must be defined as uint8_t array type data in user code	g_arc4_0_key	Key name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sce_dsa 上の DSA HAL モジュールの構成設定

ISDE Property	Value	Description
Name	g_sce_dsa_0	Module name

ISDE Property	Value	Description
Key Length	(1024, 160), (2048, 224), (2048, 256) Default: (2048, 256)	Key length used for signing/verification operations by this instance of the driver.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sce_ecc 上の ECC HAL モジュールの構成設定

ISDE Property	Value	Description
Name (for S7G2, S5D9, S5D5 devices only)	g_sce_ecc0	Module name.
Key Length	192, 256 Default: 256	Key length used for encryption/decryption operations by this instance of the driver.
Key Format	Plain Text Key, Wrapped Key Default: Plain Text Key	Key format selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sce_hash 上の HASH ドライバーの構成設定

ISDE Property	Value	Description
Name (for S7G2, S5D9, S5D5 devices only)	g_sce_hash_0	Module name
Algorithm	SHA1, MD5, SHA224 SHA256 Default: SHA256	Algorithm used for computing the message digest/hash on the message data.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sce_rsa 上の RSA HAL モジュールの構成設定

ISDE Property	Value	Description
Name	g_sce_rsa_0	Module name
Key Length	1024, 2048 Default: 2048	Key length used for signing/verification/encryption/decryption operations by this instance of the driver
Key Format	Plain Text Key, Wrapped Key Default: Plain Text Key	Key format selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sce_tdes 上の TDES HAL モジュールの構成設定

ISDE Property	Value	Description
Name	g_sce_tdes_0	Module name
Chaining Mode	EBC, CBC, CTR Default: CBC	Chaining mode selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sce_trng 上の TRNG HAL モジュールの構成設定

ISDE Property	Value	Description
Name	g_sce_trng	Module name

ISDE Property	Value	Description
Max. Attempts	2	Sets the maximum number of attempts when a newly generated random number differs from the previously generated random number.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sce_key_installation 上の Key Installation の構成設定

ISDE Property	Value	Description
Name (Not Supported for S1 Series MCUs)	g_sce_key_installation_0	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

4.2.38.6 アプリケーションでの SCE HAL モジュールの使用

r_sce HAL モジュール上の SCE ドライバーでは、さまざまな暗号化機能のためにいくつかの API が用意されています。各機能を使用する手順を以下に示しますが、手順が多いためフロー図は提供されません。

一般的なアプリケーションで r_sce HAL モジュール上の SCE ドライバーを使用する手順は次のとおりです。

1) SCE モジュールを使用するには :

   - open API を使用して、SCE 共通ドライバーで SCE および SCE HAL モジュール（R_SCE）を初期化します。これにより、関連付けられている構成パラメータで定義されたとおりにモジュールが初期化されます。
   - 注 : すべての HAL API の入力 / 出力データのエンディアン（リトルエンディアンまたはビッグエンディアン）を構成します。デフォルトでは、ビッグエンディアンモードが構成されます。エンディアンの構成の詳細については、動作に関する上記の注意事項を参照してください。
   - 注 : open 関数は、モジュールが閉じられるまで再度コールすることはできません。

2) AES の関数を使用するには :

   - open API を使用して、選択された AES モジュールを初期化します。これにより、関連付けられている構成パラメータで定義されたとおりにモジュールが初期化されます。
   - 注意 : 使用できる AES キーサイズは、128 ビット、192 ビット、256 ビットです。サポートされているチェーンモードは、ECB、CBC、CTR、GCM、XTS です。
   - createKey API を使用してキーを生成します。

– 注意：AES モジュール内の createKey API は、ラップされたキーを作成します。AES プレーンテキストキーは、TRNG モジュールのサービスを使用して生成できます。

- encrypt API を使用してデータを暗号化します。
- decrypt API を使用してデータを復号します。
- close API を使用して、インタフェースインスタンスを閉じます。

– 注意：GCM 操作には、以下のようなわずかな違いがあります。

– AES GCM 操作に対して指定される IV は、128 ビットにフォーマットされた 96 ビット IV でなければなりません。

– 例：

– 96 ビット IV：e0e00f19fed7ba0136a797f3

– 128 ビットにフォーマットされた 96 ビット IV：e0e00f19fed7ba0136a797f300000001

AES GCM 操作：各操作の後に IV は更新され、その値は後続の各操作で使用する必要があります。
AES GCM 暗号化：

a) AAD（追加の認証データ）はオプションです。使用しない場合、任意のデータを暗号化 / 復号する前に、p_dest = NULL に設定することによってこれを指定する必要があります。

b) 暗号化 : p_source に入力データを設定し、p_dest は暗号化されたデータを返します

c) タグの取得 / 計算：p_source = NULL。

AES GCM の復号：

a) setGcmTag() 関数を使用して、予想されるタグ値を設定します

b) 任意の追加認証データ（AAD）を提供して、p_dest = NULL を使用してこの API を呼び出します。

c) 復号：p_source を入力された暗号化データに設定すると、復号されたデータが p_dest で返されます。

d) タグを検証するには、p_source = NULL および p_dest = NULL を使用してこの API を呼び出します。戻り値は認証タグの検証ステータスを示します。復号されたデータは、このタグが正常に検証された場合にのみ使用されます。

データがブロックサイズの倍数でない場合は、zeroPaddingEncrypt/zeroPaddingDecrypt API を GCM 操作に使用することができます。

3) TDES の関数を使用するには :

- open API を使用して、選択された TDES モジュールを初期化します。これにより、関連付けられている構成パラメータで定義されたとおりにモジュールが初期化されます。

– 注意：TDES チェーンモードは、ECB、CBC または CTR として指定できます。

- encrypt API を使用して、データを暗号化します。
- decrypt API を使用して、データを復号します。
- close API を使用して、インタフェースインスタンスを閉じます。

4) ARC4 の関数を使用するには :

- open API を使用して、選択された ARC4 モジュールを初期化します。これにより、関連付けられている構成パラメータで定義されたとおりにモジュールが初期化されます。
  - 注意：長さ（64 ビットまたは 2048 ビット）と場所で ARC4 キーを指定できます。
- keySet API を使用して、キーを設定します。
- arc4Process API を使用してデータの暗号化または復号を行います。
- close API を使用して、モジュールを閉じます。

5) RSA の関数を使用するには :

- open API を使用して、選択されたモジュールを初期化します。これにより、関連付けられている構成パラメータで定義されたとおりにモジュールが初期化されます。
  - 注意：サポートされているキーの長さは、1024 ビットと 2048 ビットです。
  - 注意：encrypt API、decrypt API、decryptCrt API、sign および signCrt API の場合、データバッファは num_words. で示されます。これは、1024 ビットキーの場合は 32 ワード /128 バイト /1024 ビット、2048 ビットキーの場合は 64 ワード /256 バイト /2048 ビットにする必要があります。
- keyCreate API を使用してキーを生成します。
  - 注意：RSA モジュールの keyCreate API は、この API に対する入力パラメータに基づき RSA プレーンテキストキーまたはラップキーを作成します。
- encrypt API を使用し、RSA 公開キーを使用してデータを暗号化します。
- decrypt API を使用し、RSA 秘密キーを使用してデータを復号します。
- decryptCrt API を使用し、CRT フォーマットの RSA 秘密キーを使用してデータを復号します。
- sign API を使用し、標準フォーマットの RSA 秘密キーを使用して、指定のパディングされたハッシュに対する署名を生成します。
- signCrt API を使用し、CRT フォーマットの RSA 秘密キーを使用して、指定のパディングされたハッシュに対する署名を生成します。
- verify API を使用し、標準フォーマットの RSA 公開キーを使用して、指定のパディングされたハッシュに対する署名を検証します。
- close API を使用して、インタフェースインスタンスを閉じます。

6) DSA の関数を使用するには :

- open API を使用して、選択された DSA モジュールを初期化します。これにより、関連付けられている構成パラメータで定義されたとおりにモジュールが初期化されます。
  - 注意：サポートされているキーの長さは、(1024,160) ビット、(2048,224) ビット、(2048,256) ビットです。
- hashSign API を使用し、DSA 秘密キーを使用して署名を生成します。
- hashVerify API を使用し、DSA 公開キーを使用して署名を検証します。
- close API を使用して、モジュールを閉じます。

7) HASH アルゴリズムを使用するには :

- open API を使用して、選択された HASH モジュールを初期化します。これにより、関連付けられている構成パラメータで定義されたとおりにモジュールが初期化されます。
  - 注意：MD5、SHA1、および SHA256 ハッシュ方式がサポートされています。
- hashUpdate API を使用して、メッセージのダイジェストを計算します。
- close API を使用して、モジュールを閉じます。

8) 真性乱数ジェネレータの関数を使用するには :

- open API を使用して、TRNG モジュールを初期化します。
- read API を使用して、乱数を生成します。
- close API を使用して、インタフェースインスタンスを閉じます。

9) キーインストール API を使用するには：

- open API を使用して、キーインストールモジュールを初期化します。これにより、関連付けられている構成パラメータで定義されたとおりにモジュールが初期化されます。
  - 注意：ユーザーの暗号化されたキーのキーインストールキー構造と、Renesas から提供されたキーインデックス（キーサイズ、キーフォーマット、バッファへのポインタ、およびバッファの長さ）を指定してください。
  - 注意：バッファに対するポインタおよびバッファの長さを指定して、出力キーの構造を指定します。
- keyInstall2 API を使用して、キーをインストールします。
  - 注意：keyInstall API は非推奨です。
- close API を使用して、モジュールを閉じます。

10) ECC の関数を使用するには：

- open API を使用して、選択された ECC モジュールを初期化します。これにより、関連付けられている構成パラメータで定義されたとおりにモジュールが初期化されます。
  - 注意：NIST カーブのドメインパラメータを生成するには、OpenSSL コマンドを使用して下記に示すようにカーブを生成します。
  - ECC P-192：openssl ecparam -name secp192r1 -param_enc explicit -text | more
  - ECC P-224：openssl ecparam -name secp224r1 -param_enc explicit -text | more
  - ECC P-256：openssl ecparam -name secp256r1 -param_enc explicit -text | more
  - ECC P-384：openssl ecparam -name secp384r1 -param_enc explicit -text | more
  - 注意：サポートされているキーのサイズは 192 ビット、224 ビット、256 ビットです。384 ビットもサポート対象です。
  - 注意：scalarMultiplication API の場合、keyCreate、sign、および verify API：
  - データバッファのサイズは r_crypto_data_handle_t の data_length フィールドに示されます。実際のバッファは、r_crypto_data_handle_t の p_data フィールドによってポイントされる必要があります。
- keyCreate API を使用して ECC キーを生成します。
- scalarMultiplication API を使用して ECC ポイントのスカラ値の乗算を行います。

- sign API を使用し、標準フォーマットの ECC 秘密キーを使用して、指定のパディングされたハッシュに対する署名を生成します。
- verify API を使用し、標準フォーマットの ECC 公開キーを使用して、指定のパディングされたハッシュに対する署名を検証します。
- close API を使用して、モジュールを閉じます。

11) close API を使用して、SCE および SCE HAL モジュールを閉じます。

4.2.39 SDADC ドライバー

SDADC HAL モジュールは、アナログ / デジタル変換用のハイレベル API を提供し、Synergy マイクロコントローラハードウェア上で使用できる SDADC24 24 ビットのアナログ / デジタルコンバータペリフェラルをサポートしています。ユーザー定義のコールバックを作成して、新しいサンプルが利用可能になるたびにデータを処理することができます。

4.2.39.1 SDADC HAL モジュールの特長

- 24 ビットシグマデルタ A/D コンバータ
- シングルスキャンまたは連続スキャン動作モード
- シングルエンド入力または差動入力
- 差動入力での最大 32 のゲイン
- 差動入力での構成可能なオーバーサンプリング比

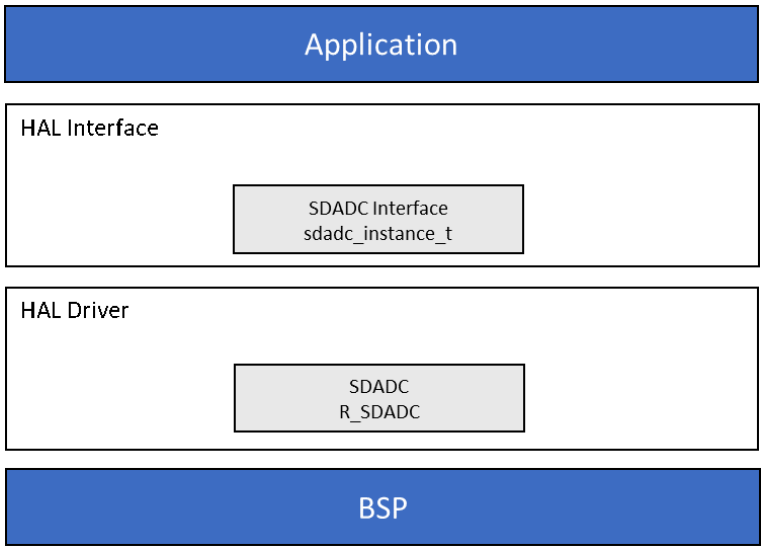


図 277:SDADC HAL モジュールのブロック図

4.2.39.2 SDADC HAL モジュールの API の概要

SDADC HAL モジュールは、オープン、スキャンの構成、スキャンの開始、スキャンの停止を行い、ADC スキャンの変換結果を読み取り、ADC ユニットを閉じるための API 関数を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

SDADC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_adc.p_api->open(g_adc.p_ctrl, g_adc.p_cfg); Initialize ADC Unit; apply power, set the operational mode, trigger sources, interrupt priority, and configurations common to all channels and sensors.
scanCfg	g_adc.p_api->scanCfg(g_adc.p_ctrl, g_adc.p_channel_cfg); Configure the scan including the channels, groups and scan triggers to be used for the unit that was initialized in the open call.
scanStart	g_adc.p_api->scanStart(g_adc.p_ctrl); Start the scan (in case of a software trigger), or enable the hardware trigger.
scanStop	g_adc.p_api->scanStop(g_adc.p_ctrl); Stop the ADC scan (in case of a software trigger), or disable the hardware trigger.
scanStatusGet	g_adc.p_api->scanStatusGet(g_adc.p_ctrl); Check scan status.
read	g_adc.p_api->read(g_adc.p_ctrl, ADC_REG_CHANNEL_13, &adc_data); Read ADC conversion result.

Function Name	Example API Call and Description
read32	g_adc.p_api->read32(g_adc.p_ctrl, ADC_REG_CHANNEL_13, &adc_data); Read ADC conversion result into a 32-bit word.
sampleStateCountSet	g_adc.p_api-> sampleStateCountSet(g_adc.p_ctrl,&adc_sample); Set the sample state count for the specified channel.
calibrate	g_adc.p_api-> calibrate(g_adc.p_ctrl, reg_id, offset); Calibrate ADC or associated PGA (programmable gain amplifier). The driver may require implementation specific arguments to the p_extend input.
offsetSet	g_adc.p_api-> offsetSet(g_adc.p_ctrl, p_extend); Set offset for input PGA configured for differential input.
close	g_adc.p_api->close(g_adc.p_ctrl); Close the specified ADC unit by ending any scan in progress, disabling interrupts, and removing power to the specified A/D unit.
infoGet	g_adc.p_api->infoGet(g_adc.p_ctrl, &adc_info); Return the ADC data register address of the first (lowest number) channel and the total number of bytes to be read for the DTC/DMAC to read the conversion results of all configured channels.

Function Name	Example API Call and Description
versionGet	g_adc.p_api->versionGet(&version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_NOT_OPEN	Unit is not open
SSP_ERR_ASSERTION	The parameter p_ctrl or p_sample is NULL.
SSP_ERR_IN_USE	Peripheral is still running in another mode; perform R_ADC_Close first.
SSP_ERR_INVALID_POINTER	The parameter p_data is NULL.
SSP_ERR_CALIBRATION_FAILED	Calibration failed.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.39.3 SDADC HAL モジュールの動作の概要

SDADC HAL モジュールは、Synergy マイクロコントローラ上の SDADC ペリフェラルを制御します。RTOS 要素を使用しないで SDADC ハードウェアを直接制御し、開発が簡単になる便利な API を提供します。

本書では、「スキャン」という用語は SDADC の AUTOSCAN 機能を示します。この機能は次のように動作します。

1) 変換は、有効になっているチャネルを対象に、チャネル番号の昇順で実施されます。1 つのチャネルに対して必要なすべての変換は、シーケンサが次のチャネルに移動する前に完了されます。
2) 変換は、SDADC オーバーサンプリングクロック周波数 / オーバサンプリング比（チャネルごとに構成）のレート（Hz 単位）で実行されます。SSP は、SDADC オーバーサンプリングクロック周波数のノーマルモードを使用します。
3) チャネルに対して平均化が有効になっている場合、各変換の終了の割り込みが発生する前に、変換数の平均化が行われます。

4) チャネルの変換数が 1 を超える場合、要求された変換数分だけのカウントが実行されます。チャネルに対して平均化が有効になっている場合、平均化された各結果は 1 つの変換としてカウントされます。
5) 要求された変換がすべて完了すると、次の有効化されたチャネルへと進みます。
6) 有効化されたチャネルすべてについてスキャンが完了すると、スキャン終了の割り込みが発生します。このドライバーは、シングルスキャンおよび連続スキャンの動作モードをサポートしています。
   - シングルスキャンモードは、トリガごとに 1 つのスキャンを実施します（scanStart を使用して開始されるハードウェアのトリガおよびソフトウェアの開始）。
   - 連続スキャンモードでは、各スキャンの完了後にスキャンが再度開始されます。SDADC の連続スキャンを開始するには、トリガが 1 つ必要です。

割り込みとコールバック

変換が完了し、ユーザーによってコールバックが指定されている場合、SDADC HAL モジュールは、adc_callback_args_t 引数を指定して、当該ユニットおよびイベント adc_cb_event_t を示すコールバック（p_callback）をコールします。

SDADC ドライバーは以下のコールバックイベントをサポートしています。

- 新しい変換データが利用可能になったことをアプリケーションに通知する ADC_EVENT_CONVERSION_COMPLETE。
- スキャンが終了した時点でアプリケーションに通知する ADC_EVENT_SCAN_COMPLETE。
- 較正プロセスが完了した時点でアプリケーションに通知する ADC_EVENT_CALIBRATION_COMPLETE。

SDADC HAL モジュールの動作に関する重要な注意事項と制限事項

SDADC でのデータ転送のトリガ

SDADC スキャンの完了時にデータ転送をトリガするには、データ転送モジュールをプロジェクトに追加し、プロジェクトの [Activation Source] パラメータを [Event ADC0 SCAN END] に設定します。アプリケーションコードで、open API 関数を使用して転送関数を開きます。enable API 関数を使用して、アクティベーションソースイベントの発生後に転送を開始します。infoGet API 関数を使用して、transfer_properties_t 構造を使用する転送関数によって生成された情報を取得します。

ELC を使用した SDADC でのデータ転送のトリガ

SDADC スキャンが完了すると、

イベントリンクコントローラ（ELC）ペリフェラルを使用して、データの転送をトリガできます。そのためには、まず r_sdadc トリガパラメータを ELC ハードウェアイベントに設定します。アプリケーションコードで ELC HAL モジュール linkSet API 関数を使用してペリフェラルとイベントをリンクします。そのためには、elc_peripheral_t 変数を ELC_PERIPHERAL_SDADC0 に設定し、elc_event_t 変数を ELC_EVENT_SDADC0_SCANEND. に設定します。詳細については、『SSP ユーザーズマニュアル』の「ELC モジュールの概要」を参照してください。

SDAD を使用した ELC イベントのトリガ

SDADC ユニットは、elc_peripheral_t にリストされている他のペリフェラルの起動をトリガできます。詳細については、『SSP ユーザーズマニュアル』の「ELC モジュールの概要」を参照してください。

このモジュールは、一部の Synergy MCU でのみ使用できます。現在の SSP リリースのリリースノートを参照して、このモジュールでサポートされている MCU を確認してください。また、使用可能なペリフェラルが MCU ハードウェアマニュアルに示されています。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.39.4 アプリケーションへの SDADC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに SDADC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユー

ザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

SDADC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(SDADC ドライバーのデフォルト名は g_sdadc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

SDADC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sdadc0 SDADC Driver on r_adc	Threads	New Stack> Driver> Analog> SDADC Driver on r_sdadc

次の図に示すように、r_sdadc の SDADC ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

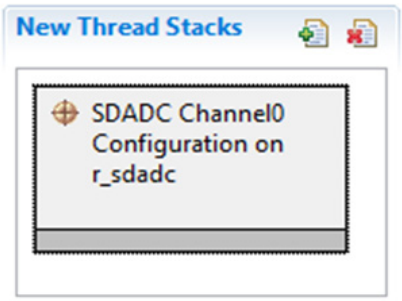


図 278:SDADC HAL モジュールのスタック

4.2.39.5 SDADC HAL モジュールの構成

ユーザーは必要な動作に合わせて SDADC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_sdadc での SDADC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.
Name	g_adc0	Module name.
Mode	Single Scan, Continuous Scan Default: Continuous Scan	In single scan mode, all channels are converted once per start trigger, and conversion stops after all enabled channels are scanned. In continuous scan mode, conversion starts after a start trigger, then continues until stopped in software.
Resolution	16 Bit, 24 Bit Default: 24 Bit	Select 24-bit or 16-bit resolution.
Alignment	Right, Left Default: Right	Select left or right alignment.
Trigger	ELC Hardware Event, Software Default: Software	Select conversion start trigger. Conversion can be started in software, or conversion can be started when a hardware event occurs if the hardware event is linked to the SDADC peripheral using the ELC API.
Vref Source	Internal, External Default: Internal	Vref can be sourced internally and output on the SBIAS pin, or Vref can be input from VREFI.

ISDE Property	Value	Description
Vref Voltage	0.8V, 1.0V, 1.2V, 1.4V, 1.6V, 1.8V, 2.0V, 2.2V, 2.4V Default: 1.0V	Select Vref voltage. If Vref is input externally, the voltage on VREFI must match the voltage selected within 3%.
Internal Calibration During Open()	Enabled, Disabled Default: Enabled	Calibration is required for all channels configured for differential input. Internal calibration is performed automatically during open for these channels unless it is disabled here.
Callback	NULL	Enter the name of the callback function to be called when conversion completes or a scan ends.
Conversion End Interrupt Priority	Priority 0 (highest), Priority 1, Priority 2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Select the interrupt priority for the conversion end interrupt. [Required]
Scan End Interrupt Priority	Priority 0 (highest), Priority 1, Priority 2, Priority 3 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for the scan end interrupt. [Required]
Calibration End Interrupt Priority	Priority 0 (highest), Priority 1, Priority 2, Priority 3 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for the calibration end interrupt. [Required]

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

SDADC HAL モジュールのクロック構成

SDADC HAL モジュールは、SDADCCLK をクロックソースとして使用します。このクロックソースは 4MHz に設定されている必要があります。

SDADC HAL モジュールのピン構成

SDADC HAL モジュールを使用するには、アナログ入力を受信するチャネルのポートピンを、統合ソリューション開発環境のピンコンフィギュレータで入力ピンとして設定する必要があります。次の表では、ISDE [Configuration] ウィンドウでのピンの選択方法を示します。

r_sdadc の SDADC HAL モジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
SDADC	Pins	Select Peripherals > Analog:SDADC>SDADCn where n is the channel number

4.2.39.6 アプリケーションでの SDADC HAL モジュールの使用

アプリケーションで SDADC HAL モジュールを使用するときの一般的な手順は次のとおりです。

1) open を使用して SDADC を初期化します。構成でオープン時の較正が無効化されていない限り、この関数での差動入力に対して構成されたすべてのチャネルで較正が実行されます。
2) 構成でオープン時の較正が無効化されている場合は、calibrate を使用して差動入力に対して構成された各チャネルを較正します。各チャネルの較正後、較正完了の割り込みを待機します。較正の手順の詳細については、R_SDADC_Calibrate を参照してください。
3) scanCfg を使用してアクティブなチャネルを構成します。(オプション)
4) scanStart を使用し、必要なトリガを使用して変換を開始します。
    a) ハードウェアトリガを使用する場合、この呼び出しにより、ADC ユニットをハードウェアトリガでトリガできるようになります。ソフトウェア トリガを使用する場合、この呼び出しにより ADC スキャンが開始されます。
5) 各変換後、スキャンがすべて完了するとコールバックがコールされます。
6) read を使用して変換結果を読み取ります。
7) scanStop をコールして ADC スキャンを停止します。(オプション)
    a) これにより、ADC は外部トリガまたはハードウェアトリガによってトリガされなくなります。また、進行中のソフトウェアトリガスキャンがある場合は強制的に停止されます。
8) 受信したデータをアプリケーションの必要に応じて操作します。
9) close API を使用し、モジュールを閉じてペリフェラルの電源をオフにします。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

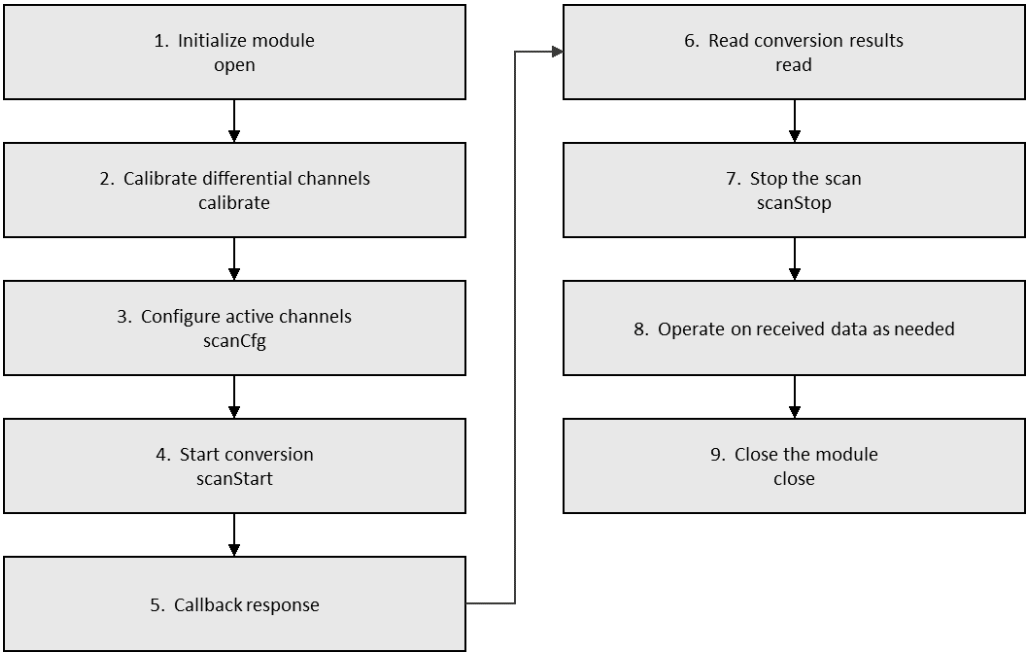


図 279:一般的な SDADC HAL モジュールアプリケーションのフロー図

4.2.40 SD/MMC ドライバーおよび SDIO ドライバー

SDMMC（SD/MMC および SDIO）HAL モジュールは、SDMMC メディアデバイスおよび SDIO カードのリード / ライト、および制御に使用されます。SDMMC モジュールはスタンドアロン SD カード、eMMC またはメディアドライバーとして使用できます。また、このモジュールは、FileX やその他の互換性のあるファイルシステムとともに使用することもできます。SDMMC HAL モジュールは、Synergy MCU 上の SDMMC ペリフェラルを使用します。

4.2.40.1 SDMMC HAL モジュールの特長

- SDSC（SD 標準容量）、SDHC（SD 大容量）、SDXC（SD 拡張容量）、および eMMC（組み込みマルチメディアカード）の各メモリデバイスをサポート
  - SD および eMMC メモリデバイスのリード、ライト、消去をサポート
  - 1 ビット、4 ビット、8 ビットデータバスをサポート（8 ビットバスは eMMC でのみサポート）
  - ハードウェアライトプロテクトの検出をサポート（SD カードのみ）
  - 下位互換モードと高速 SRD モード（eMMC）間を自動的に選択
- SDIO のサポート
  - SDIO シングルレジスタアクセスをサポート（CMD52）
  - SDIO マルチレジスタアクセスをサポート（CMD53）
  - SDIO 割り込みをサポート
  - ホスト（MCU）とデバイスの両方でサポートされる最大クロックレートにクロックを自動的に構成

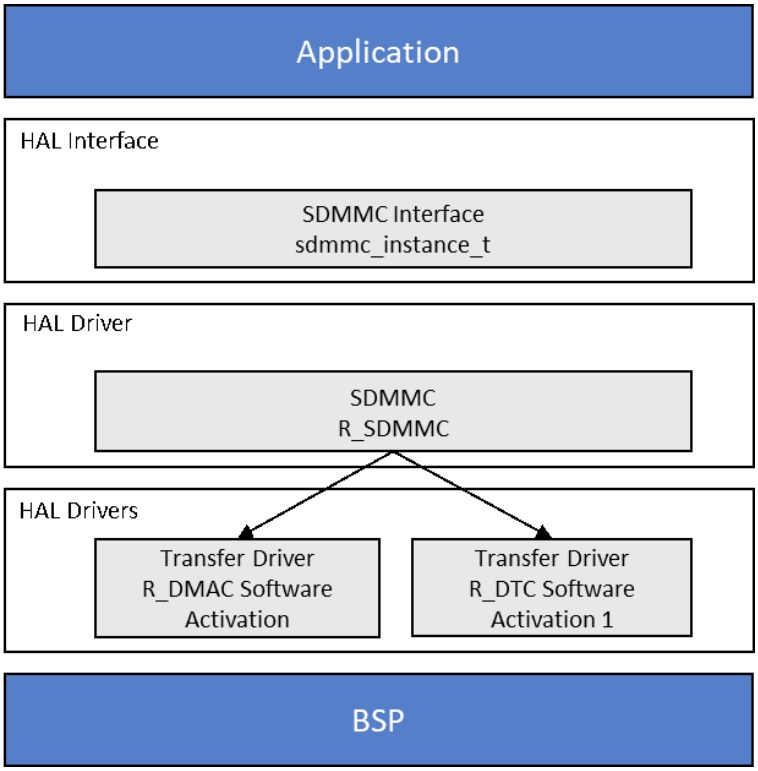


図 280:SDMMC HAL モジュールのブロック図

4.2.40.2 SDMMC HAL モジュールの API の概要

SDMMC HAL モジュールでは、オープン、クローズ、リード、ライトのための API 関数が定義されています。次の表には、使用可能なすべての API 関数のリスト、API 関数コールの例、各 API 関数の簡単な説明が記載されています。ステータス戻り値の表は API 要約表の後にあります。

SDMMC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_sdmmc.p_api->open(g_sdmmc.p_ctrl, g_sdmmc.p_cfg); Open device channel for read/write and control. Initialize driver and hardware on first call.

Function Name	Example API Call and Description
read	g_sdmmc.p_api->read(g_sdmmc.p_ctrl, &destination, start, count); Read data from SD/MMC channel.
write	g_sdmmc.p_api->write(g_sdmmc.p_ctrl, &source, start, count); Write data to SDMMC channel.
control	g_sdmmc.p_api->control(g_sdmmc.p_ctrl, command, &data); Send control commands to the SD/MMC port and receive the status of the SD/MMC port.
close	g_sdmmc.p_api->close(g_sdmmc.p_ctrl); Close open SDMMC channel. Turns off hardware if last channel open.
versionGet	g_sdmmc.p_api->versionGet(&version); Get version of Block Media SD/MMC driver.
readIo	g_sdmmc.p_api->readIo(g_sdmmc.p_ctrl, &data, function, address); Read I/O data from an SDMMC channel.
writeIo	g_sdmmc.p_api->writeIo(g_sdmmc.p_ctrl, &data, function, address, read_after_write); Write I/O data to SDMMC channel.

Function Name	Example API Call and Description
readIoExt	g_sdmmc.p_api->readIoExt(g_sdmmc.p_ctrl, &destination, function, address, count, transfer_mode, address_mode); Read I/O data, extended, from an SDMMC channel.
writeIoExt	g_sdmmc.p_api->writeIoExt(g_sdmmc.p_ctrl, &source, function, address, count, transfer_mode, address_mode); Write I/O data, extended, to SDMMC channel.
IoIntEnable	g_sdmmc.p_api->IoIntEnable(g_sdmmc.p_ctrl, enable); Enables SDIO interrupt for SDMMC channel.
infoGet	g_sdmmc.p_api->infoGet(g_sdmmc.p_ctrl, &info); Get SDMMC channel info.
erase	g_sdmmc.p_api->erase(g_sdmmc.p_ctrl, start, count); Erase SDMMC sectors.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.

Name	Description
SSP_ERR_IN_USE	The channel specified has already been opened. No configurations were changed. Call the associated Close function or use associated Control commands to reconfigure the channel.
SSP_ERR_ASSERTION	The pointer is NULL.
SSP_ERR_WRITE_PROTECTED	SD or MMC card is Write Protected.
SF_INFO_NOT_AVAILABLE	Information not available possibly because card has been removed or is defective.
SSP_ERR_NOT_OPEN	The channel is not opened.
SSP_ERR_HW_LOCKED	The hardware lock has already been acquired.
SSP_ERR_TRANSFER_BUSY	The transfer is in progress.
SSP_ERR_CARD_NOT_READY	The card is not ready yet.
SSP_ERR_NOT_ENABLED	SDIO interrupt enable failed.
SSP_ERR_READ_FAILED	Read operation failed.
SSP_ERR_WRITE_FAILED	Write operation failed.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.40.3 SDMMC HAL モジュールの動作の概要

以下では、SDMMC HAL モジュールを使用した場合に操作できる機能と要件について説明します。

割り込み構成：

DTC で SD/MMC を使用 :

- アクセス割り込み
- DTC 割り込み

DMAC で SD/MMC を使用 :

- アクセス割り込み
- DMAC 割り込み（DMAC インスタンス内）

DTC で SDIO を使用 :

- アクセス割り込み
- SDIO 割り込み

- DTC 割り込み

DMAC で SDIO を使用 :

- アクセス割り込み
- SDIO 割り込み
- DMAC 割り込み（DMAC インスタンス内）

カード割り込みはオプションで、Synergy 構成ツールの [Pins] タブで SDHIn CD ピン（n = チャネル番号）が使用可能になっている MCU パッケージでのみ使用可能です。

ブロックサイズ構成

ブロックサイズ構成は、SDIO カードで使用する場合にのみ提供されます。SDIO カードの場合、ブロックサイズは 1 ～ 512 バイトに構成できます。ブロックサイズは、SD カードと e/MMC メモリデバイスではデフォルトの 512 バイトのままにする必要があります。

カード検出構成

SDMMC HAL ドライバーでカード検出を使用する前に、*カード検出の制限事項*をお読みください。カード検出が使用不能な場合や、アプリケーションに不要な場合は、Synergy 構成ツールの r_sdmmc インスタンスの [Properties] で [Card Detection] を [Not Used] に設定する必要があります。カード検出を有効化するには、Synergy 構成ツールの r_sdmmc インスタンスの [Properties] で [Card Detection] を [CD Pin] に、[Media Type] を [Card] に設定する必要があります。

アクセス割り込み優先順位

データ転送が 4 バイトでアラインされていないか、4 バイトの倍数でない場合、ブロックサイズ（最大 512 バイト）のソフトウェアコピーはアクセス割り込みで実行されます。これにより、ソフトウェアコピーが完了するまで、プライオリティがアクセス割り込み以下の他の割り込みがブロックされます。

一般的なタイミングに関する注意事項

このドライバーの一部の関数はブロックします。open と erase は、動作全体が完了するまでブロックします。read、write、readIo、writeIo、readIoExt、writeIoExt は、単一コマンドレスポンスサイクルにわたってブロックします。マルチスレッドシステムでは、これらのいずれかの関数呼び出し中にこのドライバーがブロックする可能性のある時間よりも短いレスポンスタイムを他のスレッドが必要とする場合、このドライバーをプライオリティの低いスレッドで使用するには注意が必要です。

Open のタイミングに関する注意事項

open API は、デバイスの識別および構成プロセス全体を実行します。これには、1 ビットのバス幅と 400kHz 以下のバス速度でのいくつかのコマンドレスポンスサイクルが関係します。

Read、Write、ReadIoExt、および WriteIoExt のタイミングに関する注意事項

メディアのリードおよびライト（read および write）と拡張リードおよびライト SDIO API（readIoExt および writeIoExt）は、デバイスからレスポンスを受信するまでブロックします。データ転送操作は非ブロックであり、割り込みおよび転送インスタンス（DMAC または DTC）を必要とします。これらの API は、初期動作が正常に開始したことを示す SSP_SUCCESS を返します。アプリケーションは、イベント SDMMC_EVENT_TRANSFER_COMPLETE または SDMMC_EVENT_TRANSFER_ERROR でコールバックを待って、リードまたはライトの完了を示す必要があります。

ReadIo と WriteIo のタイミングに関する注意事項

SDIO readIo および writeIo API は、デバイスからレスポンスを受信するまでブロックします。リードまたはライト動作はこれらの API が戻ると完了します。

消去のタイミングに関する注意事項

erase API は、消去動作が完了するまでブロックします。これは、消去されるセクター数に応じて数秒以上の場合があります。

SDIO 割り込み

多くの SDIO カードはレベル割り込みを使用するため、割り込みは、割り込みがデバイスでクリアされるまでアサート解除されません。SDIO 割り込みが適切にクリアされるようにするには、次のいずれかの方法をお勧めします。

- コールバック関数がイベント SDMMC_EVENT_SDIO. で終了する前に SDIO 割り込みがデバイスでクリアされることを確認します。この方法を選択し、SDIO API を割り込みで使用する必要がある場合は、アクセス割り込みのプライオリティを SDIO 割り込みよりも高くする必要があります。
- IoIntEnable を使用してイベント SDMMC_EVENT_SDIO の発生時のコールバック関数の SDIO 割り込みを無効化します。アプリケーションの他の場所の SDIO 割り込みをクリアしてから、必要に応じて IoIntEnable を使用して SDIO 割り込みを再有効化します。

SDMMC HAL モジュールの動作に関する重要な注意事項と制限事項

SD HALA への準拠

SD の仕様に準拠したホストデバイスを開発する場合、ホストは SD ホスト機器およびペリフェラルの使用許諾契約（SD Host/Ancillary Product License Agreement）（SD HALA）に従う必要があります。

データのアラインメントとサイズ

データ転送は、4 バイトでアラインされる必要があり、可能な場合にはサイズが 4 バイトの倍数である必要があります。この推奨事項は、read()、write()、readIoExt()、writeIoExt() API に適用されます。データ転送が 4 バイトでアラインされ、4 バイトの倍数である場合、r_sdmmc ドライバーはゼロコピーで、DMAC または DTC によるハードウェア加速を十分に活用します。データ転送が 4 バイトでアラインされていないか、4 バイトの倍数でない場合、転送されるブロックごとに特別な CPU 割り込みが必要です（「アクセス割り込み優先順位」を参照）。

カード検出の制限事項

r_sdmmc ドライバーでのカード検出のサポートは制限されています。カード検出は open が完了した後でのみ使用可能で、open() はカードが挿入されていない限り完了できません。したがって、r_sdmmc ドライバーでのカード検出は、カードの取り外しと再挿入の検出にのみ役立ちます。再挿入が検出された後、ドライバーを終了して再開する必要があり、カード検出は再開が完了するまで使用可能になりません。カード検出は、Synergy 構成ツールの [Pins] タブで SDHIn CD ピン（n = チャネル番号）が使用可能になっている MCU パッケージでのみ使用可能です。MCU に SDHIn CD ピンがないか、カード検出がアプリケーションに不要な場合は、Synergy 構成ツールの r_sdmmc インスタンスの [Properties] でカード検出を無効化する必要があります。r_sdmmc ドライバーのカード検出機能の使用の代替方法は、外部 IRQ インスタンスを使用し、アプリケーションレイヤでカード検出を処理することです。カード検出がアプリケーションレイヤーで処理される場合は、カード挿入が検出された後に open を呼び出す必要があり、カード取り外しが検出された後に close を呼び出す必要があります。

このモジュールの最新の制限事項については、SSP の最新のリリースノートを参照してください。

4.2.40.4 アプリケーションへの SDMMC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに SDMMC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

SD/MMC ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（SD/MMC ドライバーのデフォルト名は g_sdmmc0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

SDMMC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sdmmc0 SD/MMC driver on r_sdmmc	Threads	New Stack> Driver> Storage> SD/MMC Driver on r_sdmmc

次の図に示すように、r_sdmmc の SDMMC HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

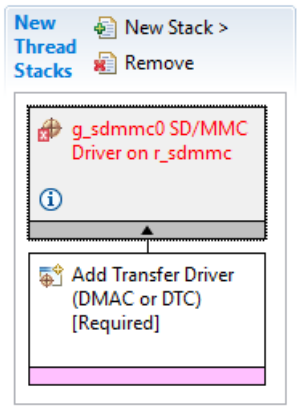


図 281:SDMMC HAL モジュールのスタック

4.2.40.5 SDMMC HAL モジュールの構成

ユーザーは必要な動作に合わせて SDMMC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_sdmmc での SDMMC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.
Name	g_sdmmc0	The name to be used for SDMMC module control block instance. This name is also used as the prefix of the other variable instances.
Channel	0	Select the channel.
Media Type	Embedded, Card Default: Embedded	Media is a card or an embedded device. This allows to firmware to know whether to look for card insertion/removal and write protect pins.
Bus Width	1 Bit, 4 Bits, 8 bits Default: 4 Bits	Data bus with as defined by hardware interface. (8 Bits for eMMC only)
Block Size	512	Select the media block size
Card Detection	Not Used, CD Pin Default: CD Pin	Select the card detection method.
Write Protection	WP Pin, Not Used Default: WP Pin	Select whether or not to use the write protect pin. Select Not Used if the MCU or device does not have a write protect pin.

ISDE Property	Value	Description
Callback	NULL	(Required if not using FileX) Set to name of user callback function. Provides event that caused interrupt: SDMMC_EVENT_CARD_REMOVED, SDMMC_EVENT_CARD_INSERTED, SDMMC_EVENT_ACCESS, SDMMC_EVENT_SDIO, SDMMC_EVENT_TRANSFER_COMPLETE, SDMMC_EVENT_TRANSFER_ERROR
Access Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Default: Priority 12	Access interrupt priority selection
Card Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Card interrupt priority selection
DTC Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	DTC interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

SDMMC HAL モジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_dmac Software Activation の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Name	g_transfer0	Module name
Channel	0	
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection

ISDE Property	Value	Description
Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Software Activation 1 の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection

ISDE Property	Value	Description
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation 1	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

SDMMC HAL モジュールのクロック構成

SDMMC MCU ペリフェラル（SDHI）は、PCLKA をそのクロックソースとして使用します。データレートを最適化する必要がない限り、SDMMC ペリフェラルに対して固有のクロックを設定する必要はありません。SDMMC ドライバーは、PCLKA 周波数と、SD、SDIO、または eMMC デバイスで許可される最大クロックレート（これはメディアデバイスの初期化時に取得されます）に基づいて、適切な内蔵分周器を選択します。

SDMMC HAL モジュールのピン構成

SDMMC 周辺機能（SDHI）の I/O ピンを構成するには、ISDE ピンコンフィギュレータを使用します。ほとんどのボード、高速メモリ、SDIO デバイスにおいて、各品のドライブ能力は「中」または「高」に設定する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

注：動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決まります。

r_riic での SDMMC HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SDHI	Pins	Select Peripherals > Storage:SHDI > SDHI0

注 : 選択シーケンスでは、SCI1 がドライバーに必要なハードウェアターゲットであることを想定しています。

SDHI ペリフェラルのピン構成設定

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Custom, SD_MMC 1 bit SD_MMC 4 bit MMC 8 bit Default: Custom	Select mode as per application
CLK	None, P413 Default: P413	Clock Pin
CMD	None, P412 Default: P412	Command Pin
DAT0-7	None, PXXX Default: PXXX	Data Pin
CD	None, P903 Default: P903	Card detection Pin
WP	None, P414 Default: P414	Card write protection pin

Pin Configuration Property	Value	Description
SDA	Disabled	SDA Pin (when Simple I2C is used)
SCL	Disabled	SCL Pin (when Simple I2C is used)

注 : 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.2.40.6 アプリケーションでの SDMMC HAL モジュールの使用

アプリケーションで SDMMC HAL モジュールを使用する一般的な手順は次のとおりです。

1) open API を使用して、ドライバーを開きます。
2) read API を使用して、カードからデータを読み取ります。
3) write API を使用して、カードにデータを書き込みます。
4) read API または write API が呼び出された後に毎回、イベント SDMMC_EVENT_TRANSFER_COMPLETE（操作が正常に完了したことを意味します）または SDMMC_EVENT_TRANSFER_ERROR（操作が正常に完了しなかったことを意味します）のコールバックを待ちます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

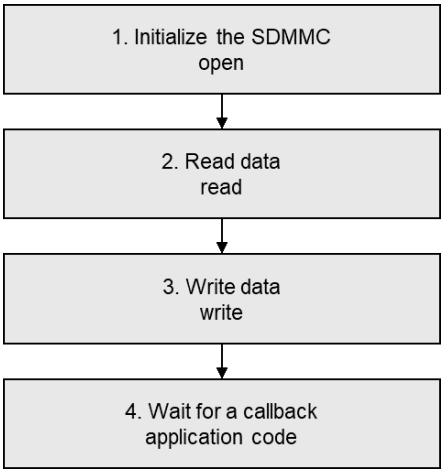


図 282:一般的な SDMMC HAL モジュールアプリケーションのフロー図

アプリケーションで SDMMC HAL モジュールを SDIO カードとともに使用する一般的な手順は次のとおりです。

1) open API を使用して、ドライバーを開きます。
2) readIo API または writeIo API を使用して、単一レジスタを設定するか読み戻します。動作はこれらの呼び出しの直後に完了し、コールバックを待つ必要はありません。
3) readIoExt API または writeIoExt API を使用して、複数レジスタを設定するか読み戻します。必要な場合は、呼び出しの間に control API を使用して、ブロックサイズを変更できます。
4) readIoExt API または writeIoExt API が呼び出された後に毎回、イベント SDMMC_EVENT_TRANSFER_COMPLETE（操作が正常に完了したことを意味します）または SDMMC_EVENT_TRANSFER_ERROR（操作が正常に完了しなかったことを意味します）のコールバックを待ちます。
5) カードが割り込みを要求する場合は、イベント SDMMC_EVENT_SDIO. でコールバックが呼び出されます。カードのドキュメントの説明に従って SDIO 割り込みを処理します（このドキュメントの「SDIO 割り込み」セクションを参照してください）。カードからの SDIO 割り込みは、IoIntEnable API を使用していつでも有効または無効にできます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

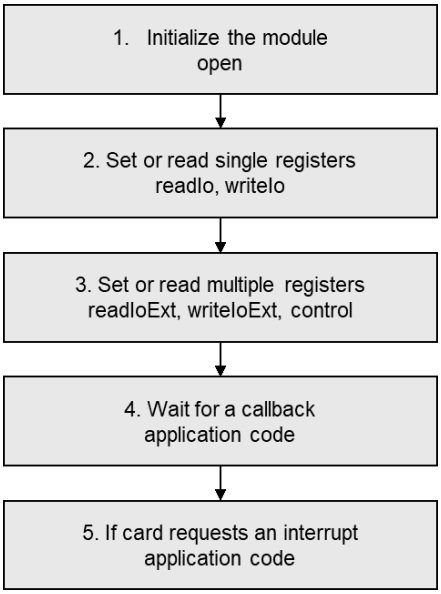


図 283:一般的な SDMMC HAL モジュールアプリケーションのフロー図

4.2.41 セグメント LCD ドライバー

セグメント LCD HAL モジュールは、セグメント LCD アプリケーションのためのハイレベルの API を提供し、セグメント LCD のデータを表示したり変更したりします。セグメント LCD HAL モジュールは、Synergy MCU 上のセグメント LCD コントローラモジュールを使用します。

4.2.41.1 SLCDC HAL モジュールの特長

- LCD ドライバー電圧ジェネレータの内部電圧ブースト : 容量分割方式または外部抵抗分割方式を選択します。
- 表示バイアス : 1/2 バイアス法、1/3 バイアス法、または 1/4 バイアス法を選択します。
- 表示のタイムスライス : 静的、2 時分割、3 時分割、4 時分割、または 8 時分割を選択します。
- 表示波形 : 波形 A または波形 B を選択します。
- 表示データ領域 : A パターン、B パターン、または点滅を選択します。表示データ領域は切り替えることができます。
- RTC 周期割り込み (PRD) を使用して、A パターンまたは B パターンの点滅表示を生成します。
- 参照電圧（電圧ブースト回路を稼働すると生成されます）を 16 ステップ（コントラスト調整）で調整します。

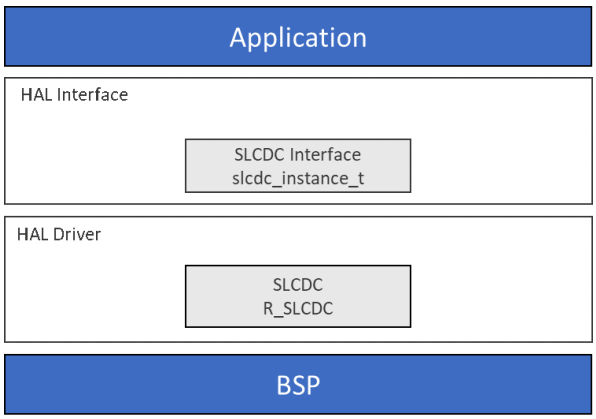


図 284:SLCDC HAL モジュールのブロック図

4.2.41.2 SLCDC HAL モジュールの API の概要

セグメント LCD コントローラ HAL モジュールでは、オープン、ライト、開始、変更、クローズなどの機能の API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

SLCDC HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_slcdc.p_api->open(g_slcdc.p_ctrl, g_slcdc.p_cfg); Open SLCD device.

Function Name	Example API Call and Description
write	g_slcdc.p_api->write(g_slcdc.p_ctrl, start_segment, &data, segment_count); Write data to SLCD segments. Specifies the initial display data. The data parameter is a pointer to an array of bytes consisting at least segment_count items, in which each byte is associated with one segment data register. When the number of time slices is static, 2, 3 or 4, the lower 4 bits of the data become an A-pattern area and the upper 4 bits become a B-pattern area. See for setting a display area.
modify	g_slcdc.p_api->modify(g_slcdc.p_ctrl, segment, data_mask, data); Rewrite data in the SLCD segment. Rewrites the LCD display data in 1-bit units. If a bit is not specified for rewriting, the value stored in the bit is held as it is. Specifies the data to rewrite.
start	g_slcdc.p_api->start(g_slcdc.p_ctrl); Enable display on the SLCD. Displays the specified data on the LCD. Before that data should be written to the segments.
stop	g_slcdc.p_api->stop(g_slcdc.p_ctrl); Disable display on the SLCD. Stops displaying data on the SLCD.
contrastIncrease	g_slcdc.p_api->contrastIncrease(g_slcdc.p_ctrl); Increase the display contrast. Increase by 1 unit. This function can be selected when the internal voltage boosting method is used for the drive voltage generator.

Function Name	Example API Call and Description
contrastDecrease	g_slcdc.p_api->contrastDecrease(g_slcdc.p_ctrl); Decrease the display contrast. Decrease by 1 unit. This function can be selected when the internal voltage boosting method is used for the drive voltage generator.
setdisplayArea	g_slcdc.p_api->setdisplayArea(g_slcdc.p_ctrl, display_area); Set LCD display area. This function sets a specified display area, A-pattern or B-pattern. This function can be used to set blink on where A-pattern and B-pattern area data will be alternately displayed. When using blinking, the RTC is required to operate before this function is executed. To configure the RTC, follow the steps below. 1) Open RTC 2) Set Periodic interrupt request, 1/2 second 3) Start RTC counter 4) Enable IRQ, RTC_EVENT_PERIODIC_IRQ Refer to the User's Manual: Microcontrollers for the detailed procedure.
close	g_slcdc.p_api->close(g_slcdc.p_ctrl); Close display device.
versionGet	g_slcdc.p_api->versionGet(&version); Retrieve the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successful
SSP_ERR_ASSERTION	Assertion error
SSP_ERR_INVALID_ARGUMENT	Invalid Argument
SSP_ERR_HW_LOCKED	SLCDC resource is locked
SSP_ERR_NOT_OPEN	Device is not opened or initialized
SSP_ERR_UNSUPPORTED	Unsupported operation
SSP_ERR_NOT_ENABLED	RTC not enabled for blink operation

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.41.3 SLCDC HAL モジュールの動作の概要

このモジュールはセグメント LCD コントローラ（SLCDC）を使用して、データをセグメント LCD に表示します。ドライバーは LCD を初期化してデータを表示し、ドライブ電圧ジェネレーター、ディスプレイ波形、タイムスライス数、および LCD を稼働するバイアスメソッドを設定します。このモジュールには、指定されたセグメントのセットにデータを表示するための関数、既存のセグメントデータを変更するための関数、ディスプレイを有効化および無効化するための関数、表示領域を設定するための関数、およびコントラストを調整するための関数が用意されています。LCD に表示される内容は、LCD 表示データレジスタの内容を変更することによって変更できます。

SLCDC HAL モジュールの動作に関する重要な注意事項と制限事項

- このドライバーは HAL ドライバーであり、ThreadX RTOS に依存しません。必要な場合は、セグメント LCD HAL モジュールを ThreadX RTOS のスレッドに追加することができます。
- セグメントのシーケンスへのライトを行うには、write API で書き込まれる開始セグメント番号およびセグメントの数を指定します。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.41.4 アプリケーションへの SLCDC HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに SLCDC HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

セグメント LCD ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(セグメント LCD ドライバーのデフォルト名は g_timer0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

SLCDC HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_slcdc0 Segment LCD Driver on r_slcdc	Threads	New Stack> Driver> Graphics> Segment LCD Driver on r_slcdc

次の図に示すように r_slcdc 上のセグメント LCD ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

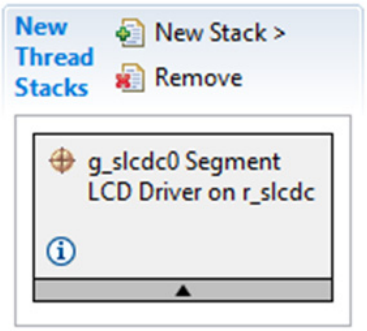


図 285:SLCDC HAL モジュールのスタック

4.2.41.5 SLCDC HAL モジュールの構成

ユーザーは必要な動作に合わせて SLCDC HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_slcdc での SLCDC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Select if extra code will be added to check parameter values
Name	g_slcdc0	Module Name
Slcdc Clock	Clock Loco, Clock Sosc, Clock Mosc, Clock Hoco Default: Clock Hoco	SLCD clock source (LCDSCKSEL).
Slcdc Clock Divisor	Clk Divisor LOCO factor of 2 from 4 to 1024 Clk Divisor HOCO factor of 2 from 256 to 524288 Default: Clk Divisor Hoco 16384	LCD clock setting (LCDC0), clock divisor
Bias Method	Bias 2, Bias 3, Bias 4 Default: Bias 2	LCD display bias method select (LBAS bit)
Time Slice	Static, Slice 2, Slice 3, Slice 4, Slice 8 Default: Static	Time slice of LCD display select (LDTY bit)
Wave Form	Wave A, Wave B Default: Wave A	LCD display waveform select (LWAVE bit).

ISDE Property	Value	Description
Slcdc Drive Voltage Generator	External resistance division, Internal voltage boosting, Capacitor split Default: External resistance division	LCD Drive Voltage Generator Select (MDSTET bit).

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

SLCDC HAL モジュールのクロック構成

SLCDC Hal モジュールの g_slcdc ドライバーのプロパティウィンドウで、SLCDC Hal モジュールクロックを構成します。SLCDC HAL モジュールの動作クロックは、[Properties] ウィンドウの [SLCDC Clock] および [SLCDC Clock Divisor] の設定で指定します。セグメント LCD HAL モジュールのソースクロックは、統合ソリューション開発環境コンフィギュレータを使用してメイン（MOSC）、HOCO（高速クロック発振器）、LOCO（低速クロック発振器）またはサブクロック（SOSC）として構成できます。HOCO および LOCO の設定では、複数のクロック除数を使用できます。

SLCDC HAL モジュールのピン構成

SLCDC 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はピンの選択例を示しています。

注 : 一部のペリフェラルでは、動作モードの選択によって、使用可能なペリフェラル信号および必要な MCU ピンが決定されます。

r_slcdc の SLCDC HAL モジュールでのピンの選択

Resource	ISDE Tab	Pin Selection Sequence
SLCDC	Pins	Select Peripherals > Graphics: SLCDC > SLCDC0

注 : 選択シーケンスでは、SLCDC0 がドライバーに必要なハードウェアターゲットであることを想定しています。

SLCDC HAL モジュールのピン構成設定

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Custom, Static, 2x Slice, 3x Slice, 4x Slice, 8x Slice Default: Custom	Select operation mode enable or disable
CAPH	None, P111 Default: None	Capacitor connection pin
CAPL	None, P112 Default: P112	Capacitor connection pin
COM0:3	None, Pn Default: P104:107	Common pins
COM4:7	None, Pn Default: None	Common pins
VL1:4	None, Pn Default: P100:103	Power supply pins
SEG00:02, SEG06:07, SEG16:17, SEG21:25, SEG46:51	None, Pn Default: None	Segment pins
SEG03	None, P303 Default: P303	Segment pin

Pin Configuration Property	Value	Description
SEG04:05	None, Pn Default: P314:315	Segment pins
SEG08	None, P902 Default: P902	Segment pin
SEG09:15	None, Pn Default: P312:306	Segment pins
SEG18:19	None, Pn Default: P808:809	Segment pins
SEG20	None, P313 Default: P313	Segment pin
SEG26:27	None, Pn Default: P806:807	Segment pins
SEG28:34	None, Pn Default: P608:614	Segment pins
SEG35:41	None, Pn Default: P606:600	Segment pins
SEG42:43	None, Pn Default: P805:804	Segment pins

Pin Configuration Property	Value	Description
SEG44:45	None, Pn Default: P800:801	Segment pins

注 : 設定例は、Synergy S7G2 MCU ファミリおよび SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と Synergy キットでは、使用可能なピン構成設定が異なる場合があります。

4.2.41.6 アプリケーションでの SLCDC HAL モジュールの使用

アプリケーションで SLCDC HAL モジュールを使用するときの一般的な手順は次のとおりです。

1) open API を使用して、SLCD HAL モジュールを初期化する
2) write API を使用して、セグメントのシーケンスを書き込む
3) 必要な場合は、modify API を使用してセグメントの内容を変更する
4) setdisplayArea API を使用して、表示領域または点滅表示を変更する
5) start API を使用して、表示を有効にする
6) contrastIncrease または contrastDecrease API を使用して、コントラストを調整する
7) stop API を使用して、表示を無効にする
8) close API を使用して、ドライバーを閉じる

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

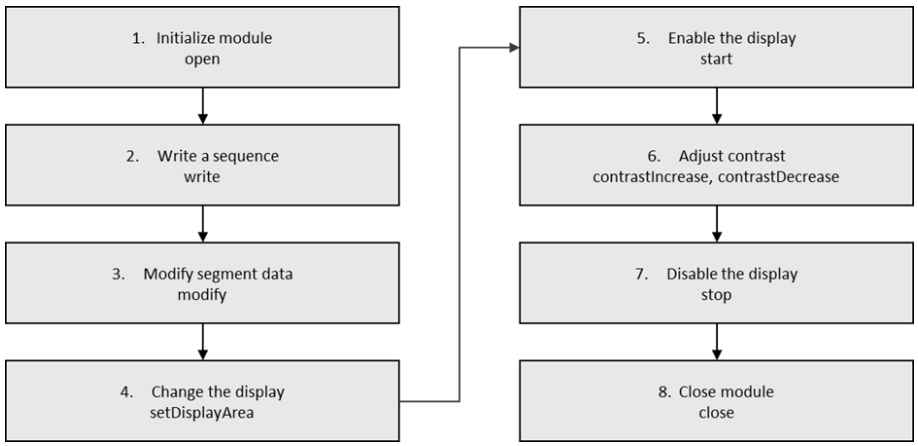


図 286:通常の SLCDC HAL モジュールアプリケーションのフロー図

4.2.42 SCI SPI ドライバー

SCI SPI HAL モジュールは、マスタベースの業界標準 SPI シリアル通信用のハイレベル API を提供し、Synergy MCU の SCI（シリアルコミュニケーションインタフェース）ペリフェラルを使用します。ユーザー定義のコールバックを作成し、SPI がデータの送信、データ転送の中断、またはエラー条件の検出を行ったときに信号を送信できます。

SCI SPI HAL モジュールは、MCU のデータトランスファコントローラモジュールを組み込むことにより、データ転送関数のサポートありで有効化されます。これは、CPU の介入なしに、DTC を通じて SPI 転送を実行します。

4.2.42.1 SCI SPI HAL モジュールの特長

SCI SPI HAL モジュールは、Synergy MCU の SPI 機能の構成と制御をサポートします。主な特長を以下に示します。

- ドライバーの初期化
- 8 ビットデータ転送を使用した SPI 動作によるシリアル通信
- 構成可能な 4 つのクロック位相とクロック極性の設定
- コールバックのサポート。コールバック関数は、次のイベントで呼び出されます。
  - 転送中断
  - 転送完了
  - オーバーランエラー
  - マスタモードでの SPI 通信。

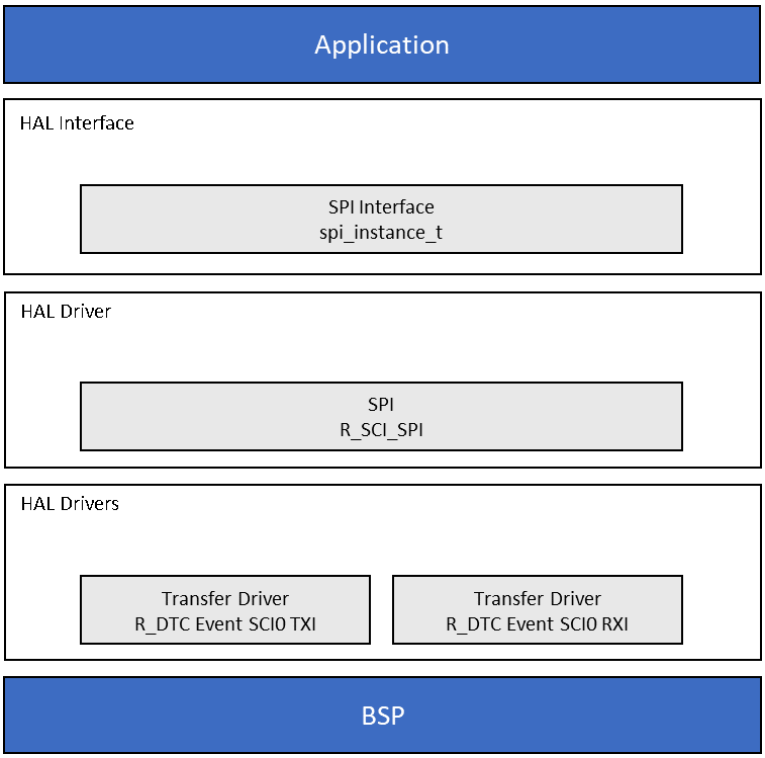


図 287:SCI SPI HAL モジュールのブロック図

4.2.42.2 SCI SPI HAL モジュールの API の概要

SPI では、SCI 周辺機能のオープンとクローズおよびデータの送信と受信のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

SCI SPI HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_spi.p_api ->open(g_spi.p_cntl, g_spi.p_cfg); Open a designated SPI device.
read	g_spi.p_api->read(g_spi.p_ctrl, dst16, length, SPI_BIT_WIDTH_16_BITS); Receive data from SPI device.
write	g_spi.p_api->write (g_spi.p_ctrl, source, length, SPI_BIT_WIDTH_8_BITS); Transmit data to SPI device
writeRead	g_spi.p_api ->writeRead (g_spi.p_cntl, &source, &destination, length, SPI_BIT_WIDTH_8_BITS, TX_WAIT_FOREVER); Simultaneously transmits data to an SPI device, while receiving data from an SPI device (full duplex). The writeRead API fetches the mutex object, handles SPI data transmission at SPI HAL layer, and receives data from the SPI HAL layer. The API uses the event flag wait to synchronize to complete the data transfer.

Function Name	Example API Call and Description
close	g_spi.p_api->close(g_spi,p_ctrl) Disable the SPI device designated by the control handle and close the RTOS services used by the bus if no devices are connected to the bus. This function removes power to the SPI channel designated by the handle and disables the associated interrupts.
versionGet	g_spi.p_api ->versionGet (&version); Get the version information of the underlying driver.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_IN_USE	Attempted to open an already open device instance OR Another transfer was in progress.
SSP_ERR_INVALID_POINTER	p_version is NULL.
SSP_INVALID_ARGUMENT	Channel number invalid.
SSP_ERR_HW_LOCKED	The lock could not be acquired. The channel is busy.
SSP_ERR_CH_NOT_OPEN	The channel has not been opened. Open channel first.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.42.3 SCI SPI HAL モジュールの動作の概要

SCI SPI HAL モジュールでは、Synergy MCU の SPI 機能を構成して使用することができます。SCI SPI HAL モジュールでは、SPI 通信プロトコルを使用してペリフェラルデバイスと通信できます。SCI SPI HAL モジュールのインスタンスを開いた後は、SCI SPI モジュールのハンドルを使用して、さまざまな転送動作を実行します。デバイス制御ハンドルは、通信対象の特定の SCI SPI デバイスを示すために、API 呼び出しの中で使用されます。

SCI SPI HAL モジュールでは、以下のことが可能です。

- モジュールを初期化します。
- SPI オペレーションによるシリアル通信です。SPI デバイスからのリードおよび SPI デバイスへのライト（および同時リード / ライト – 全二重）は、read、write、および writeRead API をコールすることによって実行されます。

ドライバーは、コールバックへのサポートも提供します。コールバック関数は、次のイベントで呼び出されます。

- 転送中断
- 転送完了
- オーバーランエラー

SCI SPI モジュールは、8 ビットのデータ転送動作のみをサポートします。SCI SPI モジュールは、チップセレクトとして構成された GPIO ピンを使用します。

クロックの設定

SCI SPI は、PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、実行時に e^2 studio または CGC Interface のクロックコンフィギュレータを使用します。

I/O ポートの設定

SPI を使用するには、出力ピンとして使用する I/O ポート pin(s) が、ピンコンフィギュレータで SCI SPI 周辺ピンとして構成されている必要があります。外部チップセレクトの場合は、GPIO 出力としてチップセレクトピンを構成します。

SCI SPI 割り込み

SCI SPI の割り込みを有効にするには、e^2 studio のプロジェクトコンフィギュレータの [Threads] タブで、ドライバーモジュールを強調表示して、SCI RXI、TXI、TEI、および ERI の各割り込みのプライオリティを設定します（Configuring Interrupts）。

これにより、ssp_cfg/bsp/bsp_irq_cfg.h で、対応する割り込みが、選択したプライオリティレベルに設定されます。

注 : 割り込みを異なる優先度に設定すると、適切に動作しなくなる可能性があります。

SCI SPI HAL モジュールの動作に関する重要な注意事項と制限事項

- チップセレクト出力は、GPIO を使用してサポートされます。
- SCI SPI HAL モジュールは、8 ビットデータ転送のみを使用します。
- 割り込みを異なる優先度に設定すると、適切に動作しなくなる可能性があります。
- SCI SPI HAL モジュールは、MCU のデータトランスファコントローラモジュールを組み込むことにより、データ転送サポートありで有効化されます。これは、CPU の介入なしに、DTC を通じて SPI 転送を実行します。DTC 転送はデフォルトで有効になっており、IRQ モード転送を使用する場合は、コンフィギュレータから DTC 転送を削除する必要があります。
- SCI に SPI を実装するときは、現在マスタモードのみがサポートされます。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.42.4 アプリケーションへの SCI SPI HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに SPI HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

SPI ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(SPI ドライバーのデフォルト名は g_spi0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

SCI SPI HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_spi0 SPI Driver on g_sci_spi	Threads	New Stack> Driver> Connectivity> SPI Driver on g_sci_spi

次の図に示すように、r_sci_spi の SPI ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

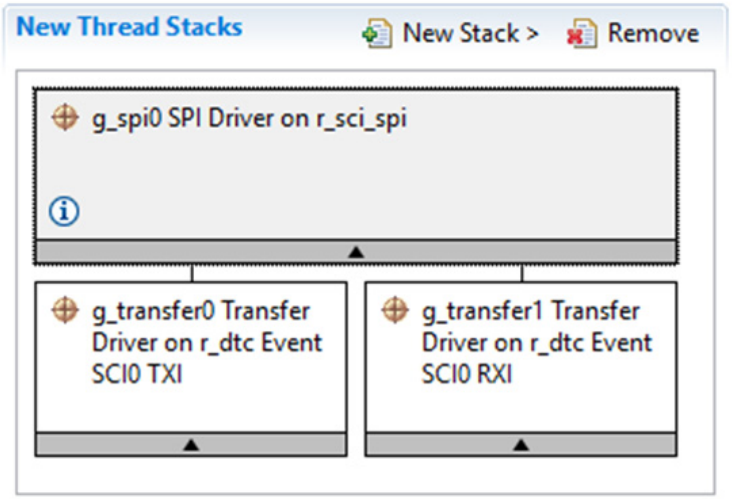


図 288:**SPI HAL モジュールのスタック**

4.2.42.5 SCI SPI HAL モジュールの構成

ユーザーは必要な動作に合わせて SPI HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_sci_spi での SCI SPI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_spi0	Module name

ISDE Property	Value	Description
Channel	0	SCI or SPI Channel number to which the device has been connected.
Operating Mode	Master	Configure as a Master or Slave device. Note: Current version of SSP supports only SPI Master mode.
Clock Phase	Data sampling on odd edge, data variation on even edge/Data sampling on even edge, data variation on odd edge Default: Data sampling on odd edge, data variation on even edge	Data sampling on odd or even clock edge.
Clock Polarity	Low when idle, High when idle Default: Low when idle	Clock level when idle.
Mode Fault Error	Enable, Disable Default: Disable	Indicates Mode fault error (master/slave conflict) flag.
Bit Order	MSB First, LSB First Default: MSB First	Select transmit order MSB/LSB first
Bitrate	100000	Transmission or reception rate. Bits per second.
Bit Rate Modulation Enable	Enable, Disable Default: Enable	Bitrate Modulation Function enable or disable

ISDE Property	Value	Description
Callback	NULL	Optional Call back function pointer.
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Receive interrupt priority selection
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Transmit interrupt priority selection.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Transmit end interrupt priority selection
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Error interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

SCI SPI HAL モジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_dtc Event SCI0 TXI 時の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency seleciton
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SCI0 TXI	Activation source selection
Auto Enable	False	Auto enable selection

ISDE Property	Value	Description
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection

ISDE Property	Value	Description
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SCI0 RXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Disabled	ELC software event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

SCI SPI HAL モジュールのクロック構成

SCI 周辺機能は、周辺クロック A（PCLKA）からクロックを供給されます。クロック周波数は、ISDE のコンフィギュレータの [Clocks] タブを使用して構成できます。無効な選択肢を選択すると、赤く表示されます。PCLKA の示されている値で必要な SPI ビットレートを実現できることを確認してください。指定したビットレートを実現できない場合、ISDE は表示されません。実行時に、SPI HAL モジュールは SCI 周辺機能を正しいビットレートに構成することを試み、目的のビットレートを設定できない場合はエラーを返します。ビットレートは、次の表の式を使用して計算されます。式の結果（N）が 0 ～ 255 の範囲内にある場合は、そのビットレートを実現できます。

ボーレートの計算式

SPI HAL	Bitrate calculation	Description
SPI on SCI	$N = \frac{PCLKA\ (MHz)}{8 * 2^{(2n-1)} * \left(256/_{M}\right) * B} - 1$	N = Peripheral register value. This must be in the range of 0 to 255 PLCKA = value of PLCKA in MHz n = 0, 1, 2 or 3 M = Bit Rate Modulation Index 128 < M < 256 *If the Bit Rate Modulation is disabled, then M=256* B = Desired Bit Rate

SCI SPI HAL モジュールのピン構成

SCI 周辺機能は、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は SPI ピンの選択例を示しています。

注 : 動作モードの選択によって、使用可能なペリフェラル信号および必要な MCU ピンが決定されます。

r_sci_spi の SCI SPI HAL モジュールのピンの選択シーケンス

Resource	ISDE Tab	Pin Selection Sequence
SPI on SCI	Pins	Select Peripherals > Connectivity:SCI > SCIx** **Where x is the required SCI peripheral channel.

注 : この選択シーケンスでは、SCI0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_sci_spi での SCI SPI HAL モジュールのピン構成設定

Pin Configuration Property	Value	Description
Pin Group Selection	Mixed, _A only, _B only	Synergy devices support peripheral functionality via multiple pins location, identified by _A, _B. Selecting Mixed allows the user to select any combination of locations (_A and _B). Selecting _A allows the user to select only _A locations. Selecting _B allows the user to select only _B locations.
Operation Mode	Disabled Custom Asynchronous UART Simple SPI Simple I2C Synchronous UART Smart Card	Set the operating mode to: Simple SPI.
TXD_MOSI	None, P411, P101	Specify the port pin to be used as MOSI.
RXD_MISO	None, P410, P100	Specify the port pin to be used as MISO.
SCK	None, P412, P102	Specify the port pin to be used as CLK.
CTS_RTS_SS	None, P413, P103	Specify the port pin to be used as SS.

注 : 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.2.42.6 アプリケーションでの SCI SPI HAL モジュールの使用

一般的なアプリケーションで SCI SPI HAL モジュールを使用する手順は次のとおりです。

注 :open API 関数は最初にコールする必要があります。残りの呼び出しは、アプリケーションの要件に応じて任意の順序で使用できます。

1) open API 関数を使用して SPI インスタンスを初期化します。(g_spi.p_api->open (g_spi.p_ctrl, g_spi.p_cfg)。ここで、p_ctrl および p_cfg は構成手順の後で自動生成される制御構造体と構成構造体のインスタンスです。)
2) write API 関数を使用して、スレーブデバイスへのライトを開始します。(g_spi.p_api->write (g_spi.p_ctrl, source, length, SPI_BIT_WIDTH_8_BITS);。ここで、g_spi.p_ctrl は open コールで使用されたものと同じ制御インスタンスです)。
3) read API 関数を使用して、スレーブデバイスからのリードを開始します。(g_spi.p_api->read(g_spi.p_ctrl, dst, length, SPI_BIT_WIDTH_8_BITS);。ここで g_spi.p_ctrl は open コールで使用されたものと同じ制御インスタンスです)。
4) writeRead API 関数を使用して、スレーブデバイスとの双方向でのデータ転送を開始します。(g_spi.p_api->writeRead(g_spi.p_ctrl, source, s_length, destination, d_length, SPI_BIT_WIDTH_8_BITS);。ここで、g_spi.p_ctrl は open コールで使用されたものと同じ制御インスタンスです。
5) close API 関数を使用して、インスタンスを閉じます。(g_spi.p_api->close(g_spi,p_ctrl)。ここで、g_spi.p_ctrl は、open コールで使用されたものと同じ制御構造体です)。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

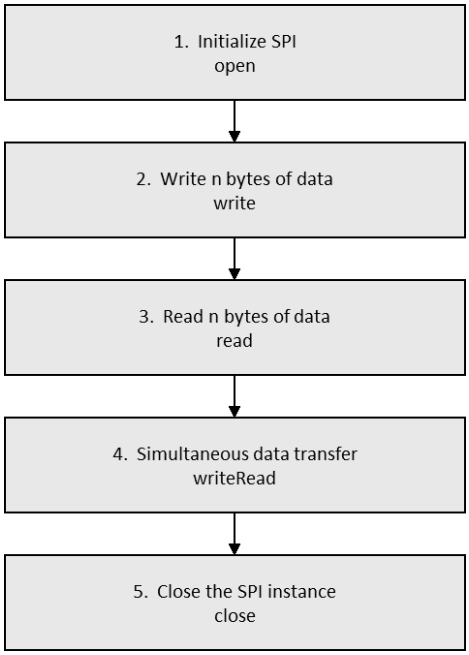


図 289:一般的な SPI HAL モジュールアプリケーションのフロー図

4.2.43 SPI ドライバー

RSPI HAL モジュールは、SPI プロトコルを使用したシリアル通信用のハイレベル API を提供します。このモジュールは、Synergy マイクロコントローラハードウェア上で利用可能な SPI（以前の RSPI）ペリフェラルをサポートしています。RSPI HAL モジュールは、標準の SPI マスタおよびスレーブモード通信機能をサポートします。転送イベントに対してコールバックが提供されています。RSPI HAL モジュールは、MCU のデータ転送コントローラモジュールを組み込むことにより、データ転送サポートありで有効化され、各フレームの割り込み処理を必要とすることなく DTC を介して SPI 転送を実行します。

4.2.43.1 RSPI HAL モジュールの特長

- ドライバーの初期化
- SPI 転送機能 :
  - 4 ワイヤ方式を使用する SPI 動作によるシリアル通信が可能
  - マスタモードとスレーブモードでシリアル通信が可能
  - シリアル転送クロックの極性の切り替え
  - シリアル転送クロックの位相の切り替え
- データ形式
  - MSB ファーストと LSB ファーストの選択が可能
  - 転送ビット長を 8、16、32 ビットから選択可能
- エラー検出
  - モード障害の検出
  - オーバーランエラーの検出
  - パリティエラーの検出
- SSL 制御機能
  - マスタモードでは、チャネルごとに最大 4 つの SSL 信号（SSLn0 から SSLn3）を内部的に選択できます
  - 外部ハードウェアスレーブ選択をマスタモードで使用できます
- 割り込み
  - RSPI 受信割り込み（受信バッファフル）
  - RSPI 送信割り込み（送信バッファエンプティ）
  - RSPI エラー割り込み（モード障害エラー、オーバーランエラー、パリティエラー）
- 遅延
  - SPI クロック遅延の追加
  - スレーブ選択ネゲート遅延の追加
  - 次アクセス遅延の追加

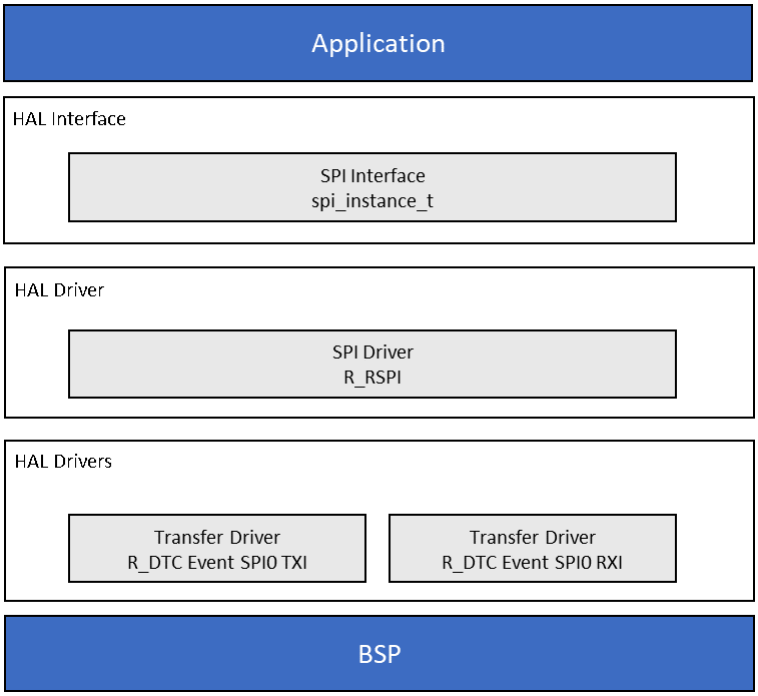


図 290:**RSPI HAL モジュールのブロック図**

4.2.43.2 RSPI HAL モジュールの API の概要

RSPI HAL モジュールでは、オープン、クローズ、リード、ライト、その他の有用な機能のための API 関数が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

RSPI HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_spi.p_api ->open(g_spi.p_cntl, g_spi.p_cfg); Open a designated SPI device.
read	g_spi.p_api->read(g_spi.p_ctrl, dst16, length, SPI_BIT_WIDTH_16_BITS); Receive data from SPI device.

Function Name	Example API Call and Description
write	g_spi.p_api->write (g_spi.p_ctrl, source, length, SPI_BIT_WIDTH_8_BITS); Transmit data to SPI device
writeRead	g_spi.p_api ->writeRead (g_spi.p_cntl, &source, &destination, length, SPI_BIT_WIDTH_8_BITS, TX_WAIT_FOREVER); Simultaneously transmits data to an SPI device, while receiving data from an SPI device (full duplex). The writeRead API fetches the mutex object, handles SPI data transmission at SPI HAL layer, and receives data from the SPI HAL layer. The API uses the event flag wait to synchronize to complete the data transfer.
close	g_spi.p_api->close(g_spi,p_ctrl) Disable the SPI device designated by the control handle and close the RTOS services used by the bus if no devices are connected to the bus. This function removes power to the SPI channel designated by the handle and disables the associated interrupts.
versionGet	g_spi.p_api ->versionGet (&version); Get the version information of the underlying driver.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function completed successfully
SSP_ERR_INVALID_MODE	Invalid mode

Name	Description
SSP_ERR_INVALID_CHANNEL	Invalid channel
SSP_ERR_IN_USE	In-use error
SSP_ERR_INVALID_ARGUMENT	Invalid argument
SSP_ERR_QUEUE_UNAVAILABLE	Queue unavailable
SSP_ERR_INVALID_POINTER	Invalid pointer
SSP_ERR_INTERNAL	Internal error
SSP_ERR_TRANSFER_ABORTED	Transfer aborted
SSP_ERR_MODE_FAULT	Mode fault
SSP_ERR_READ_OVF	Read overflow
SSP_ERR_PARITY	Parity error
SSP_ERR_OVERRUN	Overrun error
SSP_ERR_UNDEF	Unknown error
SSP_ERR_TIMEOUT	Timeout error
SSP_ERR_NOT_OPEN	Device not opened

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.43.3 RSPI HAL モジュールの動作の概要

RSPI HAL モジュールでは、SPI 通信プロトコルを使用してペリフェラルデバイスと通信できます。モジュールのインスタンスを開いた後、モジュールのハンドルを使用して、さまざまな転送動作を実行します。デバイス制御ハンドルは、通信対象の特定の SPI デバイスを示すために、API コールの中で使用されます。

ドライバーは、アプリケーションプログラムが以下を行うことができるようにします。

- ドライバーの初期化。
- SPI オペレーションによるシリアル通信の実装。

モジュールは、コールバックへのサポートも提供します。コールバック関数は、次のイベントの spi_event_t

でコールされます。

- 転送中断
- 転送完了
- モード障害
- エラー イベント

RSPI HAL モジュールは、8、16、32 ビットのデータ転送をサポートします。モジュールは、チップセレクトとして構成された GPIO ピンをサポートします。さらに、SPI 周辺機器は、専用チップ選択信号 SSLn0 から SSLn3 までをサ

ポートします。SPI 周辺機能で SSL ピンが有効になっていると、すべてのチップセレクト処理はハードウェアによって行われます。

クロック設定：SPI ペリフェラルは、PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、実行時に e^2 studio または CGC Interface のクロックコンフィギュレータを使用します。

注 :S1 デバイスの場合、SPI ペリフェラルクロックソースは PCLKB です。

IO ポート設定：SPI ペリフェラルで使用する場合、出力ピンとして使用される I/O ポートピンは、ピンコンフィギュレータで SPI ペリフェラルとして構成される必要があります。外部チップ選択を使用するには、チップを構成して、ピンを GPIO 出力として選択します。

拡張構成：SPI ドライバーには、拡張ハードウェア固有の構成が多数存在します。

注 : すべての拡張ハードウェア固有の構成パラメータは、拡張ドライバー構成構造体 spi_on_rspi_cfg_t で設定されます。

RSPI HAL モジュールの動作に関する重要な注意事項と制限事項

RSPI HAL ドライバーの構成中に、割り込みを異なる優先順位レベルに設定すると、適切に動作しなくなる可能性があります。

モジュールは、MCU のデータトランスファコントローラモジュールを組み込むことにより、データ転送サポートありで有効化されます。これは、CPU の介入なしに、DTC を通じて SPI 転送を実行します。

アプリケーションでは、DTC によるデータ転送は、通常の SPI 転送と同じ方法で使用されます。DTC 転送を有効にするには、RSPI HAL モジュールの下に DTC モジュールを追加します。

RSPI HAL モジュールは、CPU ベースと DTC ベース両方の転送モードで、8、16、32 ビットのデータ転送をサポートします。16 ビットと 32 ビットの転送は、エンディアンスワップされます。

性能に関する注意事項

RSPI HAL モジュールは、複数の異なるモードに構成でき、それぞれ性能特性が異なります。DTC をリセットするため、DTC 転送のセットアップには CPU ベースの転送よりわずかに長い時間がかかりますが、DTC 転送は、CPU による介入が必要ないため、1 フレームより長い転送の場合は優れた性能を実現します。

ライト動作では、モジュールは送信専用モードに構成され、受信割り込みは無効になり、受信データは無視されます。高ビットレートの CPU ベースのライト動作では、送信 ISR が繰り返しコールされて、他のコードの実行をブロックする可能性があります。writeRead および read 動作は、モジュールを全二重モードに構成します。

転送の間隔には 3 SPI クロックサイクルの下限が設けられているため、実質的なビットレートは構成より遅くなります。高ビットレートでは（特に CPU ベースの転送の場合）、転送と転送の間の時間がさらに長くなることがあります。

モジュールは、マスタモードのときはハードウェアがアイドル状態になるまで待ってから、スレーブモードのときはすべてのデータが送信または受信されるまで待ってから、コールバックを呼び出します。

- スレーブモードで RSPI HAL モジュールを使用する場合は、データを偶数クロックエッジでサンプリングするか、またはマスタがフレームとフレームの間でスレーブ選択ラインをアサート解除する必要があります。これはハードウェアでの制限事項です。
- S124、S128、S3A6 は、SSL レベルキープをサポートしていません。
- ドライバーが DTC を使用して開かれた後は、DTC で構成されている転送幅をすべての転送で使用する必要があります。
- 16 ビットと 32 ビットの転送は、エンディアンスワップされます。
- R_RSPI のビットレート値は 30 MHz または PCLK/2 未満の正の整数でなければなりません（どちらか最小の方）。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.43.4 アプリケーションへの RSPI HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに SPI HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

SPI ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(SPI ドライバーのデフォルト名は g_spi0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

RSPI HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_spi0 SPI Driver on r_rspi	Threads	New Stack> Driver> Connectivity> SPI Driver on r_rspi

次の図に示すように、r_rspi の SPI ドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

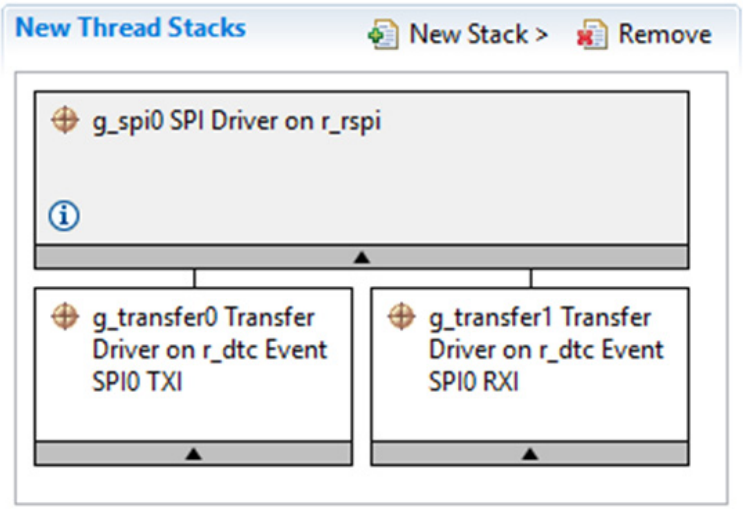


図 291:SPI HAL モジュールのスタック

4.2.43.5 RSPI HAL モジュールの構成

ユーザーは必要な動作に合わせて SPI HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_rspi での RSPI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_spi0	Module name

ISDE Property	Value	Description
Channel	0	SCI or SPI Channel number to which the device has been connected.
Operating Mode	Master, Slave Default: Master	Configure as a Master or Slave device . Note: Current version of SSP supports only SPI Master mode.
Clock Phase	Data sampling on odd edge, data variation on even edge/Data sampling on even edge, data variation on odd edge Default: Data sampling on odd edge, data variation on even edge	Data sampling on odd or even clock edge.
Clock Polarity	Low when idle, High when idle Default: Low when idle	Clock level when idle.
Mode Fault Error	Enable, Disable Default: Disable	Indicates Mode fault error (master/slave conflict) flag.
Bit Order	MSB First, LSB First Default: MSB First	Select transmit order MSB/LSB first
Bitrate	500000	Transmission or reception rate. Bits per second.
Callback	NULL	Optional Callback function pointer.

ISDE Property	Value	Description
SPI Mode	SPI Operation, Clock synchronous operation Default: SPI Operation	Select spi or clock syn mode operation.
SPI Communication Mode	Full Duplex, Transmit Only Default: Full Duplex	Select full-duplex or transmit-only communication.
Slave Select Polarity(SSL0)	Active Low, Active High Default: Active Low	Select SSL0 signal polarity
Slave Select Polarity(SSL1)	Active Low, Active High Default: Active Low	Select SSL1 signal polarity
Slave Select Polarity(SSL2)	Active Low, Active High Default: Active Low	Select SSL2 signal polarity
Slave Select Polarity(SSL3)	Active Low, Active High Default: Active Low	Select SSL3 signal polarity
Select Loopback1	Normal, Inverted Default: Normal	Select loopback1
Select Loopback2	Normal, Inverted Default: Normal	Select loopback2
Enable MOSI Idle State	Enable, Disable Default: Disable	Select MOSI idle fixed value and selection

ISDE Property	Value	Description
MOSI Ildle State	MOSI Low, MOSI High Default: MOSI Low	Select mosi idle fixed value and selection
Enable Parity	Enable, Disable Default: Disable	Enable/disable parity
Parity Mode	Parity Odd, Parity Even Default: Parity Odd	Select parity
Select SSL(Slave Select)	SSL0, SSL1, SSL2, SSL3 Default: SSL0	Select which slave to use; 0-SSL0; 1-SSL1; 2-SSL2; 3-SSL3
Select SSL Level After Transfer	SSL Level Keep, SSL Level Do Not Keep Default: SSL Level Do Not Keep	Select SSL level after transfer completion; 0-negate; 1-keep
Clock Delay Enable	Clock Delay Enable, Clock Delay Disable Default: Clock Delay Disable	Clock delay enable selection
Clock Delay Count	Clock Delay 1 thru 8 RSPCK Default: Clock Delay 1 RSPCK	Clock delay count selection
SSL Negation Delay Enable	Negation Delay Enable, Negation Delay Disable Default: Negation Delay Disable	SSL negation delay enable selection

ISDE Property	Value	Description
Negation Delay Count	Negation Delay 1 thru 8 RSPCK Default: Negation Delay 1 RSPCK	Negation delay count selection
Next Access Delay Enable	Next Access Delay Enable, Next Access Delay Disable Default: Next Access Delay Disable	Next access delay enable selection
Next Access Delay Count	Next Access Delay 1 thru 8 RSPCK Default: Next Access Delay 1 RSPCK	Next access delay count selection
Receive Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Receive interrupt priority selection
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Transmit interrupt priority selection
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Transmit end interrupt priority selection
Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Priority 2	Error interrupt priority selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

SPI HAL モジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_dtc Event SPI0 TXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte, 2 Bytes, 4 Bytes Default: 2 Bytes	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	
Destination Pointer	NULL	Destination pointer selection

ISDE Property	Value	Description
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SPI0 TXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SPI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer1	Module name
Mode	Normal	Mode selection

ISDE Property	Value	Description
Transfer Size	1 Byte, 2 Bytes, 4 Bytes Default: 2 Bytes	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SPI0 RXI	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

RSPI HAL モジュールのクロック構成

SPI 周辺機能は、周辺クロック A（PCLKA）からクロックを供給されます。クロック周波数は、ISDE のコンフィギュレータの [Clocks] タブを使用して構成できます。無効な選択肢を選択すると、赤く表示されます。PCLKA の示されている値で必要な SPI ビットレートを実現できることを確認してください。指定したビットレートを実現できない場合、ISDE は表示されません。実行時に、SPI ドライバーは SPI 周辺機能を正しいビットレートに構成することを試み、目的のビットレートを設定できない場合はエラーを返します。ビットレートは、次の表の式を使用して計算されます。式の結果（n）が 0 ～ 255 の範囲内にある場合は、そのビットレートを実現できます。

ボーレートの計算式

SPI HAL	Bitrate calculation	Description
SPI on SPI	$n = \frac{PCLKA\ (MHz)}{2 * 2^N * B} - 1$	n = Peripheral register value. Must be in the range of 0 to 255 PLCKA = value of PLCKA in MHz N = 0, 1, 2 or 3 B = Desired Bit Rate

RSPI HAL モジュールのピン構成

SPI 周辺モジュールは、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は SPI ピンの選択例を示しています。

注：動作モードの選択によって、使用可能なペリフェラル信号と必要な MCU ピンが決まります。

r_riic の SPI HAL モジュールに対するピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
RSPI	Pins	Select Peripherals > RSPI > SPI0_Pin_Option_A/B

注 : この選択シーケンスでは、SPI0 がドライバーに必要なハードウェアターゲットであることを想定しています。

r_sci_uart での SPI HAL モジュールのピン構成設定

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Custom, Enabled Default: Disabled	Select Enabled for SPI Operation
MISO	None, P100, P410 Default: None	MISO Pin selection
MOSI	None, P101, P411 Default: None	MOSI Pin selection
RSPCLK	None, P102, P412 Default: None	RSPCLK Pin selection
SSL0:3	None, P103:106, P413:415 Default: None	SSL0:3 Pin selections

注 : 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.2.43.6 アプリケーションでの SPI HAL モジュールの使用

一般的なアプリケーションで SPI HAL モジュールを使用する手順は次のとおりです。

注 :open API は最初にコールされる必要がありますが、残りのコールは、アプリケーションの要件に応じて任意の順序で使用できます。

1) open API を使用して、モジュールを初期化します。
2) write API を使用して、スレーブデバイスへのライトを行います。
3) read API を使用して、スレーブデバイスからのリードを行います。
4) close API をコールして、モジュールを閉じます。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

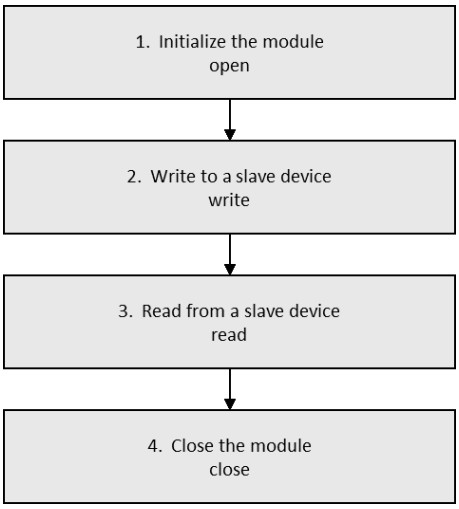


図 292:一般的な SPI HAL モジュールアプリケーションのフロー図

4.2.44 UART ドライバー

UART HAL モジュールは、業界標準のシリアル通信アプリケーション用のハイレベル API を提供し、Synergy MCU 上の SCI ペリフェラルを使用します。ユーザー定義のコールバックを作成し、必要に応じてハードウェアハンドシェイクとデータ操作を管理できます。

4.2.44.1 UART HAL モジュールの特長

UART HAL モジュールは、標準の UART プロトコルをサポートします。UART モード（SCI 上の UART）の SCI 周辺機能と共に使用される UART HAL モジュールは、(標準的な UART プロトコルに加えて) 次の機能をサポートします。

- 全二重 UART 通信
- 複数のチャネルを使用した同時通信
- 割り込み駆動型のデータ送信と受信
- イベントコードを引数として渡す、ユーザーコールバック関数の呼び出し
- 実行時のボーレートの変更
- UART トランザクション中のハードウェアリソースのロック
- CTS/RTS ハードウェアフロー制御（関連付けられた IOPORT ピンを使用し、ユーザー定義コールバック関数でサポート）
- DTC 転送モジュールとの統合
- 進行中のリード / ライト動作の中止

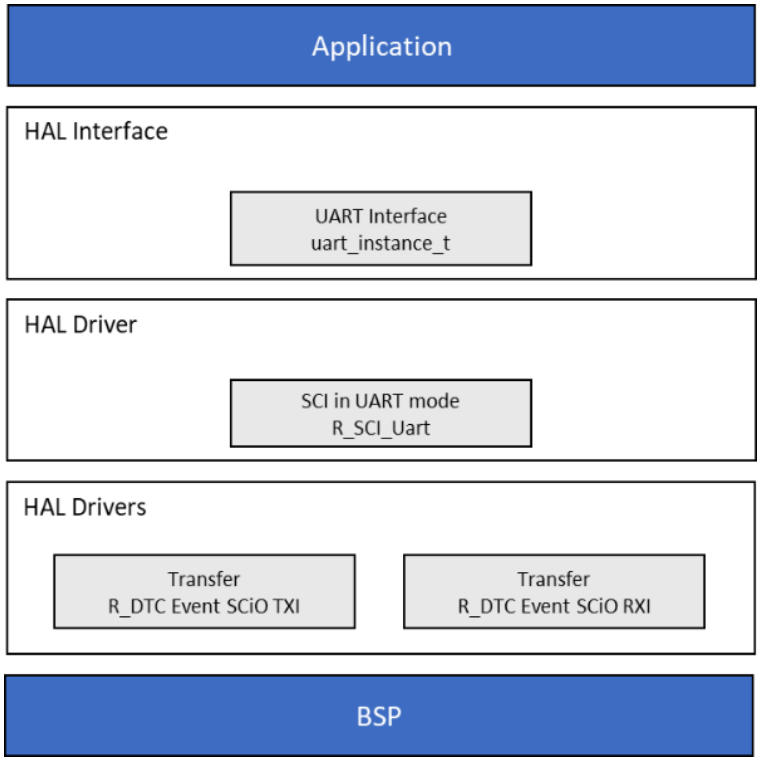


図 293:**UART HAL モジュールのブロック図**

4.2.44.2 UART HAL モジュールの API の概要

UART HAL モジュールのインタフェースでは、オープン、クローズ、リード、ライト、ボーレートの設定などの重要な機能のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

UART HAL モジュールの API の要約

Function Name	Example API Call and Description
open	g_uart0.p_api->open(g_uart0.p_ctrl, g_uart0.p_cfg); Open UART device.

Function Name	Example API Call and Description
read	g_uart0.p_api->read(g_uart0.p_ctrl, uart0_buf, uart0_rcv_num); Read from UART device. If a transfer instance is used for reception, the received bytes are stored directly in the read input buffer, uart0_buf. When a transfer is complete, the callback is called with event UART_EVENT_RX_COMPLETE. Bytes received outside an active transfer are received in the callback function with event UART_EVENT_RX_CHAR.
write	g_uart0.p_api->write(g_uart0.p_ctrl, uart0_buf, uart0_send_num) Write to UART device. The write buffer is used until write is complete. Do not overwrite write buffer contents until the write is finished. When the write is complete (all bytes are fully transmitted on the wire), the callback called with event UART_EVENT_TX_COMPLETE.
baudSet	g_uart0.p_api->baudSet(g_uart0.p_ctrl, (uint32_t)9600); Change baud rate.
infoGet	g_uart0.p_api->infoGet(g_uart0.p_ctrl, &uart_info); Get the driver specific information.
close	g_uart0.p_api->close(g_uart0.p_ctrl); Close UART device.
versionGet	g_uart0.p_api->versionGet(version); Retrieve the API version with the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Channel operates successfully.
SSP_ERR_IN_USE	Control block has already been opened or channel is being used by another instance.
SSP_ERR_ASSERTION	Pointer to UART control block is NULL or configuration structure is NULL.
SSP_ERR_HW_LOCKED	Channel is locked.
SSP_ERR_INVALID_MODE	Channel is used for non-UART mode or illegal mode is set.
SSP_ERR_INVALID_ARGUMENT	Invalid parameter setting found in the configuration structure. Or source/destination address or data size is invalid against data length.
SSP_ERR_NOT_OPEN	The control block has not been opened.
SSP_ERR_UNSUPPORTED	SCI_UART_CFG_RX_ENABLE is set to 0.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.44.3 UART HAL モジュールの動作の概要

UART HAL モジュールは、標準の UART プロトコルを使用するデータフローを管理します。ハイレベルの API は、リード、ライト、および UART インタフェースに対するボーレートの設定に使用されます。さらに、通常、割り込みはローレベルのアクティビティの管理を簡単にするために使用されます。

注 : 以下の機能が正常に動作するためには、割り込みを有効にする必要があります。

SCI RXI 割り込みでの UART

RXI 割り込みは、UART ポートから受信するデータのフローを制御するために使用されます。受信データ量が期待されるリード長に達した場合、ISR はユーザー定義のコールバック（p_callback）を呼び出して引数 uart_callback_args_t を渡し、受信データが完了したことを通知します。外部 RTS 動作オプションが有効になっている場合、ISR は RTS 外部ピン制御のための UART コールバック関数を 2 回呼び出します（ISR の先頭で 1 回、最後で 1 回）。このコールバック関数を使うと、RTS 関数をエミュレートできます（「SCI 上の UART のハードウェアフロー制御」のセクションを参照）。この割り込みは、構成パラメータ SCI_UART_CFG_RX_ENABLE で受信が有効になっている限りアクティブ化されます。

SCI 上の UART の TXI 割り込み

TXI 割り込みは、write API による要求に従い、UART ポートへの連続的なデータの転送を処理します。送信循環バッファにデータがなくなると、ISR は TXI 割り込みを非アクティブ化し、TEI 割り込みをアクティブ化して、データ送信の最後のシーケンスを処理します。この割り込みは、構成パラメータ SCI_UART_CFG_TX_ENABLE によって送信が有効になっている限り、write API でアクティブ化されます。

SCI 上の UART の TEI 割り込み

TEI 割り込みは、write API によって要求された UART ポートへの最後のデータ送信を処理するために使用されます。この割り込みは、TXI ISR によりアクティブ化され、自動的に非アクティブ化されます。ISR はユーザー定義のコールバック（p_callback）を呼び出して引数 uart_callback_args_t を渡し、データが最後まで送信されたことを通知します。

SCI 上の UART の ERI 割り込み

ERI 割り込みは、UART 受信で発生したエラーを処理するために使用されます。この割り込みは、構成パラメータ SCI_UART_CFG_RX_ENABLE によって受信が有効になっている限り、open API でアクティブ化されます。ISR はユーザー定義のコールバック（p_callback）を呼び出して引数 uart_callback_args_t を渡し、uart_event_t がエラーの原因であることを通知します。

UART HAL モジュールの動作に関する重要な注意事項と制限事項

SCI 上の UART のハードウェアフロー制御

SCI ハードウェアモジュールは、同時に RTS 信号または CTS 信号のいずれか一方のみのハードウェアフロー制御をサポートします。CTS と RTS は、CTSn/RTSn ピンで多重化され、ユースケースに応じて、いずれかのハードウェアフロー制御信号を排他的に使用できるようになっています。UART HAL モジュールは、この仕様を拡張し、RTS 信号用に追加のピンを有効にすることで、CTS 信号と RTS 信号の両方を制御できます。このモードを有効にするには、以下のように SCI の構成で UART を設定します。

- SCI_UART_CFG_EXTERNAL_RTS_OPERATION を有効に設定します。
- ctsrts_en を [CTS (true)] に設定します。
- p_extpin_ctrl で、ユーザーコールバック関数の名前を「Name of UART callback function for the RTS external pin control」に指定します。

SCI 上の UART HAL モジュールは、処理の最初と最後で、RXI ISR からユーザーコールバック関数を呼び出します。

コールバック関数の引数である level は、選択した SCI チャネルの RTS ピンの信号レベルを表します。

注意：HAL モジュールは、GPIO ピンの初期化の処理またはその制御を行いません。代わりに、ユーザーが、UART の受信を開始する前に GPIO ピンを初期化する必要があります。

次の図は、RTS 信号として外部 GPIO ピンが使用されている RTS/RTS ハードウェアフロー制御のタイミング図です。

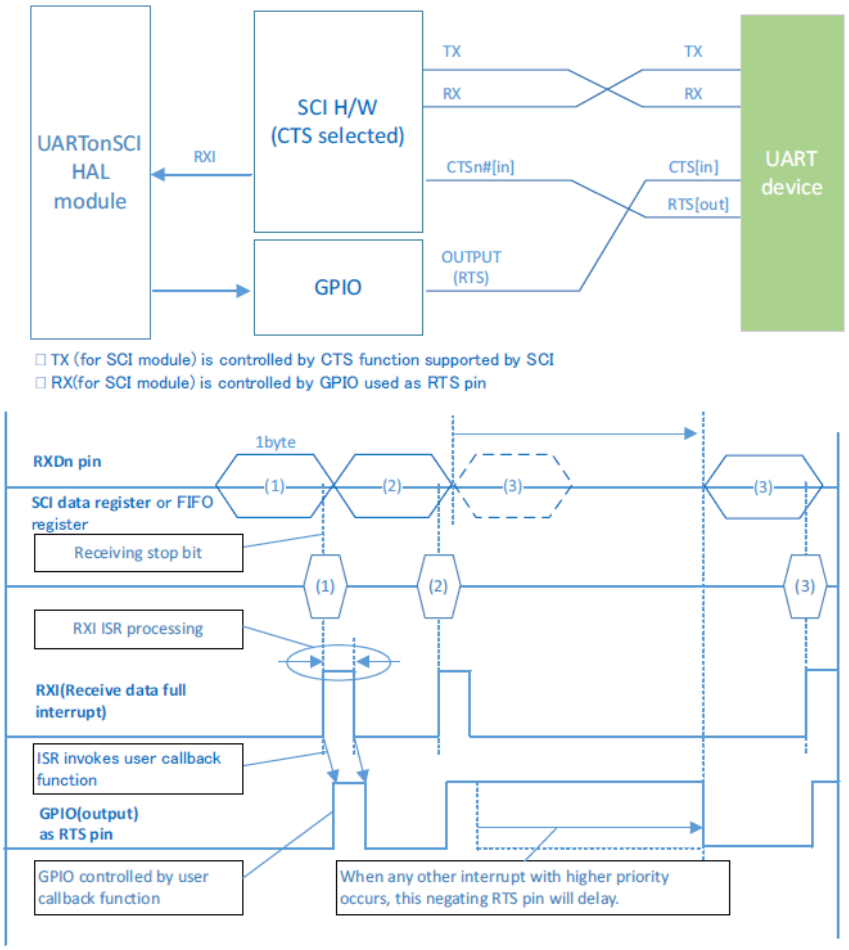


図 294:外部 GPIO を使用した UART HAL モジュールの CTS/RTS ハードウェア制御

注 :SK-S7G2 ボードの SCI 上の UART モジュールは、PORT8 の pin0（ピン P800）および J8 を使用して、RS-232C トランシーバ上の RS232C ポートをアクティブ化します。J8 のピン 1 とピン 2 を接続します。ピン P800 を IOPORT ピンとして構成し、そのレベルを目的の動作に合わせて設定します。

UART HAL モジュールの制限事項

- このモジュールは、割り込みベースの動作をサポートしていますが、ポーリング UART モードはサポートしていません。
- このモジュールは、非バッファリング UART モードをサポートしていません。
- このモジュールは、イベントリンク機能をサポートしていません。
- 転送で DTC インタフェースが使用される場合、転送サイズは 64K バイト以下でなければなりません。infoGet API を使用して、許可される最大転送サイズを取得できます。
- communicationAbort API が呼び出された後も、受信は引き続き有効です。中断の後、および次のリードのコール前に受信されたすべての文字は、イベント UART_EVENT_RX_CHAR. が指定されているコールバック関数を介して到着します。
- DTC インタフェースを受信に使用することはお勧めしません。使用した場合、データエラーが発生する可能性があります。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.44.4 アプリケーションへの UART HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに UART HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

UART ドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(UART ドライバーのデフォルト名は g_uart0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

UART HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_uart0 UART on r_sci_uart	Threads	Threads > Driver > Connectivity > UART Driver on r_sci_uart

次の図に示すように、r_sci_uart の UART HAL モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

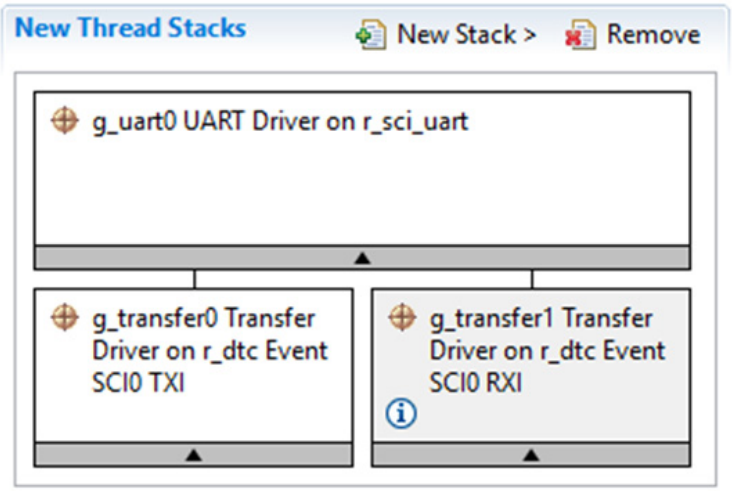


図 295:**UART HAL モジュールのスタック**

4.2.44.5 UART HAL モジュールの構成

ユーザーは必要な動作に合わせて UART HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_sci_uart での UART HAL モジュールの構成設定

ISDE Property	Value	Description
External RTS Operation	Enable, Disable Defaualt: Disable	Enable an IOPORT pin to be used as RTS signal. For RTS functionality set this configuration parameter to "Enable" and specify the configuration "Name of UART callback function for the RTS external pin control".

ISDE Property	Value	Description
Reception	Enable, Disable Defaualt: Enable	Enable or disable UART reception for all UART channels on SCI. Setting this configuration parameter to "Disable" reduces code size because the portion of code for UART reception is not compiled. You cannot set this parameter for individual UART channels.
Transmission	Enable, Disable Defaualt: Enable	Enable or disable UART transmission for all UART channels on SCI. Setting "Disable" to this configuration allows to get smaller code size due to the portion of code for UART transmission is compiled out, however, you can only set "Disable" to this configuration if any other SCI channels which work as UART ports do not perform the transmission.
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_uart0	The name to be used for UART on SCI module control block instance. This name is also used as the prefix of the other variable instances.
Channel	0	SCI channel number.
Baud Rate	9600	Baud rate selection.
Data Bits	7 bits, 8, bits, 9 bits Default: 8 bits	UART data bits.

ISDE Property	Value	Description
Parity	None, Odd, Even Default: None	UART parity bits.
Stop Bits	1 bit, 2 bits Default: 1 bit	UART stop bits.
CTS/RTS Selection	CTS (Note that RTS is available when enabling External RTS Operation mode which uses 1 GPIO pin), RTS (CTS is disabled) Default: RTS (CTS is disabled)	Select CTS or RTS for the CTSn/RTSn pin of SCI channel n. The SCI hardware supports either the CTS or the RTS control signal on this pin but not both. For an application that uses both CTS and RTS, select "CTS" for this configuration parameter and enable the configuration "External RTS Operation" specifying the configuration "Name of UART callback function for the RTS external pin control".
Name of UART callback function to be defined by user	user_uart_callback	Name must be a valid C symbol.
Name of UART callback function for the RTS external pin control to be defined by user	NULL	Name must be a valid C symbol.
Clock Source	Internal Clock, External Clock 8x baudrate, External Clock 16x baudrate Default: Internal Clock	Selection of the clock source to be used in the baud-rate clock generator block.
Baudrate Clock Output from SCK pin	Enable, Disable Default: Disable	Optional setting to output the baud-rate clock on the SCKn pin for the selected channel n.

ISDE Property	Value	Description
Start bit detection	Falling Edge, Low Level Default: Falling Edge	Start bit detection mode in the reception, usually set "Falling Edge" to this configuration.
Noise Cancel	Enable, Disable Default: Disable	Enable the digital noise cancellation on RXDn pin. The digital noise filter block in SCI consists of two-stage flip-flop circuits. For detail, refer to the Noise cancellation section in the Renesas Synergy hardware manual.
Bit Rate Modulation Enable	Enable, Disable Default: Enable	Bit rate modulation enable selection.
Receive FIFO Trigger Level	One, Max Default: Max	Receive FIFO trigger level selection
Receive Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Receive interrupt priority selection.
Transmit Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit interrupt priority selection.
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Transmit end interrupt priority selection.

ISDE Property	Value	Description
Error Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	Error interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

UART HAL モジュールのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_dtc Event SCI0 TXI 時の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Bytes	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection

ISDE Property	Value	Description
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SCI0 TXI	Activation source selection
Auto Enable	True, False Default: True	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc Event SCI0 RXI 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build

ISDE Property	Value	Description
Software Start	Enabled, Disabled Default: Disabled	Software start selection
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table selection
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Incremented	Destination address mode selection
Source Address Mode	Fixed	Source address mode selection
Repeat Area (Unused in Normal Mode	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Event SPI0 RXI	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection

ISDE Property	Value	Description
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Disabled	ELC Software Event interrupt priority selection.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。他の MCU のデフォルト値と使用可能な構成設定は異なる場合があります。

CTS 機能と RTS 機能で同時に UART を使用するときは、転送ドライバーを使用できません。ローレベルのすべての転送ドライバーを削除してください。削除した後、オプションの転送ドライバーはピンク色で表示されます。これはドライバーが推奨ではあってもオプションであることを意味します。

UART HAL モジュールのクロック構成

SCI UART 周辺機能は、PCLKA をクロックソースとして使用するか (S124 の PCLKB)、または選択されているチャネル n の SCKn ピンからの外部クロックを使用します。

UART HAL モジュールのピン構成

SCI UART 周辺機能は、MCU のピンを使用して外部デバイスと通信します。I/O ピンは、外部デバイスの要件に合うように選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は UART ピンの選択例を示します。

注 : 動作モードの選択によって、使用可能なペリフェラル信号および必要な MCU ピンが決定されます。

SCI の UART HAL モジュールのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SCI	Pins	Select **Peripherals > Connectivity: SCI > SCI0**

r_sci_uart 上の UART HAL モジュールのピン構成設定

Pin Configuration Property	Value	Description
Pin Group Selection	Mixed, _A Only, _B Only (Default: Mixed)	Pin grouping selection

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Custom, Asynchronous UART, Simple SPI, Simple I2C, Synchronous UART, SmartCard (Default: Simple SPI)	Select Operation Mode for UART on SCI
TXD_MOSI	None, P411, P101 (Default: P411)	TXD Pin
RXD_MISO	None, P410, P100 (Default: P410)	RXD Pin
SCK	None, P412, P102 (Default: P412)	SCK Pin
CTS_RTS_SS	None, P413, P103 (Default: None)	CTS Pin
SDA	Disabled	SDA Pin (when Simple I2C is used)
SCL	Disabled	SCL Pin (when Simple I2C is used)

注 : 設定例は、Synergy S7G2 および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.2.44.6 アプリケーションでの UART HAL モジュールの使用

一般的なアプリケーションで UART HAL モジュールを使用する手順は次のとおりです。

1) open API を使用して、UART HAL モジュールを初期化します。
2) Set Baud Rate with the baudSet API を使用して、ボーレートを設定します（必要な場合）。
3) 必要に応じて、read および write API とコールバックを使用して、データのリードとライトを行います。
4) リードおよびライト動作は、必要に応じて communicationAbort API を使用して中断できます。
5) close API を使用して、UART HAL モジュールを閉じます（必要な場合）。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

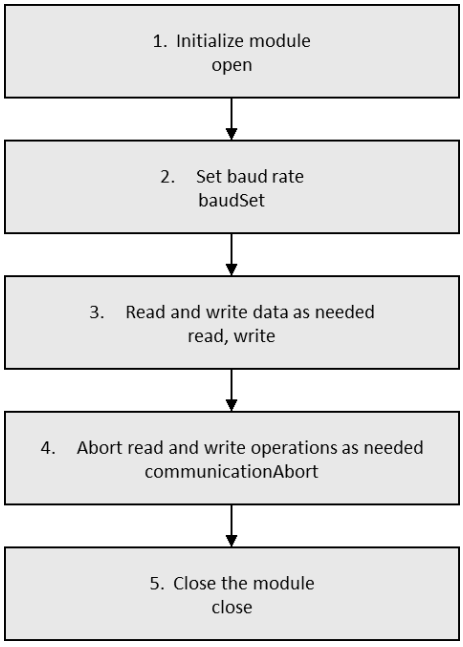


図 296:一般的な UART HAL モジュールアプリケーションのフロー図

4.2.45 ウォッチドッグ ドライバー

WDT（ウォッチドッグタイマ）HAL モジュールは、重要なタイミングアプリケーション用のハイレベル API を提供し、Synergy MCU 上の WDT ペリフェラルを使用します。ユーザー定義のコールバックを作成し、イベント通知に応答できます。

4.2.45.1 ウォッチドッグタイマ HAL モジュールの特長

WDT HAL モジュールには、以下の主要な機能があります。

- WDT が許可されたリフレッシュ ウィンドウの外でアンダーフローまたはリフレッシュされた場合、次のいずれかのイベントが生じる可能性があります。
  - デバイスのリセット
  - NMI の生成
- ウォッチドッグタイマ（WDT）周辺機能をサポートします。WDT は外部クロックを使用します。
- WDT は、WDT レジスタからレジスタ スタート モードで設定できます。
- リセット後の自動ハードウェア構成をサポートします。
- WDT はアプリケーションから開始することができます。

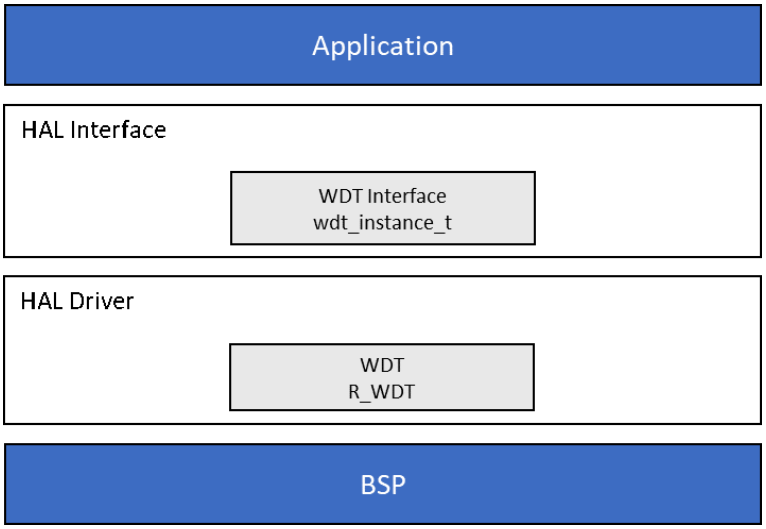


図 297:ウォッチドッグタイマ HAL モジュールのブロック図

4.2.45.2 ウォッチドッグタイマ HAL モジュールの API の概要

WDT HAL モジュールでは、状態のオープン、リフレッシュ、リード、取得のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

ウォッチドッグタイマ HAL モジュールの API の要約

Function Name	Example API Call and Description
cfgGet	g_wdt0.p_api->cfgGet(g_wdt0.p_ctrl, g_wdt0.p_cfg); Initialize the WDT in register start mode. In auto-start mode with NMI output it registers the NMI callback.
open	g_wdt0.p_api->open(g_wdt0.p_ctrl, g_wdt0.p_cfg); Initialize the WDT in register start mode. In auto-start mode with NMI output it registers the NMI callback.

Function Name	Example API Call and Description
refresh	g_wdt0.p_api->refresh(g_wdt0.p_ctrl); Refresh the watchdog timer.
statusGet	g_wdt0.p_api->statusGet(g_wdt0.p_ctrl, &status); Read the status of the WDT.
statusClear	g_wdt0.p_api->statusClear(g_wdt0.p_ctrl, clear); Clear the status flags of the WDT.
counterGet	g_wdt0.p_api->counterGet(g_wdt0.p_ctrl, &counter); Read the current WDT counter value.
timeoutGet	g_wdt0.p_api->timeoutGet(g_wdt0.p_ctrl, &timeout); Read the watchdog timeout values.
versionGet	g_wdt0.p_api->versionGet(&version); Retrieve the API version using the version pointer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	Function successfully executed.
SSP_ERR_ASSERTION	Null Pointer(s).

Name	Description
SSP_ERR_INVALID_ARGUMENT	One or more configuration options is invalid.
SSP_ERR_INVALID_MODE	An attempt to open the WDT in register-start mode when the OFS0 register is configured for auto-start mode. Or to open the WDT in auto-start mode when the OSF0 is configured for register start mode.
SSP_ERR_ABORTED	Invalid clock divider for this watchdog

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.2.45.3 ウォッチドッグタイマ HAL モジュールの動作の概要

Synergy MCU には、ウォッチドッグタイマ（WDT）と独立ウォッチドッグタイマ（IWDT）の 2 種類のウォッチドッグ周辺機能があります。どちらを選択するかは、以下の点を考慮して決定します。

- WDT はアプリケーションから開始することができます。
- WDT は、WDT レジスタからレジスタ スタート モードで設定できます。また WDT は、オプション関数選択レジスタ 0 (OFS0) に格納されているパラメーターを使用して、リセット後に自動的にハードウェアで設定することもできます。
- IWDT には、安全性を向上させる独自のクロック ソースがあります。
- IWDT は、オプション関数選択レジスタ 0 (OFS0) に格納されているパラメーターを使用して、リセット後に自動的にハードウェアで設定することができます。

ウォッチドッグタイマ HAL モジュールの動作に関する重要な注意事項と制限事項

WDT HAL モジュールは、WDT インタフェースを構成します。WDT が許可されたリフレッシュ ウィンドウの外でアンダーフローまたはリフレッシュされた場合、次のいずれかのイベントが生じる可能性があります。

- デバイスのリセット
- NMI の生成

次の図は、WDT の動作の例を示しています。カウンターの有効なリフレッシュ周期でリフレッシュされた場合、タイマー カウントの値はリセットされます。カウントがアンダーフローするか、有効なリフレッシュ周期外にリフレッシュされた場合、WDT はデバイスをリセットするか、NMI を生成します。

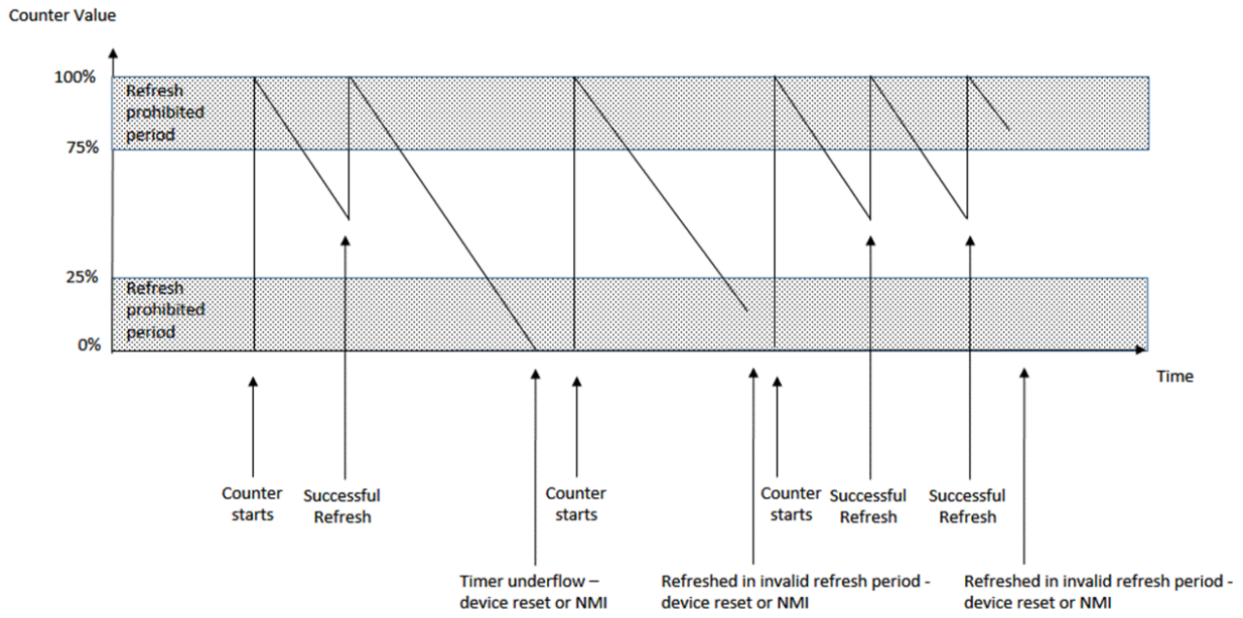


図 298:ウォッチドッグタイマ HAL モジュールの動作の図

WDT は、WDT レジスタからレジスタ スタート モードで設定できます。また、WDT は、次の表で示すように、オプション関数選択レジスタ 0（OFS0）に格納されているパラメータを使用して、リセット後にハードウェアで自動的に構成することもできます。

すべての Synergy マイクロコントローラのシリーズは、オプション設定メモリを備えています。このメモリを使用して、リセット後の周辺機能の動作状態を設定できます。OFS は、IWDT、WDT、LVD、CGC HOCO の状態の設定に使用できます。

次の表では、OFS レジスタで構成できる IWDT のパラメータを示します。

注 :IWDT は、OFS レジスタを使用することによってのみ構成できます。IWDT は、レジスタスタートモードをサポートしていません。

Control	Description
IWDT Start Mode Select	Automatically starts the IWDT after a Reset, if enabled.
IWDT Timeout Period	Specifies the IWDT timeout (number of clock cycles) 128 cycles 512 cycles 1024 cycles 2048 cycles

Control	Description
IWDT-Dedicated Clock Frequency Division Ratio	1 1/16 1/32 1/64 1/128 1/256
IWDT Window End Position	25% 50% 75% 100% (no window end position set)
IWDT Window Start Position	25% 50% 75% 100% (no window start position set)
IWDT Reset Interrupt Request	The IWDT can either generate an Interrupt Signal or a Reset signal.
IWDT Stop Control	The IWDT can continue to count or Stop counting in Low Power Mode.

*注 :OFS0 レジスタの内容の詳細については、Synergy MCU ハードウェアマニュアルを参照してください。*OFS レジスタの値は、次の図に示すように Synergy 構成エディタの [BSP] タブの [Properties] ダイアログで設定します。

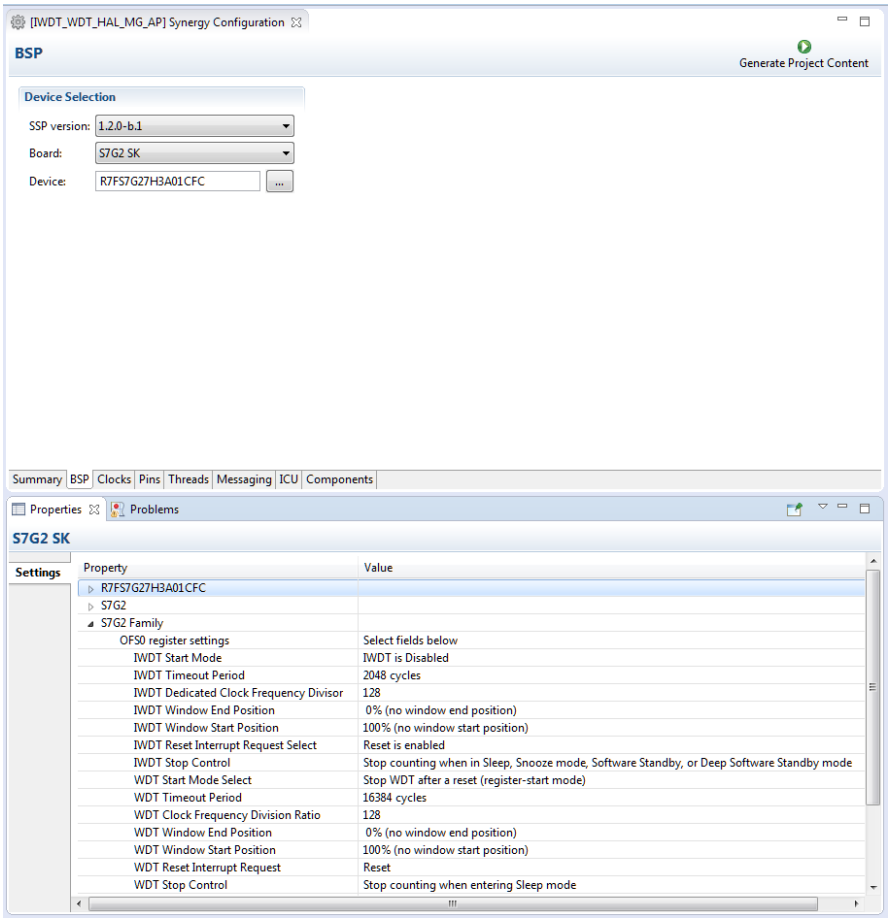


図 299:ウォッチドッグタイマ HAL モジュールの構成画面

WDT HAL モジュールの期間の計算

WDT は PCLKB から動作します。WDT で最大のパラメーターがあり、PCLKB が 60 MHz であると仮定した場合、最後のリフレッシュからデバイスのリセットまたは NMI の生成までの時間は、下記のように 2.2 秒をわずかに上回ります。

PLCKB = 60 MHz

クロック分割比 = PCLKB/8192

タイムアウト周期 = 16384 サイクル

WDT クロック周波数 = 60 MHz/8192 = 7.324 kHz

サイクル時間 = 1/7.324 kHz = 136.53 us

タイムアウト = 136.53 マイクロ秒 x 16384 サイクル = 2.23 秒

WDT HAL モジュールを使用した DMAC/DTC のトリガ

WDT カウンタがアンダーフローした場合、または有効なリフレッシュ周期外にリフレッシュが試みられた場合に、DMAC または DTC 周辺機能を使用してデータの転送をトリガするには、NMI を生成するように WDT を設定し、

activation_source を ELC_EVENT_WDT_UNDERFLOW. に設定して DMAC/DTC 転送を構成します。詳細については、対応するユーザーズガイド（DMAC、DTC）を参照してください。

WDT HAL モジュールでのイベントリンクコントローライベントのトリガ

WDT は、イベントリンクコントローラ（ELC）を使用して別の周辺機能の開始をトリガできます。使用可能な周辺機能の完全なリストについては、ELC のユーザーガイドを参照してください。

- JLink デバッガーを使用する場合、WDT カウンターはカウントしないため、デバイスのリセットや NMI の生成は行われません。
- 実行できるアクティブなタスクがない場合、ThreadX は MCU をスリープ モードにします。WDT が低消費電力モードでカウンタを停止するように構成されている場合、ThreadX RTOS で使用されているときは、アプリケーションでタイマを再度開始する必要があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.2.45.4 アプリケーションへのウォッチドッグタイマ HAL モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションにウォッチドッグタイマ HAL モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

ウォッチドッグタイマドライバーをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。（ウォッチドッグタイマドライバーのデフォルト名は g_wdt0. です。この名前は、対応する [Properties] ウィンドウで変更できます。）

ウォッチドッグタイマ HAL モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_wdt0 Watchdog Driver on r_wdt	Threads	New Stack>Driver>Monitoring>Watchdog Driver on r_wdt

次の図に示すように、r_wdt のウォッチドッグタイマドライバーがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

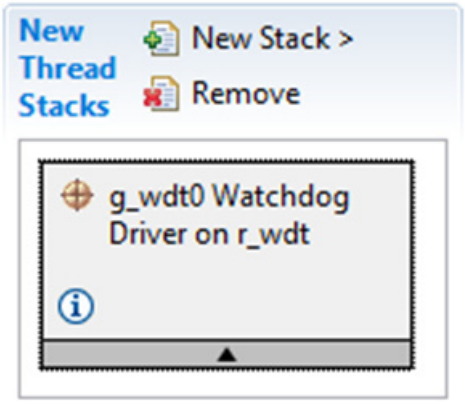


図 300:ウォッチドッグタイマ HAL モジュールのスタック

4.2.45.5 ウォッチドッグタイマ HAL モジュールの構成

ユーザーは必要な動作に合わせてウォッチドッグタイマ HAL モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

r_wdt のウォッチドッグタイマ HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Name	g_wdt0	Module Name
Start Mode	Register, Auto Default: Register	Configures the start mode as register start or auto-start.

ISDE Property	Value	Description
Start Watchdog After Configuration	True, False Default: True	Controls whether WDT is started during initialization
Timeout	1024 cycles, 4096 cycles, 8192 cycles, 16384 cycles Default: 16384 cycles	WDT timeout period.
Clock Division Ratio	PCLK/4, PCLK/64, PCLK/128, PCLK/512, PCLK/2048, PCLK/8192 Default: PCLK/8192	WDT clock divider.
Window Start Position	100% (Window Position Not Specified), 75%, 50%, 25% Default: 100% (Window Position Not Specified)	Permitted refresh period start postion.
Window End Position	0% (Window Position Not Specified), 25%, 50%, 75% Default: 0% (Window Position Not Specified)	Permitted refresh period end postion.
Reset Control	Reset Outpout, NMI Generated Default: Reset Output	Select whether WDT should reset the MCU or generate an NMI.
Stop Control	WDT Count Enabled in Low Power Mode, WDT Count Disabled in Low Power Mode Default: WDT Count Disabled in Low Power Mode	Select whether the WDT should stop counting in low power modes.

ISDE Property	Value	Description
NMI Callback	NULL	Callback. A user callback function can be registered in open. If this callback function is provided, it will be called from the interrupt service routine (ISR) each time the IRQn triggers. Warning: Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

オプション機能選択レジスタ 0（OFS0）の構成

すべての Synergy マイクロコントローラのシリーズは、オプション設定メモリを備えています。このメモリを使用して、リセット後の周辺機能の動作状態を設定できます。OFS は、IWDT、WDT、LVD、CGC HOCO の状態の設定に使用できます。このドキュメントで前述した動作の概要に関するセクションの説明を参照してください。

WDT HAL モジュールの割り込みの構成

WDT モジュールの他のオプション構成と同様に、WDT 割り込みを構成します。アンダーフローまたは無効なリフレッシュで NMI 割り込みを生成するよう WDT が構成されている場合、割り込みは BSP で有効になっている必要があります。

割り込みを有効にするには、[CWDT] > [CWDT NMIUNDF N n] で優先順位を設定します。これにより、ssp_cfg/bsp/bsp_irq_cfg.h の BSP_IRQ_CFG_WDT_UNDERFLOW が選択した優先順位レベルに設定されます。

CWDT NMIUNDF N 割り込みが BSP で有効になっている場合、対応する ISR が定義されます。ISR は、ユーザーコールバック関数が open API で登録されている場合、それをコールします。

ウォッチドッグタイマ HAL モジュールのクロック構成

初期設定では、WDT クロックは PCLKB 周波数に基づいて動作します。ISDE のクロックコンフィギュレータを使用して、または実行時に CGC インタフェースを使用して、PCLKB の周波数を設定できます。PCLKB が 60 MHz で動作している状態での最大タイムアウト周期は約 2.2 秒です。

ウォッチドッグタイマ HAL モジュールのピン構成

WDT は、動作のためにピンを必要としません。

4.2.45.6 アプリケーションでのウォッチドッグタイマ HAL モジュールの使用

アプリケーションでウォッチドッグタイマ HAL モジュールを使用する際の一般的な手順は次のとおりです。

1) open API を使用して、レジスタスタートモードまたは自動スタートモードで WDT HAL モジュールを初期化します。
2) cfgGet API を使用して、レジスタスタートモードまたは自動スタートモードの WDT HAL モジュールの構成を読み取ります。
3) refresh API を使用して、ウォッチドッグタイマをリフレッシュします。
4) statusGet API を使用して、WDT のステータスフラグを読み取ります。
5) statusClear API を使用して、WDT HAL モジュールの状態フラグとエラーフラグをクリアします。
6) counterGet API を使用して、WDT の現在のカウンタ値を読み取ります。
7) timeoutGet API を使用して、ウォッチドッグのタイムアウト値を読み取ります。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

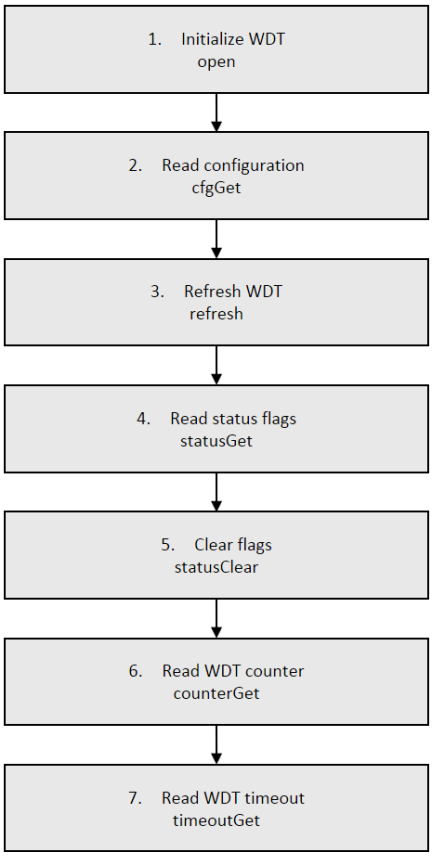


図 301:一般的なウォッチドッグタイマ HAL モジュールアプリケーションのフロー図

4.3 Express Logic 社のモジュール

Renesas SSP Express Logic モジュールに関する以下のドキュメントでは、広範なトピックが取り上げられています。これらのドキュメントは、開発プロセスの推進に必要な、特定の情報を簡単に見つけることができるように編成されています。必要な情報を素早く見つけるには、以下のポイントを確認してください。

- Express Logic コンポーネントは、ThreadX RTOS のサポート、FileX ファイルシステムのサポート、GUIX グラフィックスユーザインタフェースのサポート、USBX ユニバーサルシリアルバスインターフェースのサポート、NetX および NetX Duo ネットワーキングプロトコルのサポートという、主要なアプリケーション機能によって編成されています。以下の項目は、これらと同じ主要なアプリケーション領域に従って編成されています。
- Express Logic コンポーネントの中には、主要な要素や概念の多くを説明する概要ドキュメントが用意されているものがあります。この概要ドキュメントには、個別のモジュールの概要のすべてに含まれているものと重複した、一般的な情報も含まれている場合もあります。
- Express Logic コンポーネントには、ソースモジュール内で利用できる構成可能なオプションを説明する、ソースモジュールドキュメントが含まれているものがあります。これらのドキュメント、および関連する Express Logic 社のユーザーマニュアル（Web ページ https://www.renesas.com/en-us/products/synergy/software/ssp.html の下の方で、X-Ware コンポーネントドキュメントとして入手可能）は、何らかのソース構成設定の変更を試行する前に理解しておく必要があります。
- Express Logic コンポーネントの中には、SSP と Express Logic コンポーネントの間のインタフェースの詳細について説明したポートフレームワークのドキュメントが用意されているものがあります。たとえば、NetX ポートイーサネットモジュール（sf_el_nx）は、NetX や NetX Duo と、Synergy ハードウェアの間のインタフェースとしての役割を果たします。
- Express Logic コンポーネントには、さまざまなプロトコルまはた類似したローレベルの機能が用意されているものがあります。これらは一般に SSP に個別のモジュールとして組み込まれており、それぞれに動作について説明したモジュールの概要ドキュメントが用意されています。たとえば、DHCP、FTP、Telnet のそれぞれについて、個別のモジュールの概要が用意されています。
- Express Logic のネットワーキングコンポーネントには、複数のプロトコルを取り上げたモジュールの概要が含まれています。多くの場合、これは NetX と NetX Duo の実装が非常に似ており、違いについては、まとめられたドキュメントで簡単に確認することができるためです。これにより、不要な重複を回避することができます。（NetX Duo は IPv4 と IPv6 の両方をサポートしていますが、NetX は IPv4 のみをサポートします。動作の違いは、必要に応じて簡単に説明できます。）たとえば、NetX/NetX Duo HTTP サーバー、NetX/NetX Duo HTTP クライアントモジュールの概要では、これらの標準の機能の NetX および NetX Duo 実装の両方を取り上げています。

ThreadX

ThreadX

FileX

FileX ポート ブロック メディア フレームワーク

FileX ソースコンポーネントモジュール

GUIX

GUIX Synergy ポートフレームワーク

GX_SRC フレームワーク

LevelX

sf_el_lx_nor 上の Port LevelX フレームワーク

NetX/NetX Duo

NetX ポートイーサネット

NetX および NetX Duo ソースモジュールの概要

Express Logic 社の NetX の概要

Express Logic 社の NetX Duo の概要

NetX/NetX Duo Auto IP

NetX/NetX Duo BSD サポート

NetX/NetX Duo DHCP クライアント

NetX/NetX Duo DHCP サーバー

NetX Duo DHCP IPv6 クライアント

NetX Duo DHCP IPv6 サーバー

NetX/NetX Duo DNS クライアント

NetX/NetX Duo FTP クライアント

NetX/NetX Duo FTP サーバー

NetX/NetX Duo HTTP クライアント

NetX/NetX Duo HTTP サーバー

NetX Duo Web HTTP/HTTPs クライアント

NetX Web HTTP/HTTPs サーバーフレームワーク

NetX/NetX Duo SMTP クライアント

NetX/NetX Duo SNMP エージェント

NetX/NetX Duo SNTP クライアント

NetX/NetX Duo POP3 クライアント

NetX/NetX Duo Telnet クライアント

NetX/NetX Duo Telnet サーバー

NetX/NetX Duo TFTP クライアント

NetX/NetX Duo TFTP サーバー

NetX Duo MQTT クライアント

NetX Duo NAT

NetX Duo TLS セッション

USBX

Express Logic 社の USBX の概要

USBX ソースコンポーネントモジュール

USBX Synergy ポートフレームワーク

USBX デバイスクラス CDC-ACM

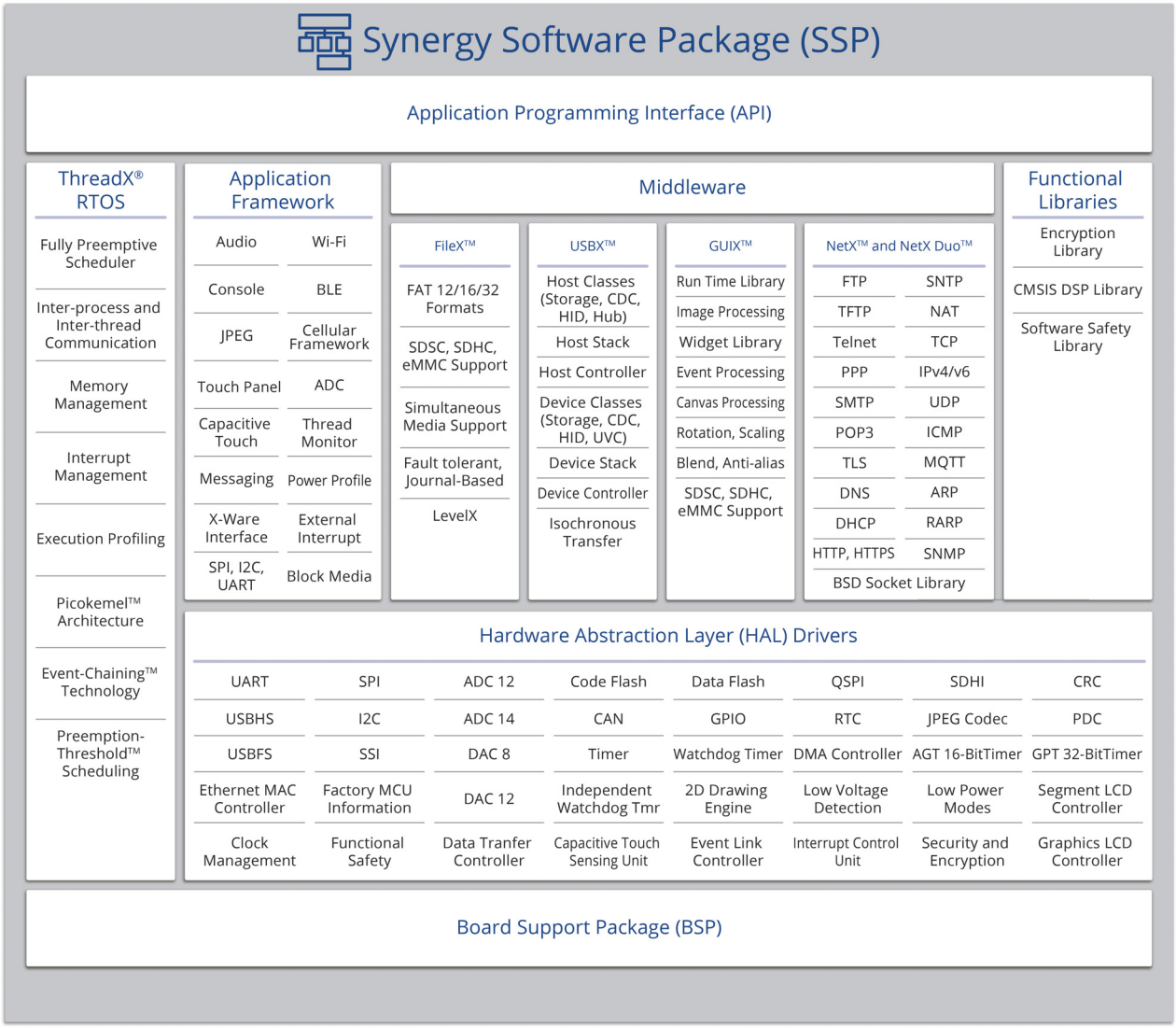
USBX デバイスクラス HID
USBX デバイスクラス記憶装置
USBX ホストクラス CDC-ACM
USBX ホストクラス HID
USBX ホストクラスハブ
USBX ホストクラス記憶装置
USBX ホストクラスビデオ

4.3.1 ThreadX

SSP には、Express Logic 社の ThreadX カーネル（tx）が組み込まれています。Express Logic 社の ThreadX カーネルは、マルチスレッド SSP アプリケーションの基盤です。スレッド作成サービスと同期サービスを提供しており、メッセージキュー、カウンティングセマフォ、ミューテックス、イベントフラグ、メモリブロックプール、メモリバイトプール、アプリケーションタイマなどを備えています。

4.3.1.1 Express Logic 社の ThreadX モジュールの特長

- Synergy プラットフォームに ThreadX RTOS を実装
- スレッドが作成されると自動的に追加
- ThreadX のソースモジュールを使用してソースコードを追加可能
- 以下のためのスレッドサービスを提供
  - 同期
  - メッセージキュー
  - カウンティングセマフォ
  - ミューテックス
  - イベントフラグ
  - メモリブロックおよびバイトプール
- アプリケーションタイマ



4.3.1.2 Express Logic 社の ThreadX モジュールの動作の概要

tx モジュールの Express Logic 社の ThreadX は、Synergy Software Package 内に ThreadX RTOS カーネルを実装します。これはスケジューリング、内部プロセスの通信、メモリ管理、割り込み管理、その他各種の RTOS 関連機能を提供します。一般に、アプリケーションコードはカーネルによって提供された機能を利用するだけです。動作に関する以下の注意は、ThreadX および ThreadX ソースを使用したさらに詳細な操作によって、アプリケーションに対してメリットがある可能性がある領域について説明しています。

Express Logic 社の ThreadX モジュールの動作に関する重要な注意事項と制限事項

プリエンプションしきい値

プリエンプションしきい値機能は、コードサイズを縮小するために、提供されている ThreadX ライブラリでは無効化されています。プリエンプションのサポートが必要な場合、ThreadX ソースをプロジェクトに組み込む必要があ

り、プリエンプションしきい値は ThreadX ソースプロパティで有効に設定されている必要があります。スレッドのプリエンプションしきい値は、コンフィギュレータによってスレッドのプライオリティに設定されています。これは、tx_thread_preemption_change 関数によって変更することができます。

IAR ライブラリサポート（IAR のみ）

IAR ツールのスレッドセーフサポートは、ThreadX ソースモジュールの [Properties] を以下のように設定するだけで有効にできます。この表で統合ソリューション開発環境プロパティは、e^2 studio 統合ソリューション開発環境の [Properties] タブの名前を示しています。

ThreadX ソース IAR ライブラリサポートの構成

ISDE Property	Setting	Description
IAR Library Support	Enabled, Disabled (Default: Disabled)	If enabled, defines TX_ENABLE_IAR_LIBRARY_SUPPORT to provide thread safe access to IAR standard library functions such as malloc() and printf().

また、以下の行をリンカ制御ファイルに追加します（まだ記述されていない場合）：

```
initialize by copy with packing = none { section __DLIB_PERTHREAD };
// Required in a multi-threaded application.
```

注 :TX_ENABLE_IAR_LIBRARY_SUPPORT. を変更する前に、ThreadX ソースモジュールを [Threads] タブ（HAL/Common）に追加します。注 :TX_ENABLE_IAR_LIBRARY_SUPPORT マクロは、IAR 実行時ライブラリへのスレッドセーフアクセスに必要なスレッドの作成および破壊の拡張機能を有効化します。TLS（スレッドローカルストレージ）メモリはヒープから各インスタンスに割り当てられます。現在、SSP は、スレッドセーフな方法でライブラリ関数を使用するのに必要な同期メカニズムを実装していません。あらかじめ用意された機能だけを使ってこのライブラリを使用しても安全なのは、再入不可能な方法でライブラリ関数にアクセスしている場合のみです。つまり、1 つのスレッドのみがこのライブラリへのコールを行う場合に限り、ライブラリを安全に使用できます。完全なマルチスレッドサポートを利用するには、IAR の IAR Dlib のスレッドサポートドキュメントで、同期メカニズムの実装方法の詳細を確認してください。この実装は、SSP によって生成されたファイルである tx_src_user.h ファイルで行う必要があります。

ThreadX ソースの高度な構成

ThreadX ソースモジュールをプロジェクトに追加した場合、[Properties] ウィンドウでは、ThreadX ソースライブラリの高度な構成を行うことができます。構成オプションをハイライトすると、そのオプションの説明が e^2 studio GUI の左下隅に表示されます。構成オプションの入力フィールドが空の場合は、デフォルト値が使用されます。詳細については、『ThreadX ユーザーガイド』の構成オプションの章を参照してください。

高度なプロパティの最後に、TX_<COMPONENT>_EXTENSION マクロがあります。ここで、<COMPONENT> は拡張のタイプを示しています。これらの拡張マクロは、高度なユースケースでのみ使用します。ほとんどのプロジェクトで、これらのマクロは変更されません。NetX の BSD サポートなど、場合によっては、拡張マクロの使用が必要になります。その場合は、拡張マクロの使用方法を明示的に記述します。

- ARM CM23 および CM0+ MCU では、ハードウェアスタックのモニタリングはサポートされていません。ThreadX のソースがこれらの MCU とともに使用されている場合は、SSP コンフィギュレータを使用して [Enable Hardware Thread Stack Monitoring] プロパティを [Disabled] に設定します。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.1.3 アプリケーションでの Express Logic 社の ThreadX モジュールの使用

プロジェクトの構成時に、新しいスレッドを作成すると、ThreadX は自動的に追加されます。

ThreadX オブジェクト（ミューテックス、セマフォ、イベントフラグ、キューなど）は、< スレッド名 > オブジェクトのセクションにある [Threads] に追加できます。ここで < スレッド名 > は、アプリケーションスレッドの名前を表します。たとえば、スレッドに新しいキューを追加するには、スレッドをハイライトし、[New Object]>[Queue] の順に選択します。

別のスレッドからキューにアクセスするには、そのキューが定義されているスレッドのヘッダーファイルを含めます。たとえば、アプリケーションに main_thread という名前のスレッドがあり、キューの名前が message_queue, である場合は、main_thread.h という名前のファイルが生成され、message_queue. の extern が宣言されます。また、アプリケーションに background_thread, という名前のスレッドがあり、background_thread が message_queue, にアクセスする必要がある場合は、main_thread.h を background_thread_entry.c に含め、background_thread_entry.c が message_queue. の定義にアクセスできるようにする必要があります。

4.3.2 FileX ポート ブロック メディア フレームワーク

FileX ポートブロックメディアフレームワークモジュールは、Express Logic 社の FileX システムをサポートします。FileX は、ディープエンベデッドアプリケーション用の完全な FAT フォーマットメディアであり、ファイル管理システムでもあります。

4.3.2.1 FileX ポートブロックメディアフレームワークモジュールの特長

- FAT32、FAT16、FAT12 をサポート
- 特定のメディアでの最初の FAT パーティションのマウントをサポート
- 無制限の FileX オブジェクト（メディア、ディレクトリ、ファイル）
- 動的 FileX オブジェクト作成と削除
- 柔軟なメモリ使用
- サイズを自動的に拡張
- 小さなフットプリント
- ThreadX との完全な統合

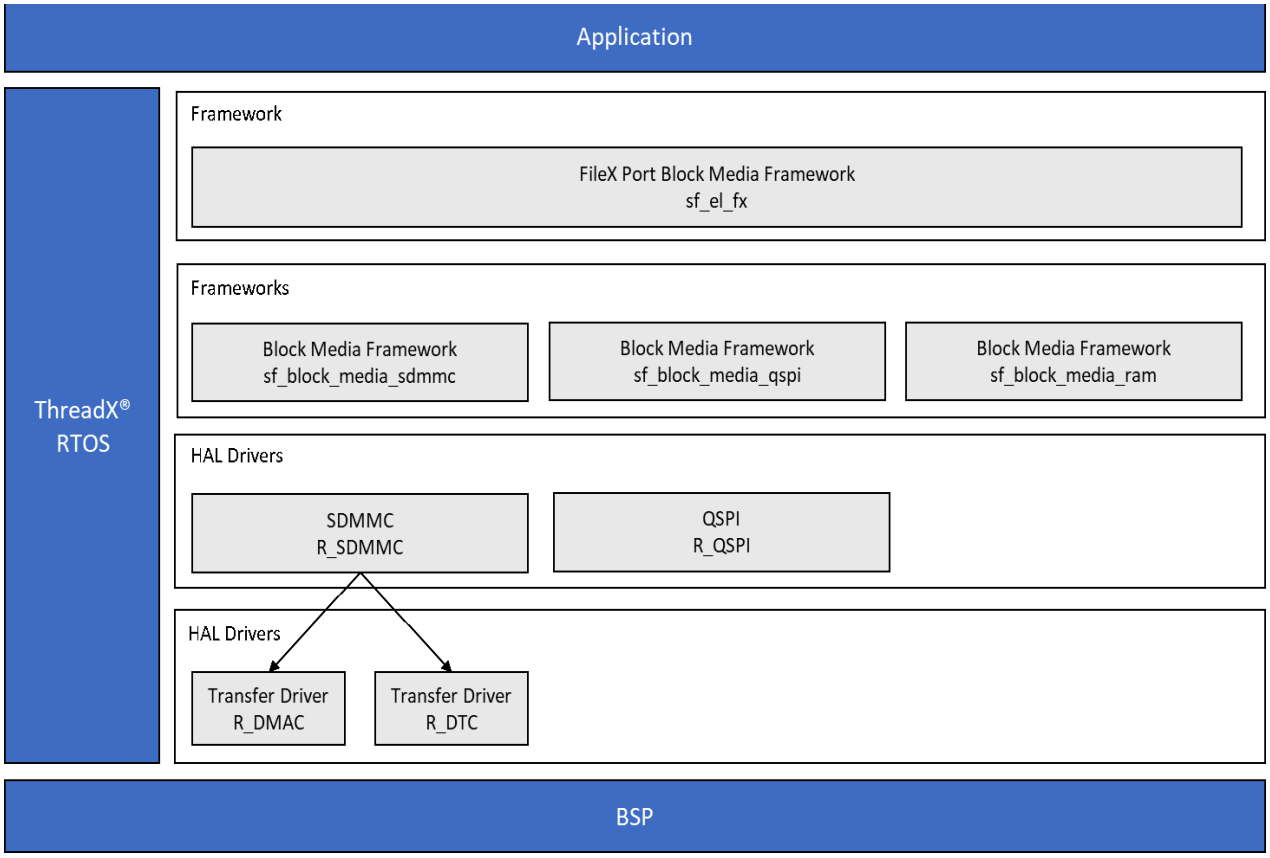


図 302:FileX ポートブロックメディアフレームワークモジュールのブロック図

4.3.2.2 FileX ポートブロックメディアフレームワークモジュールの API の概要

FileX ポートブロックメディアフレームワークは、オープン、クローズ、設定、初期化の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

FileX ポートブロックメディアフレームワークモジュールの API の要約（選択した例のみ）

Function Name	Example API Call and Description
fx_directory_attributes_read	fx_directory_attributes_read(&my_media, "mydir", &attributes); Reads the directory's attributes from the specified media.

Function Name	Example API Call and Description
fx_file_open	fx_file_open(&my_media, &my_file, "myfile.txt", FX_OPEN_FOR_READ); Open “myfile.txt” file for read.
fx_file_create	fx_file_create(&my_media, "myfile.txt"); Create file with name “myfile.txt”
fx_media_read	fx_media_read(&my_media, 22, my_buffer); Read the logic sector (in this example from sector 22) from media specified by &my_media, and places it into the buffer my_buffer.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.3.2.3 FileX ポートブロックメディアフレームワークモジュールの動作の概要

FileX は、ディープ エンベデッド アプリケーション用の完全なファイル アロケーション テーブル (FAT) フォーマット メディアであり、ファイル管理システムでもあります。FileX は以下のメディア デバイスをサポートします。同時にサポートできるデバイスの数に制限はありません。

- RAM ディスク
- FLASH マネージャー
- その他複数の物理デバイス

FileX は、12 ビット、16 ビット、および 32 ビットの FAT フォーマットと連続的なファイル割り当てをサポートします。FileX はサイズと性能の両面で高度に最適化されています。

FileX ポートブロックメディアフレームワークモジュールの動作に関する重要な注意事項と制限事項

- メディアは、開く前に FAT12、FAT16、FAT32 のいずれかのファイルシステムにフォーマットする必要があります。この操作は、メディアの挿入前または実行時に行うことができます。
- SF_EL_FX では現在 exFAT がサポートされていません。
- SF_EL_FX を SD または eMMC とともに使用する場合は、FileX ブロックサイズを 512 バイトにする必要があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.2.4 アプリケーションへの FileX ポートブロックメディアフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに FileX ポートブロックメディアフレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

FileX ポートブロックメディアフレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

FileX ポートブロックメディアフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_el_fx0 FileX Port Block Media Framework on sf_el_fx	Threads	New Stack> Framework > File System> FileX Port Block Media Framework on sf_el_fx

次の図に示すように、FileX ポートブロックメディアフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

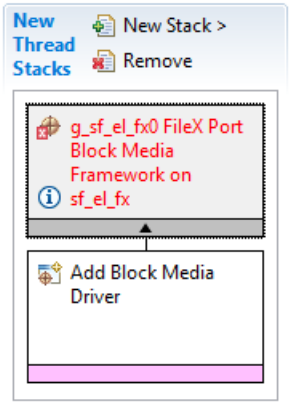


図 303:FileX ポートブロックメディアフレームワークモジュールのスタック

4.3.2.5 FileX ポートブロックメディアフレームワークモジュールの構成

ユーザーは必要な動作に合わせて FileX ポートブロックメディアフレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_el_fx の FileX ポートブロックメディアフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_el_fx0	Module name.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

FileX ポートブロックメディアフレームワークのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

sf_block_media_sdmmc でのブロックメディアフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_block_media_sdmmc0	Module name.

ISDE Property	Value	Description
Block size of media in bytes	512	Specify the size of a block in bytes.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_sdmmc 上の SDMMC HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.
Name	g_sdmmc0	The name to be used for SDMMC module control block instance. This name is also used as the prefix of the other variable instances.
Channel	0	Select the channel.
Media Type	Embedded, Card Default: Embedded	Media is a card or an embedded device. This allows to firmware to know whether to look for card insertion/removal and write protect pins.
Bus Width	1 Bit, 4 Bits, 8 Bits Default: 4 Bits	Data bus with as defined by hardware interface. (8 Bits for eMMC only)
Block Size	512	Block size selection.
Card Detection	Not Used, CD Pin Default: CD Pin	Card detection selection.

ISDE Property	Value	Description
Callback	NULL	(Not required if using Filex) Set to name of user callback function. Provides event that caused interrupt: SDMMC_EVENT_CARD_REMOVED, SDMMC_EVENT_CARD_INSERTED, SDMMC_EVENT_ACCESS, SDMMC_EVENT_SDIO, SDMMC_EVENT_TRANSFER_COMPLETE, SDMMC_EVENT_TRANSFER_ERROR
Access Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Access interrupt priority selection.
Card Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Card interrupt priority selection.
DTC Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	DTC interrupt priority selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dmac 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Name	g_transfer0	Module name
Channel	0	Select the channel
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation	Activation source selection
Auto Enable	False	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection

ISDE Property	Value	Description
Transmit End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Transmit end interrupt priority selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section selection.
Name	g_transfer0	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection

ISDE Property	Value	Description
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation 1	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	ELC Software Event interrupt priority selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_block_media_qspi のブロックメディアフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_block_media_qspi	Module name.
Block size of media in bytes	4096	Block size selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_qspi の QSPI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_qspi0	Module name.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_block_media_ram のブロックメディアフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_block_media_ram0	Module name.
Block size of media in bytes	512	Block size of media in bytes selection
Number of blocks to allocate	16	To make use of file system, block count must be assigned by considering the boot record, FAT area and root directory.
Enter the valid section for RAM buffer allocation	noinit	Enter the valid section for RAM buffer allocation

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

FileX ポートブロックメディアフレームワークモジュールのクロック構成

SDHI は PCLKA をそのクロックソースとして使用します。データレートを最適化する必要がない限り、SDMMC ペリフェラルに対して固有のクロックを設定する必要はありません。SDMMC ドライバーは、PCLKA 周波数と、SD、SDIO、または eMMC デバイスで許可される最大クロックレート（これはメディアデバイスの初期化時に取得されます）に基づいて、適切な内蔵分周器を選択します。

FileX ポートブロックメディアフレームワークモジュールのピン構成

e^2 studio ピン コンフィギュレーターを使用して、SDMMC 周辺機器の I/O ピンを設定します。ほとんどのボード、高速メモリ、SDIO デバイスにおいて、各品のドライブ能力は「中」または「高」に設定する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表はモジュールピンの選択例を示しています。

注 : 動作モードの選択によって、使用可能なペリフェラル信号が決まり、それに従って必要な MCU ピンが決定されます。

SDHI ペリフェラルのピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
SDHI	Pins	Select Peripherals > Storage:SHDI > SDHI0

注 : 選択シーケンスでは、SDHI0 がドライバーに必要なハードウェアターゲットであることを想定しています。

SDHI ペリフェラルのピン構成設定

Pin Configuration Property	Settings	Description
Operation Mode	Disabled, Custom, SD_MMC 1 bit SD_MMC 4 bit MMC 8 bit Default: Custom	Select mode as per application
CLK	None, P413 Default: P413	Clock Pin
CMD	None, P412 Default: P412	Command Pin
DAT0-7	None, PXXX Default: PXXX	Data Pin

Pin Configuration Property	Settings	Description
CD	None, P903 Default: P903	Card detection Pin
WP	None, P414 Default: P414	Card write protection pin

注 : 例の値は、Synergy S7G2 および DK-S7G2 キットを使用するプロジェクトを想定したものです。他の Synergy キットと他の Synergy MCU では使用可能なピン構成設定が異なる可能性があります。

4.3.2.6 アプリケーションでの FileX ポートブロックメディアフレームワークモジュールの使用

一般的なアプリケーションで FileX ポートブロックメディアフレームワークモジュールを使用する際の手順は次のとおりです。

1) FileX API fx_system_initialize を使用してメディアを初期化します（sf_el_fx は [FileX Common on fx] で [Auto Initialization] プロパティが [Enabled] に設定されている場合にこれを自動的にコールします）
2) オプションで、FileX API fx_media_format を使用してメディアをフォーマットします
   a) [Format media during initialization] プロパティが [Enabled] に設定されている場合、sf_el_fx は生成された初期化関数中にメディアを自動的にフォーマットします)。
   b) [Filesystem on SDMMC] が有効な場合は、隠しセクターの後のメディア全体がフォーマットされ、[FileX on Block Media] の [Total Sectors] 構成は無視されます。
3) FileX API fx_media_open を使用してメディアを開きます（sf_el_fx は [FileX on Block Media] インスタンスで [Auto Initialization] が [Enabled] に設定されている場合にメディアを自動的に開きます）
4) FileX API の 1 つ（たとえば、fx_file_create,、fx_file_delete,、fx_directory_create,、fx_directory_delete）を使用して、必要に応じてファイルとディレクトリを作成および削除します
5) FileX API の 1 つ（たとえば、fx_file_read や fx_file_write）を使用して、必要に応じてメディア上のファイルに対してリードやライトを行います
6) FileX API の 1 つ（たとえば、fx_media_read や fx_media_write）を使用して、必要に応じてメディアに対して直接リードやライトを行います
7) FileX API fx_media_close を使用して物理メディアを閉じます。

注 :fx_media_open 呼び出しに成功した後は、すべての FileX ファイルおよびディレクトリ関連 API を使用できます。すべての使用可能な関数については、『FileX ユーザーガイド』を参照してください。

これらの一般的な手順を、次の図の通常の動作フローに示します。

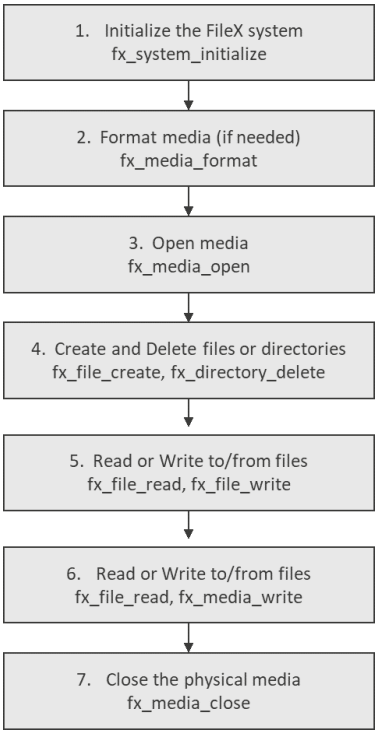


図 304:一般的な FileX ポートブロックメディアフレームワークモジュールアプリケーションのフロー図

4.3.3 FileX ソースコンポーネントモジュール

4.3.3.1 FileX ソースコンポーネントモジュールの概要

このドキュメントの目的は、e^2 studio の FileX ソースコンポーネントに関する簡単なリファレンスを提供することです。プロパティについて、各プロパティで提供されるフッターコメントよりもかなり詳細に説明しています。デフォルト値の変更がどのような場合に必要であるかを理解しやすくするため、コンテキスト固有の使用方法が記載されています。このドキュメントを使えば、Express Logic 社の『FileX ユーザーガイド』を参照することなく、より容易に FileX ソースコンポーネントを使用でき、開発者は FileX の機能をより素早く習得することができます。

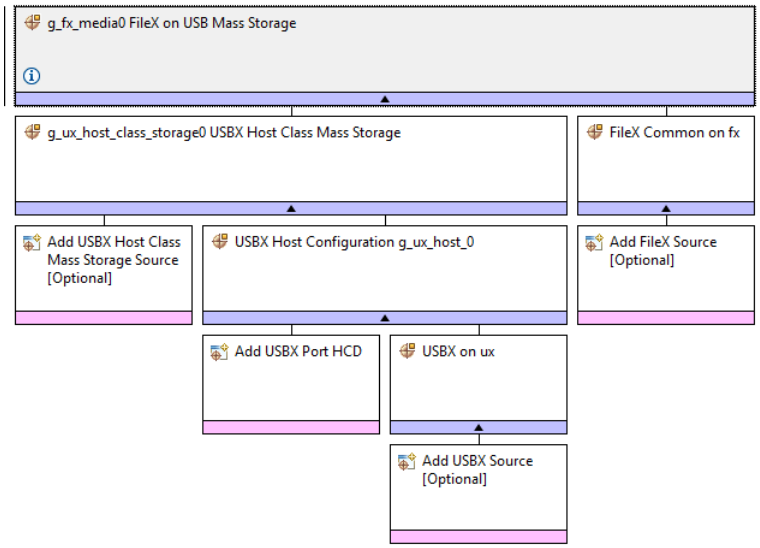
4.3.3.2 どのような場合に FileX ソースコンポーネントを含めるか

Synergy コンフィギュレータ環境を使用する開発者は、FileX ソースコンポーネントを追加することで、FileX ライブラリをカスタマイズしたり、デフォルト設定から値を変更したり、特定の機能を有効化または無効化したりできるようになります。FileX ソースコンポーネントを追加しない場合は、ビルド済みの FileX ライブラリを使用する必要があります。非常にシンプルなものを除くほとんどのプロジェクトでは、通常、開発者が FileX 環境をカスタマイズする必要があります。ThreadX ソースコンポーネントは、よりハイレベルなソースコンポーネント（FileX、NetX、NetX Duo、GUIX、USBX など）を追加するときに自動的に追加されることに留意してください。

FileX ソースコンポーネントを追加しない場合、e^2 studio コンフィギュレータは、FileX のデフォルト設定を使用して事前にビルドされたライブラリを使用します。

4.3.3.3 FileX ソースコンポーネントの追加

e2 studio コンフィギュレータで、[Threads] リストから任意のスレッドを選択し、[New Stack] ボタンを押してメニューから [X-Ware] > [FileX] > [FileX Source] を選択することで、ThreadX ソースコンポーネントを追加します。多くの場合、ハイレベルのフレームワークを作成するときにオプションとして FileX ソースを使用できます。たとえば、下のスレッドスタックの図のように、USB 大容量記憶装置上の FileX フレームワークモジュールでは、オプションとして FileX ソースモジュールを使用できます。



4.3.3.4 FileX ソースコンポーネントプロパティの変更

FileX プロパティの設定を変更した後、開発者は [Generate Project Content] ボタンをクリックして、e2 studio のプロジェクトコンフィギュレータを更新し、FileX ライブラリを再ビルド（つまり、プロジェクトを再ビルド）する**必要があります**。プロジェクトを再ビルドせず単にプロパティを変更（またはプリプロセッサリストで #define を適用）しても、e2 studio は以前にビルドされたライブラリを使用するため、いずれの変更にも影響しません。

多くの場合、デフォルト設定はほとんどの一般的なユースケースで必要とされる設定です。

4.3.3.5 FileX ソース

FileX ソースコンポーネントのプロパティについて、Synergy コンフィギュレータの [Properties] ウィンドウに表示される順序で説明します。

[Error Checking] – デフォルト値は有効 – 一般に、開発およびデバッグフェーズでは有効にし、リリースバージョンをビルドするときは無効にします。このプロパティを有効にすると、FileX によって実際の API をコールする前に入力およびその他のパラメータをチェックするエラーチェックサービスが含められます。チェック対象の例を以下に示します。

- NULL ポインタ入力
- 無効なファイル名やディレクトリ名など無効な非ポインタパラメータ。
- 必須の設定可能オプションが有効になっていること。たとえば、取得サービスをコールするにはパフォーマンス情報を有効にする必要があります。
- データ構造体 ID が期待される値と一致していること。例：examplefile_ptr -> fx_file_id != FX_FILE_ID /* ファイルインスタンスの構造体を確認

- データ構造体のサイズ、たとえば FileX ファイルのサイズが FileX ライブラリのデータ構造体のサイズと一致していること。

最後の 2 つのチェックは、アプリケーションが使用しているバージョンとは別のバージョンの FileX ライブラリが使用されていることを検出するために行われます。

FileX のエラーチェック API を無効にすると、パフォーマンスが（30% 程度）向上し、コードサイズが小さくなります。

[Max Long Name Len]（Maximum Long Name Length）– デフォルト値は非表示、33 文字を使用 – FileX でサポートされるロングファイルネームの最大サイズを定義します。デフォルト値は 33 です。最小値は 13、最大値は 256 です。

[Max Last Name Len]（Maximum Last Name Length）– デフォルト値は非表示、256 文字を使用 – FileX でサポートされる最後に開いたファイル名の最大サイズを定義します。デフォルト値は 256 です。最小値は 13、最大値は 256 です。[Max Long Name Len] 以上のサイズにする必要があります。

[Max Sector Cache]（Maximum Sector Cache Size）- デフォルト値は非表示、256 を使用 – FileX でキャッシュ可能な論理セクターの最大数を定義します。実際にキャッシュできるセクターの数は、fx_media_open で FileX に供給されるキャッシュメモリによって決まります。最小値は 2 です。その他の値はすべて 2 の累乗にする必要があります。

[Fat Map Size]（FAT Map Size）- デフォルト値は非表示、128 を使用 – セカンダリ FAT セクターの更新に使用されるビットマップのサイズをバイト単位で定義します。値が大きいほど、不要なセカンダリ FAT セクターのライトは少なくなります。最小値は 1 で、最大値はありません。

[Max Fat Cache]（Maximum FAT Cache）- デフォルト値は非表示、16 を使用 – 内部 FAT キャッシュ内のエントリ数を定義します。最小値は 8 です。すべての値を 2 の累乗にする必要があります。

[Update Rate (seconds)] - デフォルト値は非表示、10 秒を使用 - FileX のシステム時刻を調整するレートを指定します。デフォルト値の 10 は、FileX のシステム時刻が 10 秒ごとに更新されることを意味します。

[No Timer] – デフォルト値は無効 – 有効にすると、FileX は時間パラメータを更新せずにビルドされます。ThreadX タイマのセットアップを無視して FileX システムの時刻と日付を更新します。これにより、すべてのファイル操作にデフォルトの時刻と日付が適用されます。

[Single Thread] – デフォルト値は無効 – 有効にすると、FileX はシングルスレッド環境で実行され、スレッド保護が不要になります。FileX ソースから ThreadX 保護ロジックを削除します。FileX が 1 つのスレッドからのみ使用されている場合、または FileX が ThreadX なしで使用されている場合のみ、有効にすることをお勧めします。

[Don't Update Open Files] – デフォルト値は無効 – 有効にすると、FileX は既に開かれているファイルを更新しません。

[Media Search Cache] – デフォルト値は有効 – 有効にすると、開いているメディアを検索するときに、最適化のためにキャッシュが使用されます。このオプションを無効にするとこの最適化は削除され、コードサイズとメモリフットプリントが削減されますがパフォーマンスは低下します。

[Direct Data Read Cache Fill] – デフォルト値は有効 – 有効にすると、アクセスを高速化するためにデータセクターのリードがキャッシュされます。この機能を無効にすると、コードサイズとメモリフットプリントが削減されますがパフォーマンスは低下します。

[Media Statistics] – デフォルト値は有効 – メディアの統計情報を保持および収集するかどうかを決定します。無効にすると、メディア統計情報は使用できません。これによりパフォーマンスがわずかに向上し、コードサイズが小さくなります。FX_MEDIA_STRUCT 構造体の各インスタンスはかなり小さくなります。

[Single Open Legacy] – デフォルト値は無効 – 有効にすると、同じファイルにレガシーの単一オープンロジックが使用されます。これは、古い FileX アプリケーションコードを移行する場合に、古いバージョンの FileX と同じ方法で fx_file_open を動作させるために必要な場合があります。

[Rename Path Inherit] – デフォルト値は無効 – 有効にすると名前変更でパス情報が継承されます。つまり、新しいファイル名の前にパスを付加し、名前を変更するファイルの古いファイル名を新しいファイル名で上書きします。

[No Local Path] – **デフォルト値は無効** – 有効にするとローカルパスロジックが無効になります。無効にしていると、各スレッドのローカルパスが保持され、相対パスで実行されるすべての操作は、このローカルパスを基準にして行われます。

[Fault Tolerant Data] – **デフォルト値は無効** – 有効にすると、データセクターのライト要求がドライバーに即座にフラッシュされます。これにより、停電やリセット時にデータの損失を防げる可能性が高まりますが、パフォーマンスに影響します。

[Fault Tolerant] – **デフォルト値は無効** – 有効にすると、システムセクターのライト要求（FAT およびディレクトリエントリ要求を含む）がドライバーに即座にフラッシュされます。これにより、停電やリセット時にデータの損失を防げる可能性が高まりますが、パフォーマンスに影響します。

[64-bit LBA] – **デフォルト値は有効** – 有効にすると、I/O ドライバーで 64 ビットセクターアドレスが使用されます。これにより、より大きなメディアおよびより大きなファイルが可能になります。

[Fault Tolerant Service] – **デフォルト値は無効** – FileX フォールトトレラントサービスを有効化または無効化します。FileX フォールトトレラントモジュールは、ファイルまたはディレクトリ更新時の割り込みによるファイルシステム破損を防ぐよう設計されています。たとえば、ファイルにデータを追加するとき、FileX はファイルの内容、ディレクトリエントリ、さらに場合によっては FAT エントリを更新する必要があります。この更新シーケンスが中断された場合（電源障害や更新中のメディアの取り出しなど）、ファイルシステムが矛盾した状態になり、ファイルシステム全体の整合性が損なわれて、その他のファイルが破損する可能性があります。

[Fault Tolerant Boot Index] – **デフォルトは 116** – フォールトトレラントログ用のクラスタがあるブートセクターのバイトオフセットを定義します。デフォルトでは、この値は 116 です。このフィールドは 4 バイトを占有します。FAT 12/16/32/exFAT 仕様で予約済みとしてマークされているため、116 ～ 119 のバイトを選択します。

[Fault Tolerant Minimal Cluster Size] – **デフォルトは 3072** – クラスタの最小サイズの要件をバイト単位で定義します。この値はセクターサイズの倍数である必要があります。デフォルト値は 3072 です。これはロングファイルネームの名前変更の最悪のケースでも機能します。

4.3.3.6 FileX フォールトトレラントモジュール

アプリケーションがデータをファイルに書き込むときに、FileX はデータクラスタとシステム情報の両方を更新します。これらの更新は、ファイルシステム内の情報の一貫性を維持するためのアトミック操作として完了する必要があります。たとえば、ファイルにデータを追加するとき、FileX はメディア内で使用可能なクラスタを見つけて、FAT チェーンを更新し、ディレクトリエントリに記録されている長さを更新する必要があります。また場合によってはディレクトリエントリ内の開始クラスタ番号も更新する必要があります。更新シーケンスは電源障害またはメディアの取り出しによって中断される可能性があり、この場合ファイルシステムが矛盾した状態になります。矛盾した状態を修正しないと、更新中のデータが失われることがあります。また、システム情報が損傷するため、その後のファイルシステムの動作によって、メディア上の他のファイルやディレクトリが損傷する可能性があります。

FileX フォールトトレラントモジュールの動作は、ファイルを更新するために必要な手順をジャーナリングしてからファイルシステムに適用するという流れになっています。ファイルの更新が成功すると、これらのログエントリは削除されます。ファイルの更新が中断された場合、ログエントリはメディアに保存されます。メディアが次にマウントされるときに、FileX は以前の（未完了の）ライト動作からこれらのログエントリを検出します。このような場合、FileX は、ファイルシステムに既に加えられた変更をロールバックするか、必要な変更を再度適用して前の操作を完了することによって、障害から復旧することができます。このように、更新動作中にメディアの電源が失われた場合でも、FileX フォールトトレラントモジュールはファイルシステムの整合性を維持します。

注 :FileX フォールトトレラントモジュールは、有効なデータが格納されている物理メディアの破損によるファイルシステムの破損を防ぐよう設計されてはいません。

FileX フォールトトレラントモジュールでメディアを保護した後、フォールトトレラントが有効な FileX 以外でこのメディアをマウントしてはなりません。そうすると、ファイルシステムのログエントリがメディア上のシステム情報と矛盾する可能性があります。別のファイルシステムによってメディアが更新された後、FileX フォールトトレラントモジュールがログエントリを処理しようとすると、復帰手順が失敗し、ファイルシステム全体が予期しない状態になる場合があります。

FileX フォールトトレラント機能は、FAT12、FAT16、FAT32、exFAT を含む、FileX でサポートされているすべての FAT ファイルシステムで使用できます。フォールトトレラント機能を有効にするには、[Fault tolerant service] オプションを有効にして FileX をビルドする必要があります。実行時に、アプリケーションは、fx_media_open. へのコールの後、すぐに fx_fault_tolerant_enable() をコールすることでフォールトトレラントサービスを開始します。フォールトトレラントを有効にした後は、指定されたメディアに対するすべてのファイルライト動作が保護されます。デフォルトでは、フォールトトレラントモジュールは無効です。

アプリケーションは、fx_fault_tolerant_enable() がコールされる前に、ファイルシステムがアクセスされていないことを確認する必要があります。フォールトトレラントが有効になる前にアプリケーションがデータをファイルシステムに書き込むと、ライト動作によってメディア（以前のライト動作が完了していなかった場合）およびファイルシステムが破損する可能性があります。

FileX フォールトトレラントログは、フラッシュ内で 1 つの論理クラスタを占有します。そのクラスタの開始クラスタ番号のインデックスは、フォールトトレラントログエントリを使用して復元されなかったブートセクターに記録されています。ログフォーマットの詳細については、『FileX ユーザーガイド』を参照してください。

4.3.3.7 exFAT サポートについて

exFAT ファイルシステムフォーマットは Microsoft 社が特許を保有しており、その使用には特別なライセンスが必要です。exFAT 対応の FileX バージョンのライセンス取得およびこれに対するアクセスの詳細については、Renesas の担当者にお問い合わせください。

4.3.4 GUIX Synergy ポートフレームワーク

SF_EL_GX（GUIX ポート）モジュールは、Synergy MCU グループ向けに開発された Express Logic 社の GUIX 適応レイヤーであり、GLCDC、DRW（2DG エンジン）、または JPEG デコードエンジンを搭載しています。この API では、GUIX 用のグラフィックハードウェアエンジン設定、グラフィックスのレンダリング、およびハードウェアエンジンによって加速される表示をサポートします。このモジュールでは、GUIX ローレベルディスプレイドライバー機能のフルセットを定義します。DRW（2DG エンジン）または JPEG によって加速されるグラフィックスを描画したり、GLCDC を使用してグラフィックスを表示したりします（『GUIX ユーザーガイド』の第 5 章「GUIX ディスプレイドライバー」を参照）。このモジュールは、グラフィックスのレンダリングのためにハードウェア加速を促進するだけでなく、ハードウェアのサポートなしでソフトウェア処理も可能にします。

4.3.4.1 GUIX Synergy ポートフレームワークモジュールの特長

- GUIX を SSP の一番上に適応する
- SSP ディスプレイインタフェースドライバーを GUIX ディスプレイドライバーインタフェースに設置する
- Synergy D2W（2DG）エンジンにより、GUIX が高速でウィジェットを描画できるようにする
- Synergy JPEG エンジンにより、GUIX が高速でウィジェットを描画できるようにする
- 切れ目が発生することなく画面の移行を可能とする、ダブルバッファ切り替え制御をサポートする
- 画面のローテーションをサポートする（90/180/270 度）
- 各種の色出力形式をサポートする
  - 32bpp（ARGB8888、RGB-888）
  - 16bpp（RGB565）
  - 8bpp（8 ビットパレット（CLUT））

- ユーザー コールバック関数をサポートする

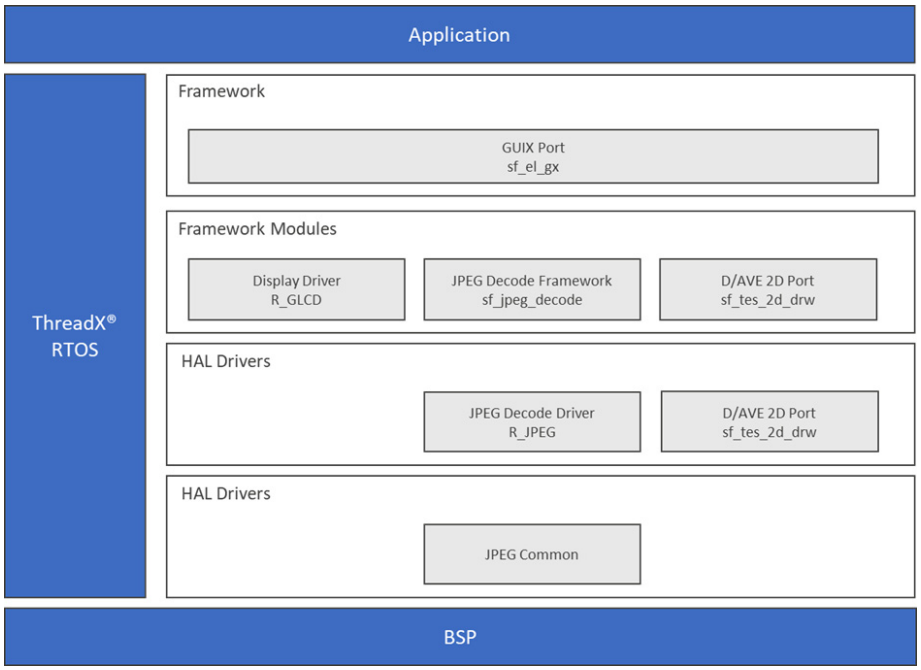


図 305:**GUIX Synergy ポートフレームワークモジュールのブロック図**

4.3.4.2 GUIX Synergy ポートフレームワークモジュール API の概要

GUIX Synergy ポートフレームワークは、オープン、クローズ、設定、初期化の API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

GUIX Synergy ポートフレームワークモジュール API の要約

Function Name	Example API Call and Description
open	g_sf_el_gx.p_api->open(g_sf_el_gx.p_ctrl, g_sf_el_gx.p_cfg); Opens the SF_EL_GX Module. The API can only be called from a thread. The API passes the configuration pointer to define low-level graphics device drivers and frame buffers and register the user callback function. This function does not actually initialize low-level drivers. Instead, the API setup initializes the low-level drivers.

Function Name	Example API Call and Description
close	g_sf_el_gx.p_api->close(g_sf_el_gx.p_ctrl); Closes the SF_EL_GX Module. This API closes the low-level drivers. Normally, the API is not called since GUIX will not be closed once initialized.
versionGet	g_sf_el_gx.p_api->versionGet(&version); Returns the version of the Module in the version pointer.
setup	gx_studio_display_configure (MAIN_DISPLAY, g_sf_el_gx.p_api->setup, LANGUAGE_ENGLISH, MAIN_DISPLAY_THEME_1, &p_window_root); This is the interface to initialize low-level graphics device drivers and must be passed to GUIX through GUIX (Studio) service call gx_studio_display_configure() as the function pointer. GUIX then calls the API back and, at that moment, the API configures the SSP device drivers based on the configuration passed by open. The reason for this procedure to initialize low-level drivers is that the API has the GUIX-compliant argument (GX_DISPLAY *) type and does not allow applying the detailed configuration of the SSP graphics device drivers generated from e^2 studio.
canvasInit	g_sf_el_gx.p_api->canvasInit(g_sf_el_gx.p_ctrl, p_window_root); This is the GUIX helper API to determine the memory address of GUIX canvas. The API has an argument with (GX_WINDOW_ROOT *) type and the API provides GUIX the start address of canvas memory, which is needed for the low-level graphics device drivers to draw/display images.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API call successful.
SSP_ERR_ASSERTION	NULL pointer error happens.
SSP_ERR_IN_USE	SF_EL_GX is in-use.
SSP_ERR_INTERNAL	Error happen in Kernel service calls.
SSP_ERR_NOT_OPEN	SF_EL_GX is not opened.
SSP_ERR_TIMEOUT	A task times out (or exceeds retry limit) before completion in display driver.
SSP_ERR_D2D_ERROR_DEINT	Error occurred in D/AVE 2D driver.
GX_SUCCESS	Device driver setup is successfully done.
GX_FAILURE	Device driver setup failed.
SSP_ERR_INVALID_CALL	Function call was made when the driver is not in SF_EL_GX_CONFIGURED state.
SSP_ERR_D2D_RENDERING	The D/AVE 2D returns error at opening a display list buffer
SSP_ERR_INVALID_ARGUEMENT	Invalid non-pointer (e.g. parameter) input
SSP_ERR_UNSUPPORTED	Specified color format is not supported
SSP_ERR_D2D_ERROR_INIT	The D/AVE 2D returns error at the initialization.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.3.4.3 GUIX Synergy ポートフレームワークモジュールの動作の概要

次のブロック図は、sf_el_gx モジュールが、動作レベルで他の Synergy コンポーネントとどのようにやり取りするかを示しています。また、コールバックおよびローレベルモジュールがシステムの他の部分と通信するレイヤーを示しています。この図を参照する際は、以下に記載する動作の概要と動作に関する注意事項の説明を読んでください。

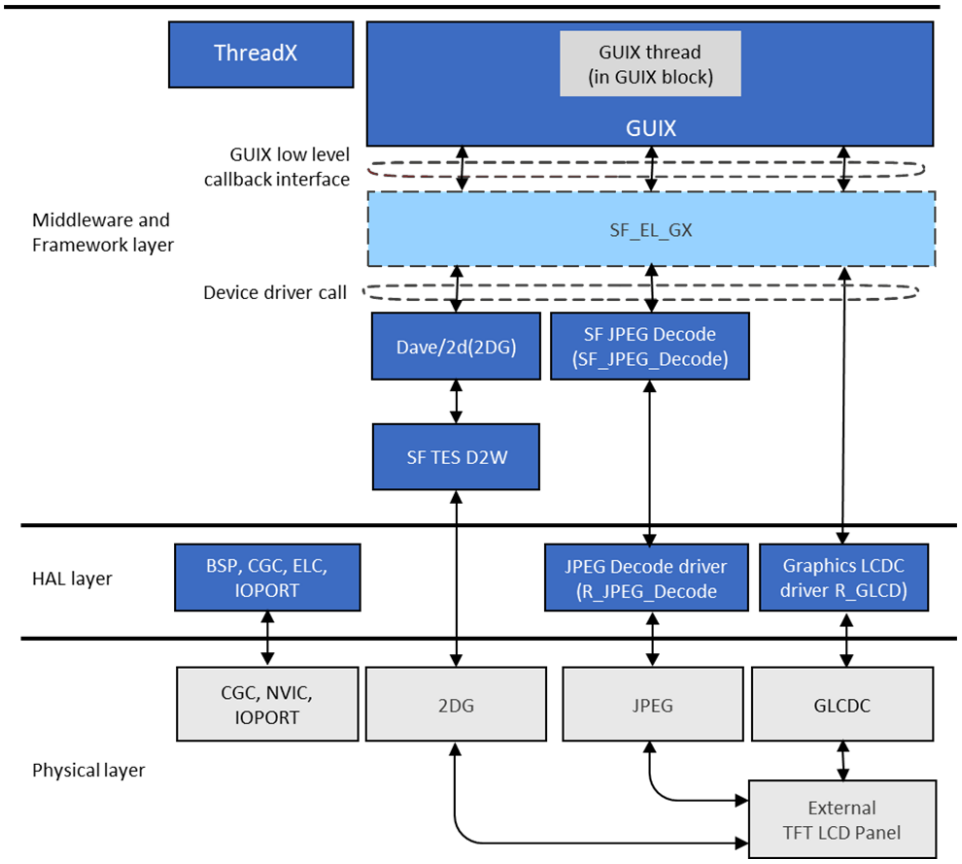


図 306:GUIX Synergy ポートフレームワークモジュールのフローチャート

モジュールの初期化

SF_EL_GX は、GUIX システムの実行に必要な Synergy グラフィックスハードウェア設定をサポートします。モジュールは、Express Logic GUIX™ および GUIX Studio™ で生成されたコードに依存します。GUIX システムの初期化では、一般的なガイダンスとして以下のシーケンスに従う必要があります。

1) SF_EL_GX モジュールの 'open' を実行して SF_EL_UX コントロールブロックを初期化し、モジュール構成を渡します。
2) GUIX Studio で生成された API gx_studio_display_configure. で GUIX ディスプレイオブジェクトを初期化します。この API を通じて、SF_EL_GX 'setup' API が GUIX に入力され、Synergy グラフィックスハードウェア設定が完了します。また、GUIX で初期化されるルートウィンドウは、ユーザーアプリケーションに出力されます。
3) canvasInit API で GUIX キャンバスバッファのプライマリメモリアドレスを初期化します。
4) GUIX Studio で生成された API gx_studio_named_widget_create. でルートウィンドウを作成します。
5) GUIX API gx_window_show API でルートウィンドウを表示します。
6) GUIX API gx_system_start. で GUIX システムを起動します。

ピンポン フレーム バッファ管理

SF_EL_GX モジュールは、ピンポンフレームバッファを備えたグラフィックスシステムでバッファ切り替え動作を管理します。下図に、SF_EL_GX モジュールによって管理されるピンポンバッファグラフィックスシステムを示します。このモジュールは、GUIX とローレベルディスプレイドライバーの機能を使用してイメージの描画（2D 描画エンジン（engine(DRW）または JPEG）とイメージの表示（ディスプレイモジュール、たとえば GLCDC など）を行います。SF_EL_GX 構成では、単一のフレームバッファを持つ設計も可能です。しかし、2 つのフレームバッファを使用して、単一フレームバッファシステムで起こり得る切れ目の問題を回避することをお勧めします。

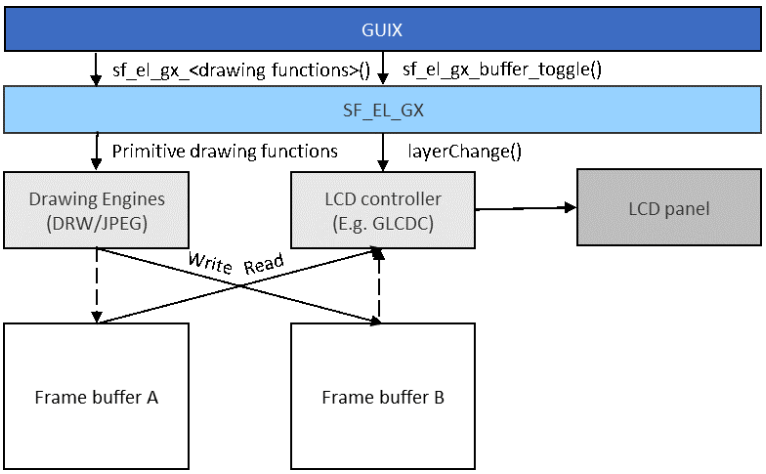


図 307:GUIX Synergy ポートフレームワークのピンポンフレームバッファシステム

GUIX Synergy ポートフレームワークモジュールの動作に関する重要な注意事項と制限事項

Synergy 2D 描画エンジンのサポート

モジュールは、2D 描画エンジン（DRW）で加速されるグラフィックスイメージを描画して、グラフィックス性能を向上させることができます。ユーザーは、以下の構成で 2D 描画エンジンを有効化できます。この構成は、Synergy コンフィギュレータを通じて利用できます。

- gx_user.h で GX_USE_SYNEGY_DRW（1）を定義する
- DRW (SF_TES_2D_DRW) 割り込み優先順位を設定する

Express Logic GUIX Studio (v5.2.8 以降) の [プロジェクト構成] ウィンドウから、[ターゲット CPU] 設定で [Renesas Synergy] が選択され、[Synergy 詳細設定 (Synergy Advanced Settings)] ウィンドウの [グラフィックス アクセラレーターを有効にする (Enable Graphics Accelerator)] がチェックされていることを確認してください。[Edit Pixel map] ウィンドウの Pixelmap 出力フォーマットについては（[pixelmap] -> [edit settings] を右クリックするとウィンドウが開きます）、[Compress Output] を選択してください。[非圧縮フォーマット] を選択しないでください。このように設定することで、GUIX Studio により Targa RLE フォーマットでエンコードされた画像リソース データが生成されます。2D 描画エンジンハードウェアはこのフォーマットを読み取れるうえに、フレームバッファに画像のデコードと描画を実行できます。

Synergy JPEG のサポート

JPEG でエンコードされたイメージが GUIX イメージリソースとして使用されている場合、モジュールは、JPEG エンジンで加速されるグラフィックスイメージを描画して、グラフィックス性能を向上させることができます。ユーザーは、以下の構成で JPEG エンジンを有効化できます。この構成は、Synergy コンフィギュレータを通じて利用できます。

- gx_user.h で GX_USE_SYNEGY_JPEG（1）を定義する
- JPEG (R_JPEG_DECODE) 割り込み優先順位を設定する

Express Logic GUIX Studio (v5.2.8 以降) の [プロジェクト構成] ウィンドウから、[ターゲット CPU] 設定で [Renesas Synergy] が選択され、[デコーダ タイプ JPEG: (Decoder Types JPEG:)] ドロップダウン メニューの [ハードウェア

JPEG デコーダ (Hardware JPEG Decoder)] がチェックされていることを確認してください。[Exit Pixel map] ウィンドウの Pixelmap 出力フォーマットについては（[pixelmap] -> [edit settings] を右クリックするとウィンドウが開きます)、[Raw Format] を選択してください。このように設定することで、GUIX Studio により、圧縮されていない JPEG エンコードされた画像リソース データが生成されます。JPEG ハードウェアはこのフォーマットをリードできるうえに、フレーム バッファに画像のデコードと描画を実行できます。

GUIX キャンバスバッファ

モジュールでは、グラフィックス画面イメージを回転するために使用される GUIX キャンバスバッファがサポートされます。GUIX キャンバスバッファのサイズは、ディスプレイモジュールのフレームバッファとまったく同じでなければなりません。GUIX キャンバスバッファを使用すると、追加のグラフィックイメージ処理が必要なため、グラフィックス性能に影響します。このため、GUIX キャンバスバッファは画面のローテーションが必要な場合にのみ使用する必要があります。それ以外の場合は、sf_el_gx_cfg_t::p_canvas に NULL を設定して、GUIX が画像をフレームバッファに直接描画するようにします。

画面のローテーション

このモジュールは、画面のローテーションをサポートしています。GUIX は GUIX Studio で設計されたとおりに画面イメージを GUIX キャンバスバッファに描画しますが、SF_EL_GX モジュールは画面のローテーションを実行して、回転された画面イメージをフレームバッファに描画できます。たとえば、GUIX Studio で横長イメージとして設計された GUI 画面を回転し、横長形状のディメンションの LCD パネルに表示できます。サポートされているローテーション角度は反時計回りに 90、180、270 度で、これは sf_el_gx_api_t::open で設定できます。動的な画面ローテーションはサポートされていません。画面のローテーション機能を有効にするには、GUIX キャンバス バッファを使用する必要があります。GUIX はまずキャンバスに画面更新を描画し、その後 GUIX ポートがフレーム バッファへの画面コピーを反時計回りにイメージを回転させて処理します。*Synergy 2D 描画エンジンサポート*が有効になっている場合 (GX_USE_SYNEGY_DRW = 1)、ローテーションは 2D 描画エンジンによってテクスチャマッピングで処理されます。有効になっていない場合 (GX_USE_SYNEGY_DRW = 0)、ローテーションはソフトウェアコピーによって処理されます。

（sf_el_gx_cfg_t::rotation_angle = 0）かつ（sf_el_gx_cfg_t::p_canvas = NULL 値以外）という構成は禁止されてはいませんが、行わないでください。この構成は、GUIX キャンバスバッファからフレームバッファへの画面イメージコピーに余分なバス帯域を消費します。したがって、sf_el_gx_cfg_t::p_canvas に NULL を設定し、GUIX キャンバスバッファを使用しないようにします。

JPEG 作業バッファのサイズ

JPEG 作業バッファと JPEG のデコード速度は、バッファ サイズに対してトレードオフの関係にあります。画面のウェジェットが JPEG でフォーマットされている場合は、JPEG 作業バッファは一時ストレージ メモリとして使用され、デコードされた画像が作成されます。画像全体をデコードするにあたってバッファ サイズの大きさが不足している場合は、JPEG デコーディングは出力バッファ ストリーミング モードで実行されます。2D 描画エンジンの BitBLT 操作では、バッファの JPEG ラスタ画像のピースがデコードされ、その後フレームバッファに転送されます。JPEG 作業バッファの最小サイズは、{（水平ラインのピクセル数）x（ディスプレイフォーマットの bpp（1 ピクセルあたりのバイト数））x 8（ライン）} です。たとえば、デコードした画像が水平ライン 800 ピクセルで RGB565 フォーマットの場合では、この数は 800 x 2 x 8= 12 800 (バイト) となります。バッファサイズがこの数値より小さい場合、JPEG デコーディングは実行されません。スループットを向上させるには、[JPEG 作業バッファのサイズ] のパラメータをもっと大きくする必要があります。それにより、JPEG デコードのスループットが向上します。JPEG 出力バッファストリーミングモードは部分的な JPEG デコード操作を繰り返し、補充がオーバーヘッドになります。

D/AVE 2D バッファキャッシュ

D/AVE 2D バッファキャッシュは、以下の Synergy コンフィギュレータの構成を通じて有効化または無効化することができます。解像度が高い、32 ビット ARGB8888 カラーフォーマットの画像を使用する場合は無効にしてください。

- D/AVE 2D フレームバッファキャッシュ（D/AVE 2D 描画エンジンが有効な場合に有効）

単一のバッファ設計における画面の切れ目

画面の切れ目は、ディスプレイデバイスが複数フレームからの情報を単一の画面描画で表示するビデオ表示の視覚的アーティファクトです。一般的に、単一フレームバッファを使用するシステムは、LCD パネルにおける画面の切れ目問題の原因となります。単一フレームバッファを使用することはできますが（sf_el_gx_cfg_t::p_framebuffer_b に NULL を設定)、画面の切れ目がモジュールによって対処されることはありません。2 つのフレームバッファから構成されるピンポンフレームバッファシステムを使用することをお勧めします。

- SF_EL_GX は、GLCDC を搭載した Synergy MCU（必須）、2D 描画エンジン、JPEG エンジン（オプション）にのみ適用できます
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.4.4 アプリケーションへの GUIX Synergy ポートフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに GUIX Synergy ポートフレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

GUIX Synergy ポートフレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

GUIX Synergy ポートフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_el_gx0 GUIX Port on sf_el_gx	Threads	New Stack> Framework> Graphics> GUIX Port on sf_el_gx

次の図に示すように、GUIX Synergy ポートフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

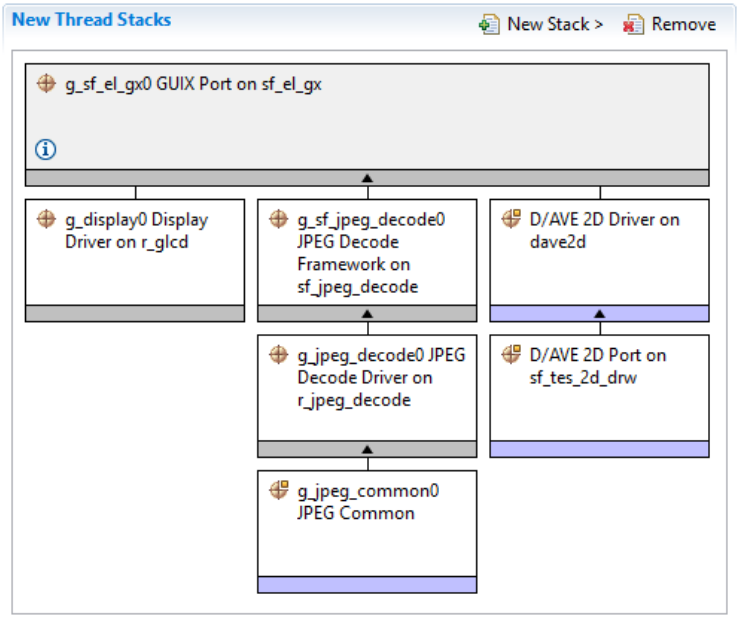


図 308:**GUIX Synergy ポートフレームワークモジュールのスタック**

4.3.4.5 GUIX Synergy ポートフレームワークモジュールの構成

ユーザーは必要な動作に合わせて、GUIX Synergy ポートフレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_el_gx 上の GUIX Synergy ポートフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.

ISDE Property	Value	Description
Name	g_sf_el_gx0	Name of SF_EL_GX instance which will be generated by ISDE. Specify the instance name of this module. Name must be a valid C symbol.
Display Driver Configuration Instance	Inherit Graphics Screen 1, Inherit Graphics Screen 2 Default: Inherit Graphics Screen 1	Display drive configuration instance selection
Name of User Callback function	NULL	Name of User Callback function invoked by the Module when events happen. It must be a valid C symbol and NULL is allowed.
Screen Rotation Angle (Clockwise)	0	Angle of screen rotation (degree). If non-zero value is selected, screen rotation is enabled and GUIX draws screen image on a frame buffer, rotating the image with the angle in the counter clockwise way.
GUIX Canvas Buffer (required if rotation angle is not zero)	Not used	If enabling the screen rotation, a canvas buffer must be used. The size of canvas buffer must be exactly the same as a frame buffer for the display module.
Size of JPEG Work Buffer (valid if JPEG hardware acceleration enabled)	768000	The JPEG work buffer size in bytes. Value must be a valid integer value and zero is allowed to be set if JPEG acceleration is not used. Larger buffer size shortens the drawing time. See Size of JPEG Work Buffer

ISDE Property	Value	Description
Memory section for GUIX Canvas Buffer	sdram	Name of memory section where you want to allocate the GUIX Canvas Buffer. Enter a valid section name defined in the linker script file. Name must be a valid C symbol.
Memory section for JPEG Work Buffer	sdram	Name of memory section where you want to allocate the JPEG Work Buffer. Enter a valid section name defined in the linker script file. Name must be a valid C symbol.
D/AVE 2 2D Frame Buffer Cache (Valid if D/AVE 2D Drawing Engine is enabled)	Enable, Disable Default: Enable	If Synergy 2D Drawing Engine (DRW) Support is enabled, the rotation is processed by Synergy DRW with texture mapping. If not enabled, the rotation is processed by software copy.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

GUIX Synergy ポートフレームワークのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_glcd の GLCD HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.

ISDE Property	Value	Description
Name	g_display0	The name to be used for a GLCDC module control block instance. This name is also used as the prefix of the other variable instances.
Name of display callback function to be defined by user	NULL	Name must be a valid C symbol.
Input - Panel clock source select	Internal clock(GLCDCLK), External clock(LCD_EXTCLK) Default: Internal clock (GLCDCLK)	Choose the panel clock source depends on your system.
Input - Graphics screen1	Used, Not used Default: Used	Specify "Used" if the graphics screen N is used. Then the frame buffer named "display_fb_background" for graphics screen1 and "display_fb_foreground" for graphics screen2 is auto-generated by ISDE. If not using either of the graphics screens, specify "Not used". Then the frame buffer is not created. Note that there is no memory read access to the frame buffer when you specify "Not used", which reduces the consumption of bus bandwidth.
Input - Graphics screen1 frame buffer name	fb_background	Custom name for frame buffer.
Input - Number of Graphics screen1 frame buffer	2	Number of graphics selection.
Input - section where Graphics screen1 frame buffer allocated	sdram	Specify the section name to allocate the frame buffer. This is valid if "Input - Graphics screen1" is set as "Used."

ISDE Property	Value	Description
Input - Graphics screen1 input horizontal size	800	Specify the number of horizontal pixels. Default value is the size for an image with 800x480 pixels
Input - Graphics screen1 vertical size	480	Specify the number of vertical pixels. Default value is the size for an image with 800x480 pixels.
Input - Graphics screen1 input horizontal stride (not bytes but pixels)	800	Specify the memory stride for a horizontal line. This value must be specified with the number of pixels, not actual bytes. Typically, this parameter is set to same number as parameter 'input horizontal size'. Default value is the size for an image with 800x480 pixels.
Input - Graphics screen1 input format	32bits ARGB888, 32bits RGB888, 16bits RGB565, 16bits ARGB1555, 16bits ARGB4444, CLUT 8, CLUT 4, CLUT 1 Default: 16bits RGB565	Specify the graphics screen Input format. If selecting CLUT formats, you must write CLUT data using clut before performing start. Default setting supports a RGB565 formatted image.
Input - Graphics screen1 input line descending	Used, Not used Default: Not used	Specify "On" if image data descends from the bottom line to the top line in the frame buffer. Usually "Off".
Input - Graphics screen1 input line repeat	On, Off Default: Off	Specify "On" if expecting to repeatedly read a raster image which is smaller than the LCD panel size. Usually "Off". For details, see the description of Line Repeating function.
Input - Graphics screen1 input line repeat times	0	Specify the number of repeating times for a raster image which is read repeatedly in a frame.

ISDE Property	Value	Description
Input - Graphics screen1 layer coordinate X	0	Specify the horizontal offset in pixels of the graphics screen from the background screen.
Input - Graphics screen1 layer coordinate Y	0	Specify the vertical offset in pixels of the graphics screen from the background screen.
Input - Graphics screen1 layer background color alpha	255	Based on the alpha value, either the graphics screen2 (foreground graphics screen) is blended into the graphics screen1 (background graphics screen) or the graphics screen1 is blended into the monochrome background screen.
Input - Graphics screen1 layer background color Red	255	Specify the background color in the graphics screen N.
Input - Graphics screen1 layer background color Green	255	Specify the background color in the graphics screen N.
Input - Graphics screen1 layer background color Blue	255	Specify the background color in the graphics screen N.
Input - Graphics screen1 layer fading control	None, Fade-in, Fade-out Default: None	Specify "On" when performing a fade-in for the graphics screen. The transparent screen changes gradually to opaque. Specify "Off" when performing the fade-out for the graphics screen. The opaque screen changes gradually to transparent. Note that this processing is accelerated by the GLCDC hardware and cannot stop once started. The transition status can be monitored by statusGet.

ISDE Property	Value	Description
Input - Graphics screen1 layer fade speed	0	Specify the number of frames for the fading transition to complete.
Input - Graphics screen2	Used, Not used Default: Not used	Specify "Used" if the graphics screen N is used. Then the frame buffer named "display_fb_background" for graphics screen1 and "display_fb_foreground" for graphics screen2 is auto-generated by ISDE. If not using either of the graphics screens, specify "Not used". Then the frame buffer is not created. Note that there is no memory read access to the frame buffer when you specify "Not used", which reduces the consumption of bus bandwidth.
Input - Graphics screen2 frame buffer name	fb_foreground	Custom name for frame buffer.
Input - Number of Graphics screen2 frame buffer	2	Number of graphics selection.
Input - section where Graphics screen2 frame buffer allocated	sdram	Specify the section name to allocate the frame buffer. This is valid if "Input - Graphics screen1" is set as "Used."
Input - Graphics screen2 input horizontal size	800	Specify the number of horizontal pixels. Default value is the size for an image with 800x480 pixels
Input - Graphics screen2 vertical size	480	Specify the number of vertical pixels. Default value is the size for an image with 800x480 pixels.

ISDE Property	Value	Description
Input - Graphics screen2 input horizontal stride (not bytes but pixels)	800	Specify the memory stride for a horizontal line. This value must be specified with the number of pixels, not actual bytes. Typically this parameter is set to same number as parameter 'input horizontal size'. Default value is the size for an image with 800x480 pixels.
Input - Graphics screen2 input format	32bits ARGB888, 32bits RGB888, 16bits RGB565, 16bits ARGB1555, 16bits ARGB4444, CLUT 8, CLUT 4, CLUT 1 Default: 16bits RGB565	Specify the graphics screen Input format. If selecting CLUT formats, you must write CLUT data using clut before performing start. Default setting supports a RGB565 formatted image.
Input - Graphics screen2 input line descending	On, Off Default: Off	Specify "On" if image data descends from the bottom line to the top line in the frame buffer. Usually "Off".
Input - Graphics screen2 input line repeat	On, Off Default: Off	Specify "On" if expecting to repeatedly read a raster image which is smaller than the LCD panel size. Usually "Off". For details, see the description of Line Repeating function.
Input - Graphics screen2 input line repeat times	0	Specify the number of repeating times for a raster image which is read repeatedly in a frame.
Input - Graphics screen2 layer coordinate X	0	Specify the horizontal offset in pixels of the graphics screen from the background screen.
Input - Graphics screen2 layer coordinate Y	0	Specify the vertical offset in pixels of the graphics screen from the background screen.

ISDE Property	Value	Description
Input - Graphics screen2 layer background color alpha	255	Based on the alpha value, either the graphics screen2 (foreground graphics screen) is blended into the graphics screen1 (background graphics screen) or the graphics screen1 is blended into the monochrome background screen.
Input - Graphics screen2 layer background color Red	255	Specify the background color in the graphics screen N.
Input - Graphics screen2 layer background color Green	255	Specify the background color in the graphics screen N.
Input - Graphics screen2 layer background color Blue	255	Specify the background color in the graphics screen N.
Input - Graphics screen2 layer fading control	None, Fade-in, Fade-out Default: None	Specify "On" when performing a fade-in for the graphics screen. The transparent screen changes gradually to opaque. Specify "Off" when performing the fade-out for the graphics screen. The opaque screen changes gradually to transparent. Note that this processing is accelerated by the GLCDC hardware and cannot stop once started. The transition status can be monitored by statusGet.
Input - Graphics screen2 layer fade speed	0	Specify the number of frames for the fading transition to complete.
Output - Horizontal total cycles	1024	Specify the total cycles in a horizontal line. Set to the number of cycles defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.

ISDE Property	Value	Description
Output - Horizontal active video cycles	800	Specify the number of active video cycles in a horizontal line. Set to the number of cycles defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.
Output - Horizontal back porch cycles	46	Specify the number of back porch cycles in a horizontal line. Back porch starts from the beginning of Hsync cycles, which means back porch cycles contain Hsync cycles. Set to the number of cycles defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.
Output - Horizontal sync signal cycles	20	Specify the number of Hsync signal assertion cycles. Set to the number of cycles defined in the data sheet of LCD panel sheet in your system. Default value matches LCD panel on S7G2 PE-HMI1 board.
Output - Horizontal sync signal polarity	Low active, High active Default: Low active	Select the polarity of Hsync signal to match your system. Default setting matches the LCD panel on S7G2 PE-HMI1 board.
Output - Vertical total lines	525	Specify number of total lines in a frame. Set to the number of lines defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.

ISDE Property	Value	Description
Output - Vertical active video lines	480	Specify the number of active video lines in a frame. Set to the number of lines defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.
Output - Vertical back porch lines	23	Specify the number of back porch lines in a frame. Back porch starts from the beginning of Vsync lines, which means back porch lines contain Vsync lines. Set to the number of lines defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.
Output - Vertical sync signal lines	10	Specify the Vsync signal assertion lines in a frame. Set to the number of lines defined in the data sheet of LCD panel sheet in your system. Default value matches the LCD panel on S7G2 PE-HMI1 board.
Output - Vertical sync signal polarity	Low active, High active Default: Low active	Select the polarity of Vsync signal to match to your system. Default setting matches LCD panel on S7G2 PE-HMI1 board.
Output - Format	24bits RGB888, 18bits RGB666, 16bits RGB565, 8bits serial Default: 24bits RGB888	Specify the graphics screen output format to match to your LCD panel. Default setting matches the LCD panel on S7G2 PE-HMI1 board.
Output - Endian	Little endian, Big endian Default: Little endian	Select data endian for output signal to LCD panel. Default setting matches the LCD panel on S7G2 PE-HMI1 board.

ISDE Property	Value	Description
Output - Color order	RGB, BGR Default: RGB	Select data order for output signal to LCD panel. The order of blue and red can be swapped if needed. Default setting matches the LCD panel on S7G2 PE-HMI1 board.
Output - Data Enable Signal Polarity	Low active, High active Default: High active	Select the polarity of Data Enable signal to match to your system. Default setting matches the LCD panel on S7G2 PE-HMI1 board.
Output - Sync edge	Rising Edge, Falling Edge Default: Rising Edge	Select the polarity of Sync signals to match to your system. Default setting matches the LCD panel on S7G2 PE-HMI1 board.
Output - Background color alpha channel	255	Specify the background color of the background screens.
Output - Background color R channel	0	Specify the background color of the background screens.
Output - Background color G channel	0	Specify the background color of the background screens.
Output - Background color B channel	0	Specify the background color of the background screens.
CLUT	Used, Not used Default: Not used	Specify "Used" if selecting CLUT formats for a graphics screen input format. Then, a buffer named "CLUT_buffer" for the CLUT source data is generated in the ISDE auto-generated source file.

ISDE Property	Value	Description
CLUT - CLUT buffer size	256	Specify the number of entries for the CLUT source data buffer. Each entries consumes 4 bytes (1 word). Words of CLUT source data specified by this parameter are generated in the ISDE auto-generated source file.
TCON - Hsync pin select	Not used, LCD_TCON0, LCD_TCON1, LCD_TCON2, LCD_TCON3 Default: LCD_TCON0	Select the TCON pin used for the Hsync signal to match to your system. Default setting is for LCD panel on S7G2 PE-HMI1 board.
TCON - Vsync pin select	Not used, LCD_TCON0, LCD_TCON1, LCD_TCON2, LCD_TCON3 Default: LCD_TCON1	Select TCON pin used for Vsync signal to match to your system. Default setting is for LCD panel on S7G2 PE-HMI1 board.
TCON - DataEnable pin select	Not used, LCD_TCON0, LCD_TCON1, LCD_TCON2, LCD_TCON3 Default: LCD_TCON2	Select TCON pin used for DataEnable signal to match to your system. Default setting is for LCD panel on S7G2 PE-HMI1 board.
TCON - Panel clock division ratio	1/1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, 1/9, 1/12, 1/16, 1/24, 1/32 Default: 1/8	Select the clock source divider value. See the note at bottom of this table about the source clock for the pixel clock.
Color correction - Brightness	Off, On Default: Off	Specify "On" when performing brightness control. If specifying "Off", the setting below does not affect the output color.
Color correction - Brightness R channel	512	Output color level is calculated as follows: Output color level = Input color level +/ - 512. Set the value for each of R, G, B channels.

ISDE Property	Value	Description
Color correction - Brightness G channel	512	Output color level is calculated as follows: Output color level = Input color level +/ - 512. Set the value for each of R, G, B channels.
Color correction - Brightness B channel	512	Output color level is calculated as follows: Output color level = Input color level +/ - 512. Set the value for each of R, G, B channels.
Color correction - Contrast	Off, On Default: Off	Specify "On" when performing contrast control. If specifying "Off", the setting below does not affect the output color.
Color correction - Contrast(gain) R channel	128	Output color level is calculated as follows: Output color level = Input color level x (/128). Set the value for each of R, G, B channels.
Color correction - Contrast(gain) G channel	128	Output color level is calculated as follows: Output color level = Input color level x (/128). Set the value for each of R, G, B channels.
Color correction - Contrast(gain) B channel	128	Output color level is calculated as follows: Output color level = Input color level x (/128). Set the value for each of R, G, B channels.
Color correction - Gamma correction(Red)	Off, On Default: Off	Control for each channel R/G/B. Specify "On" when performing gamma correction for the red channel. If specifying "Off", the settings for gain and threshold do not affect the output color.

ISDE Property	Value	Description
Color correction - Gamma gain R[0-15]	0	Set the gain value for the red channel in the area N on the gamma correction curve. The gain setting for area N is applied to the input data with a color level between ((Gamma threshold R[N-1])<<2) and ((Gamma threshold R[N])<<2). The output value is calculated as follows: Output color level = Input color level / 1024 (/128).
Color correction - Gamma threshold R[0-15]	0	Set the threshold value for the red channel in the area N on the gamma correction curve. The gain setting for area N is applied to the input data with a color level between Gamma threshold R[N-1] and Gamma threshold R[N]. The output value is calculated as follows: Output color level = Input color level / 1024 (/128).
Color correction - Gamma correction(Green)	Off, On Default: Off	Control for each channel R/G/B. Specify "On" when performing gamma correction for the red channel. If specifying "Off", the settings for gain and threshold do not affect the output color.
Color correction - Gamma gain G[0-15]	0	Set the gain value for the red channel in the area N on the gamma correction curve. The gain setting for area N is applied to the input data with a color level between ((Gamma threshold R[N-1])<<2) and ((Gamma threshold R[N])<<2). The output value is calculated as follows: Output color level = Input color level / 1024 (/128).

ISDE Property	Value	Description
Color correction - Gamma threshold G[0-15]	0	Set the threshold value for the red channel in the area N on the gamma correction curve. The gain setting for area N is applied to the input data with a color level between Gamma threshold R[N-1] and Gamma threshold R[N]. The output value is calculated as follows: Output color level = Input color level / 1024 (/128).
Color correction - Gamma correction(Blue)	Off, On Default: Off	Control for each channel R/G/B. Specify "On" when performing gamma correction for the red channel. If specifying "Off", the settings for gain and threshold do not affect the output color.
Color correction - Gamma gain B[0-15]	0	Set the gain value for the red channel in the area N on the gamma correction curve. The gain setting for area N is applied to the input data with a color level between ((Gamma threshold R[N-1])<<2) and ((Gamma threshold R[N])<<2). The output value is calculated as follows: Output color level = Input color level / 1024 (/128).

ISDE Property	Value	Description
Color correction - Gamma threshold B[0-15]	0	Set the threshold value for the red channel in the area N on the gamma correction curve. The gain setting for area N is applied to the input data with a color level between Gamma threshold R[N-1] and Gamma threshold R[N]. The output value is calculated as follows: Output color level = Input color level / 1024 (/128).
Dithering	Off, On Default: Off	Dithering enable. Specify "On" when applying the dither effect to reduce the banding in case of selecting RGB666 or RGB565 output formats. Dithering can be applied when converting. If specified "Off", the settings for dithering below do not affect the output. For details on the dither effect, see Output Control Block Panel Dither Correction Register (OUT_PDTHA) in the hardware manual.
Dithering - Mode	Truncate, Round off, 2x2 Pattern Default: Truncate	Specify the dither mode. For detail, see the Output Control Block Panel Dither Correction Register (OUT_PDTHA) in the hardware manual.
Dithering - Pattern A	Pattern 00, Pattern 01, Pattern 10, Pattern 11 Default: Pattern 11	Specify the dither pattern for 2X2 pattern mode. Forndetails, see the Output Control Block Panel Dither Correction Register (OUT_PDTHA) in the hardware manual.

ISDE Property	Value	Description
Dithering - Pattern B	Pattern 00, Pattern 01, Pattern 10, Pattern 11 Default: Pattern 11	Specify the dither pattern for 2X2 pattern mode. Forndetails, see the Output Control Block Panel Dither Correction Register (OUT_PDTHA) in the hardware manual.
Dithering - Pattern C	Pattern 00, Pattern 01, Pattern 10, Pattern 11 Default: Pattern 11	Specify the dither pattern for 2X2 pattern mode. Forndetails, see the Output Control Block Panel Dither Correction Register (OUT_PDTHA) in the hardware manual.
Dithering - Pattern D	Pattern 00, Pattern 01, Pattern 10, Pattern 11 Default: Pattern 11	Specify the dither pattern for 2X2 pattern mode. Forndetails, see the Output Control Block Panel Dither Correction Register (OUT_PDTHA) in the hardware manual.
Misc - Correction Process Order	Brightness and Contrast then Gamma, Gamma then Brightness and Contrast Default: Brightness and Contrast then Gamma	Specify the color correction processing order if needed.
Line Detect Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	The driver needs valid interrupt priority setting and it won't work if disabled.
Underflow 1 Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	The driver needs valid interrupt priority setting and it won't work if disabled.

ISDE Property	Value	Description
Underflow 2 Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	The driver needs valid interrupt priority setting and it won't work if disabled.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_jpeg_decode 上の JPEG デコードフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_jpeg_decode0	The name to be used for a JPEG Decode Framework module instance.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_jpeg_decode 上の JPEG デコードドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter error checking.
Name	g_jpeg_decode0	The name to be used for a JPEG Decode module instance.

ISDE Property	Value	Description
Byte Order for Input Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2) Default: Normal Byte order	Specify the byte order for input data. The order is swapped as specified in every 8-byte.
Byte Order for Output Data Format	Normal byte order (1)(2)(3)(4)(5)(6)(7)(8), Byte Swap (2)(1)(4)(3)(6)(5)(8)(7), Word Swap (3)(4)(1)(2)(7)(8)(5)(6), Word-Byte Swap (4)(3)(2)(1)(8)(7)(6)(5), Longword Swap (5)(6)(7)(8)(1)(2)(3)(4), Longword-Byte Swap (6)(5)(8)(7)(2)(1)(4)(3), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2), Longword-Word Swap (7)(8)(5)(6)(3)(4)(1)(2) Default: Normal Byte order	Specify the byte order for output data. The order is swapped as specified in every 8-byte.
Output Data Color Format	Pixel Data RGB565 format, Pixel Data ARGBB888 format Default: Pixel Data RGB565 format	Specify the output data format.
Alpha value to be applied to decoded pixel data (only valid for ARGB8888 format)	255	Specify the alpha value for the output data format (only valid for ARGB8888 format).

ISDE Property	Value	Description
Name of user callback function	NULL	Specify the name of user callback function.
Decompression Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Decompression interrupt priority selection.
Data Transfer Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	Data transfer interrupt priority selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

JPEG 共通インスタンス

ISDE Property	Value	Description
Name	g_sf_jpeg_common	Module name.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

dave2d 上の D/AVE 2D ドライバーの構成設定

ISDE Property	Value	Description
No configurable properties()	-	-

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

sf_tes_2d_drw 上の D/AVE 2D ポートの構成設定

ISDE Property	Value	Description
Work memory size for display lists in bytes	32768	Work memory size for display lists selection
DRW Interrupt Priority	Priority 0 (highest), Priority 1:14 Priority 15 (lowest - not valid if using ThreadX) Default: Priority 12	DRW INT selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

GUIX Synergy ポートフレームワークモジュールのクロック構成

GUIX Synergy ポートモジュールは、論理モジュールであるため、ARM Cortex-M コア SysTick タイマ設定を除くハードウェア設定は不要です。

GUIX Synergy ポートフレームワークモジュールのピン構成

GUIX Synergy ポートモジュールは、論理モジュールであるため、ピン設定は不要です。

4.3.4.6 アプリケーションでの GUIX Synergy ポートフレームワークモジュールの使用

これらの重要な設定は、Synergy コンフィギュレータで行われ、モジュールの初期化に使用されます。

- モジュールクロック設定および GLCDC 割り込み優先順位を含む GLCDC 構成をセットアップします。通常、構成は Synergy コンフィギュレータを使用して自動生成することができます。
- モジュールクロック設定およびハードウェア割り込み優先順位を含む 2D 描画エンジンまたは JPEG エンジンの構成をセットアップします。通常、構成は Synergy コンフィギュレータを使用して自動生成することができます。

一般的なアプリケーションで sf_el_gx モジュール上の GUIX ポートを使用する手順は次のとおりです。

1) sf_el_gx_api_t::open API をコールして SF_EL_GX コントロールブロックを初期化し、モジュール構成設定を渡します。
2) GUIX Studio で生成された gx_studio_display_configure API をコールして初期化を完了し、下の図に示すように SF_EL_GX セットアップ関数を渡します。この関数コールによって、Synergy グラフィックスハードウェアアクセラレータの初期化が完了します。
   call.gx_studio_display_configure(MAIN_DISPLAY,g_sf_el_gx0.p_api->setup,LANGUAGE_ENGLISH,MAIN_DISPLAY_THEME,&p_window_root);

を使用して、GUIX によって初期化されたルートウィンドウのアドレスを取得します。

3) sf_el_gx_api_t::canvasInit API をコールすることで、GUIX キャンバスバッファのプライマリメモリアドレスを初期化します。
4) GUIX gx_studio_named_widget_create API をコールしてルートウィンドウを表示します。
5) GUIX gx_widget_show API をコールしてルート画面を表示します。

6) GUIX gx_system_start API をコールして、GUIX システムを起動します。

SF_EL_GX モジュールは、GUIX システムの起動後は GUIX により制御されます。この後、アプリケーションからはいずれの操作も実行する必要がありません。

これらの一般的な手順を、次の図の通常の動作フローに示します。

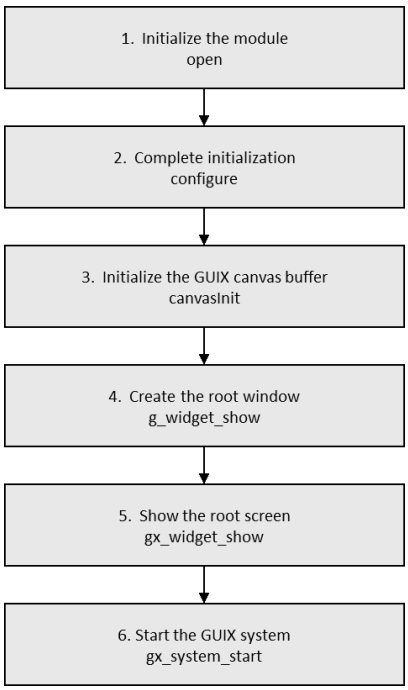


図 309:通常の GUIX Synergy ポートフレームワークモジュールアプリケーションのフロー図

4.3.5 GX_SRC フレームワーク

Express Logic GUIX ソースコンポーネント（GX_SRC）は、GUIX の RTOS コンポーネント（システムタイマなど）を修正するために GUIX で使用できるプロパティのリストです。GUIX コンポーネントのリストについては、『GUIX ユーザーガイド』の第 3 章「GUIX システムコンポーネント」を参照してください。GUIX のサービス呼び出しには、GUIX の「エンジン」がグラフィックスを管理している間にアプリケーションが GUIX サービスを使用できるよう、保護のための相互排除の機能が組み込まれています。GUIX は、ThreadX スレッド、タイマ、およびメッセージキューを使用して、表示イベントを管理し、画面をレンダリングします。

4.3.5.1 GUIX GX_SRC フレームワークモジュールの特長

GUIX GX_SRC フレームワークモジュールには、以下のオプションが含まれます。

- マルチスレッドサポートの無効化
- UTF8 サポートの無効化
- ハードウェアに合わせたシステムタイマの設定

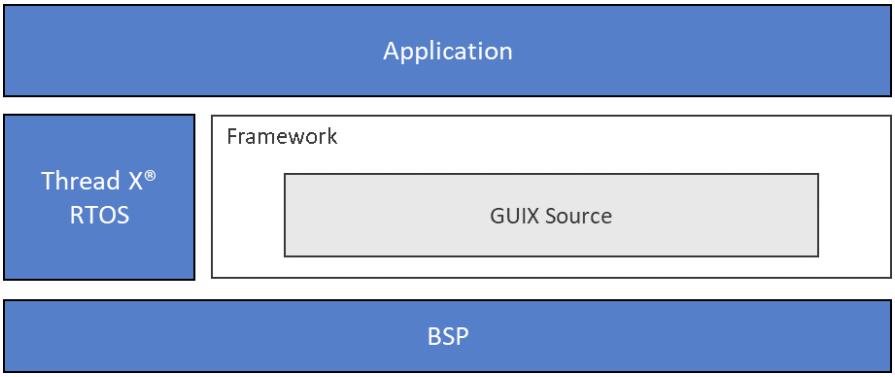


図 310:GUIX GX_SRC フレームワークモジュールのブロック図

4.3.5.2 GUIX GX_SRC フレームワークコンポーネントの概要

現在のリリースの GUIX で使用できる API 関数は 300 以上あります。詳細な API のリストおよび説明については、『GUIX ユーザーガイド』を参照してください。GUIX コンポーネントのリストは、次のように説明されています。

GUIX コンポーネント

GUIX システム – GUIX エンジンの実行に必要なタイマ、スレッド、および内部オブジェクト

GX_CANVAS* – gx_canvas_create. で設定します。GUIX デザインが単純な場合、この関数は GUIX Studio の仕様ファイルによってコールされるため、自動的に処理されます。

GUIX CONTEXT – GUIX の描画先（画面、キャンバス、ブラシなど）を構成するすべてのコンポーネント。

GX_DISPLAY* – GUIX がキャンバスに描画する必要がある、描画関数を含むすべてのコンポーネント。GUIX ディスプレイドライバーは、基盤となる物理画面とのすべての対話を担当します。ディスプレイドライバーには、初期化、描画、フレームバッファ表示の 3 つの基本機能があります。初期化は、物理ディスプレイハードウェアを準備し、GUIX に物理ディスプレイハードウェアのプロパティを通知し、GUIX にどの描画関数を使用すべきかを知らせるために行います。メインディスプレイドライバーの初期化は、GUIX の gx_display_create API からコールされます。この関数は、リソースおよび仕様を使用するアプリケーションの gx_studio_display_configure からコールされます

GUIX Studio では、仕様ファイルによりディスプレイドライバーが自動的にセットアップされます。GUIX Studio の [Configure Displays] ダイアログボックスで 16bpp が選択されている場合、_gx_display_driver_565rgb_setup が使用されます。32bpp が選択されている場合は、_gx_display_driver_32argb_setup, が取得されるなど、以下同様となります。

Synergy の GUIX は次のフォーマットをサポートします。

- 32bpp（ARGB8888、RGB-888）
- 16bpp（RGB565）
- 8bpp（8 ビットパレット（CLUT））

GUIX Studio でターゲット CPU が Synergy に設定されている場合、ディスプレイドライバー関数は、GUIX Studio の仕様ファイルにセットアップされます（Synergy 向けの GUIX Studio セットアップの詳細については、IOTSG-782 モジュールガイドドキュメントを参照してください）。

```
UINT _gx_synergy_display_driver_setup(GX_DISPLAY *display)
{
    _gx_display_driver_565rgb_setup(display, GX_NULL, _gx_dave2d_buffer_toggle);
```

ユーザーアプリケーションでは、ディスプレイドライバーの指定を含めたメインディスプレイの構成を gx_studio_display_configure API が行います。ディスプレイドライバーは Synergy によって既に SF_EL_GX インスタ

ンスに関連付けられています。次のように、SF_EL_GX ドライバーセットアップ関数ポインタによってディスプレイドライバーが提供されます。

```
gx_studio_display_configure (MAIN_DISPLAY,
                             g_sf_el_gx0.p_api->setup,…
```

GX_WIDGET* – イベントを生成するか、しないかにかかわらず画面上に配置するすべての可視項目。

GUIX UTILITY – 基本的にユーティリティ関数のツールボックス。

注 : GUIX の構造体データ型として定義されます。*

4.3.5.3 GUIX GX_SRC フレームワークモジュールの動作の概要

コンフィギュレータでの GUIX のセットアップ

GUIX ソースコンポーネントはオプションです。GUIX ソースを使用しない場合、デフォルト設定でビルド済みの GUIX ライブラリが使用されます。GUIX ソースコンポーネントを追加した場合、またはそのプロパティを変更した場合、開発者はコンフィギュレータペインで [Generate Project Content] をクリックしてプロジェクトを再ビルドする必要があります。再ビルドしないと、事前にビルドされた GuiX（gx.a）ライブラリがいずれのプロパティも変更されずに使用されます。

開始時の GUIX

GUIX 画像をレンダリングする前に、次の手順を実行する必要があります。

- gx_system_initialize API をコールして、GUIX システムを初期化します。
- GUIX ドライバーを初期化します（g_sf_el_gx.p_api->open）。
- 表示インスタンスを作成および構成します（gx_studio_display_configure のコールでドライバー入力として g_sf_el_gx.p_api->setup を指定）。
- 表示用のキャンバスを作成および初期化します（g_sf_el_gx.p_api->canvasInit API）。
- GUIX Studio の仕様およびリソースファイルを使用する場合は、メイン画面の表示を準備します（gx_studio_display_configure API）。
- [オプション] 作成されていない場合は他のウィンドウまたはウィジェットを作成します（gx_widget_create）.。
- ルートウィンドウを表示します（gx_window_show API）。
- GUIX エンジンを開始します（gx_system_start API）。

gx_system_initialize API で、GUIX は、すべての描画、タイマ、および GUIX 関連のタスクを実行するために必要な自身のスレッド、イベントキュー、およびタイマを作成します。また、マルチスレッド環境のためのミューテックスも作成します。アプリケーションスレッドは、GUIX と同じリソースにアクセスできます。たとえば、GUIX がリフレッシュ動作を行っている場合、ミューテックス保護は、画面のリフレッシュ中、表示されているウィジェットに他のスレッドがアクセスできないようにします。

open、setup および canvasInit API は、sf_el_gx インスタンスの関数ポインタフィールドを使用してアクセスできます。gx_studio_display_configure コールでは、言語とディスプレイ画面名を除く、その他の詳細のみ提供する必要があります。

GX_SRC プロパティ

GX_SRC のための GUIX 設定可能オプション

System Timer	20 msec	Used to calculate GX_SYSTEM_TIMER_TICKS. See explanation following. Equivalent option in Express Logic is GX_SYSTEM_TIMER_MS
Disable Multithread Support	No (Disabled)	If not defined, ThreadX can support multiple threads by defining and using locking and unlocking functions to define critical sections. Equivalent option in Express Logic is GUIX_DISABLE_MULTITHREAD_SUPPORT
Disable UTF8 Support	No (Disabled)	If defined, this disables UTF8 format string encoding in GUIX and allows only 8-bit ASCII character plus Latin-1 code page character encoding. Equivalent option in Express Logic is GX_UTF8_SUPPORT

Express Logic 社のユーザーガイドに記載されている GUIX オプション（ISDE では構成不可）

GX_THREAD_STACK_SIZE	4096 bytes	GUIX thread stack size
GX_SYSTEM_TIMER_TICKS	2 ticks (20 msec)	GUIX timer frequency (interval on which the timer thread entry function checks for tasks (e.g. periodic tasks, timeouts) that need to be executed
GX_TICKS_SECOND	2 ticks	This is only used if porting GUIX to another RTOS other than ThreadX
GX_MAX_VIEWS	32	Number of simultaneous views

GX_MAX_EVENT_TIMERS	32	Number of timers available in GUIX to be assigned to one or more widgets
GX_DISABLE_THREADX_BINDING	Not enabled	If not defined, GUIX expects a ThreadX RTOS; GUIX can work with other RTOS's but this is not supported in Synergy.

マルチスレッドサポートの無効化

gx_system_initialize APIは、GUIXの描画およびイベント処理スレッドを作成します。このスレッドは、gx_system_start API がコールされると開始されます。GUIX アプリケーションスレッドでは、GUIX システムスレッドの実行中に GUIX API をコールする必要が生じることがあります。たとえば、アプリケーションスレッドがポップアップエラーウィンドウを作成し、これをルートウィンドウにアタッチして表示するような場合です。問題は、2 つ以上のスレッドが内部 GUIX リソースに同時にアクセスすると、予期せぬ結果が生じることです。このため、マルチスレッド環境の場合、GUIX ではリンクされたリストやその他の内部オブジェクトを更新する重要なコードセクションを保護する必要があります。GUIX は、この重要なコードセクションの保護のためにミューテックスを使用しています。マルチスレッドサポートプロパティが無効になっていない場合には、GUIX マクロにより重要なセクションの保護を実現することになります。通常、システムのリソース（メモリ）の制約が非常に厳しい場合を除き、このオプションは無効にしておくことをお勧めします。

アプリケーションがいずれの GUIX API もコールしない場合、GUIX に影響するスレッドはシステムスレッドだけです。マルチスレッドサポートを無効にすると、重要なコードセクションの保護のオーバーヘッドが削減され、パフォーマンスが向上します。

GUIX システムタイマ

GUIX システムタイマは、GUIX タイマが実行されるインターバルであり、GUIX 内部で必要な定期処理のために GUIX が使用します。システムタイマは ThreadX タイマの倍数にする必要があり、通常は 10 ミリ秒（100 ティック / 秒）に設定します。例：

GUIX システムタイマを 10 にします。

ThreadX タイマは 100 です。

```
GUIX timer tick = ((10msec * 100 ticks per second) / 1000 msec per second) == 1
tick
```

これにより、GUIX タイマティックが 1 ティックに設定されます。GUIX システムタイマが ThreadX タイマの倍数でない場合、この値は切り捨てられます。上の例で、GUIX システムタイマを 17 に設定した場合でも、GUIX タイマティックは 1 のままです。GUIX タイマティックを ThreadX タイマティックの 2 ティックごとにするには、GX_SRC のシステムタイマプロパティを 20 に設定します。

```
GUIX timer tick = ((20 * 100) / 1000) == 2.
```

UTF8 サポート

デフォルトでは UTF8 が有効です。UTF8 を無効にすると、GUIX は ASCII タイプ（グリフあたり 1 バイト）に限られます。アプリケーションが ASCII のみを使用している場合でも、UTF8 サポートに関連した実行時オーバーヘッドは常に存在します。これは、UTF8 を使用している場合、GUIX は文字列の文字が 1 バイト文字であると仮定できず、2 バイト目の文字値を計算する関数を呼び出す必要があるためです。

UTF8 を無効にする代わりに、アプリケーションは 255 より大きい文字値を使用しないようにすることもできます。UTF8 を無効にすると、GUIX に各文字が単純に 1 バイトであることが伝わり、next-character-compute 関数を呼び出す必要がなくなります。

ディスプレイメモリアーキテクチャ

GUIX では、いくつかのディスプレイメモリアーキテクチャがサポートされています。ディスプレイメモリアーキテクチャは実際、非常に小さなハードウェアの移植層であるディスプレイドライバーによって定義され、GUIX のコアライブラリコードには影響しません。最も一般的なモデルは、「作業」バッファと「可視」バッファの 2 つのキャンバスメモリ領域を用意するというものです。GUIX は作業キャンバスに対する描画更新を実行し、描画シーケンスが完了すると作業キャンバスバッファと可視キャンバスバッファを切り替えます。

以下に 4 つの基本的なメモリモデルを説明します。

次の図のモデル 1 とモデル 2 にはそれぞれ GRAM とフレームバッファのみが含まれています。これは低速であるため、非常に小さいディスプレイにのみ推奨されます。モデル 1 では、ディスプレイはピクセルデータを保持するのに十分な大きさの独自のメモリを備えています。この外部メモリは通常「GRAM」またはグラフィックス RAM と呼ばれ、コア CPU に SPI アクセスなどの非ランダムアクセスが提供されます。このモデルでは、ディスプレイドライバーが SPI インタフェースを介してピクセルデータを書き込むことで、すべての描画操作を実行します。

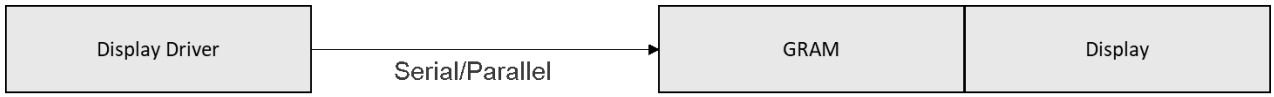


図 311:モデル 1

図 312:モデル 2

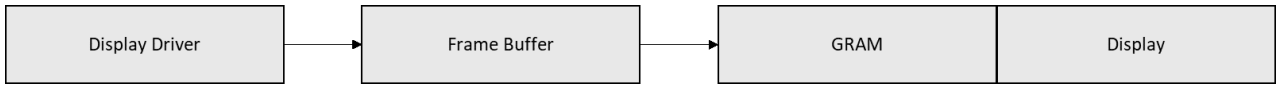


図 313:モデル 3（ローカルフレームバッファ + 外部 GRAM）

モデル 3 は、最初の 2 つのモデルの組み合わせです。このモデルでは、1 つのフレームバッファを保持するのに十分なローカルメモリが存在します。加えて、ディスプレイデバイスは外部 GRAM を備えており、GRAM に提供されたデータを使用して自動的に自身をリフレッシュします。このアーキテクチャでは、ユーザーが多くの場合オンボード DMA チャネルを利用して、1 回のブロック転送でローカルフレームバッファの変更された部分を外部 GRAM に転送できるため、更新の効率が向上するという利点があります。また、このモデルでは完成したグラフィックスコンテンツのみが外部 GRAM にコピーされるため、最初の 2 つのモデルで発生する可能性がある切れ目きやちらつきもなくなります。

モデル 4 は、2 つのローカルフレームバッファを提供するのに十分なメモリを必要とします。この場合、GUIX は 1 つのフレームバッファをアクティブなフレームバッファとして処理し、もう 1 つを作業フレームバッファとして処理します。処理中のディスプレイ更新や描画動作は、作業バッファで行われます。描画動作が完了するとバッファが切り替えられ、作業バッファがアクティブなバッファになり、アクティブなバッファが作業バッファになります。モデル 3 と同様、このモデルでも単一バッファのシステムで発生する可能性がある画面のちらつきや切れ目がなくなります。

モデル 4（ピンポンフレームバッファ）：

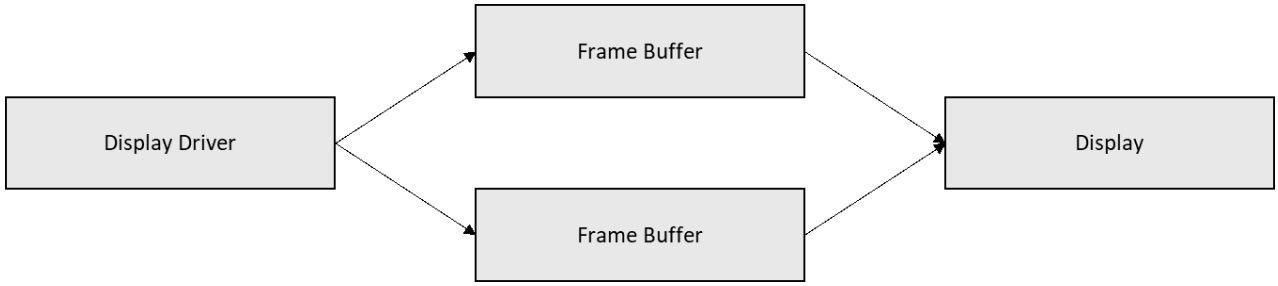


図 314:モデル 4（ピンポンフレームバッファ）

Synergy（SF_EL_GX モジュール）はモデル 4 を実装しています（モデル 3 も実装可能です）。

ハードウェアからのイベントの受信

GUIX の信号

ハードウェアイベントなどの情報を取得し、GUIX レベルに到達させるためには、信号を使用します。ハードウェアイベントは、GUIX イベントキューに到達するまでに、ハードウェアおよびメッセージフレームワークドライバーによって GX_SIGNALs に変換されます。GX_SIGNAL マクロは、イベントを生成したウィジェットの ID（GUIX Studio でユーザー定義されたもの）とウィジェットが生成したイベントタイプを組み合わせて、一意の gx_event_type 識別子コードにするマクロです。GX_EVENT のタイプと信号の詳細については、IOTSG-782 メタデータのモジュールガイドドキュメントを参照してください。

ハードウェアインタフェースのドライバー

Synergy ドライバーハードウェアでは、ThreadX の定期スレッドタスクが作成され、ハードウェアイベントが継続的にチェックされます。イベントが検出されると、ハードウェアドライバーはこのイベントに関する情報をスレッドタスクに送信します。たとえばタッチスクリーンでは、タッチパネルドライバーは X 座標と Y 座標、およびイベントタイプを中継します。タッチパネルドライバーが使用するペイロードデータタイプの例を以下に示します。

```
/** Touch data payload posted to the message queue. */
typedef struct st_sf_touch_panel_payload
{
    sf_message_header_t     header;     // header for messaging framework.
    int16_t                 x;          // X coordinate.
    int16_t                 y;          // Y coordinate.
    sf_touch_panel_event_t  event_type; // Touch event type.
} sf_touch_panel_payload_t;
```

次に、タッチパネルスレッドタスクは、画面のローテーションなどの追加処理をこのペイロードインスタンスに適用します。メッセージフレームワークとのインタフェースであるタッチパネルドライバーは、メッセージフレームワークの post API を使用して、メッセージをサブスクライバリストに入れます。ここから、アプリケーションはメッセージフレームワークを使用して、GUIX が理解できるデータ形式でイベントを取得することができます。

Synergy のメッセージフレームワーク

SSP は、スレッド間通信（ITC）用のメッセージフレームワークを使用して、ハードウェアとソフトウェア間のメッセージのパブリッシュ / サブスクライブモデルをサポートします。内部的にはメッセージのキューイングに ThreadX キュー（GUIX イベントキューとは異なる）が使用されます。これらのメッセージには、ペンアップやキーストロークなどのユーザー入力イベントに関する情報が含まれています。パブリッシャ（ハードウェアなど）は入力を検出したときにメッセージをポストし、メッセージサブスクライバはメッセージのパブリッシャを待機（例：クエリ）します。GUIX Hello World サンプルプロジェクト（https://www.renesas.com/en-us/software/D6002539.html からダウ

ンロード）では、タッチイベントメッセージを検出して送信する SF_TOUCH_PANEL_I2C 内部スレッドがパブリッシャで、gx_src_thread_entry 関数によってコールされた thermostat_app 関数内でイベントをクエリするために Synergy によって作成されたメッセージインスタンスがサブスクライバです。

モジュールガイドプロジェクトは、メッセージフレームワークの 2 つの API、pend と bufferRelease を使用します。pend API はキューからメッセージを取得し、bufferRelease API はメッセージを取得するために割り当てられたメモリを解放してシステムに返します。複数のサブスクライバが存在する場合、bufferRelease はリスナのカウンティングセマフォを使用して、メッセージメモリをいつ解放できるか（リスナの残りがゼロ）を判断します。このプロジェクトの例ではリスナが 1 つしかないため、メッセージに対して最初に bufferRelease が呼び出されたときにメモリが解放されます。

```
ssp_err_t SF_MESSAGE_Pend(sf_message_ctrl_t const * const p_ctrl,
                          TX_QUEUE const * const p_queue,
                          sf_message_header_t ** pp_buffer,
                          uint32_t const wait_option);
```

例：

```
sf_message_header_t * p_message = NULL;
err = g_sf_message0.p_api->pend(g_sf_message0.p_ctrl,
                                &main_thread_message_queue,
                                (sf_message_header_t **) &p_message,
                                TX_WAIT_FOREVER);

ssp_err_t SF_MESSAGE_BufferRelease(sf_message_ctrl_t * const p_ctrl,
                                   sf_message_header_t * const p_buffer,
                                   sf_message_release_option_t const option);
```

例：

```
err = g_sf_message0.p_api->bufferRelease(g_sf_message0.p_ctrl,
               (sf_message_header_t *) p_message,
                SF_MESSAGE_RELEASE_OPTION_FORCED_RELEASE);
```

このように、アプリケーションは、メッセージの取得元がどのハードウェアであるかについて考慮する必要がありません。これにより、たとえばアプリケーションがキーパッドのタッチスクリーンを切り替える場合など、ハードウェアの変更に対するアプリケーションロジックの修正が容易になります。

SF メッセージヘッダーの構造は以下のように定義されます。

```
typedef struct st_sf_message_header
{
     union
    {
        uint32_t  event;
        struct
        {
            uint32_t  class_code     : 8;      /* Event class code            */
            uint32_t  class_instance : 8;      /* Event class instance number */
            uint32_t  code           : 16;     /* Event code                  */
        }event_b;
    };
```

```
} sf_message_header_t;
```

アプリケーションは、event_b クラスの class_code および code フィールドを調べ、発生したイベントと GUIX キューに送信するメッセージを判断します。Hello World サンプルでは、以下のようにタッチスクリーンドライバーがメッセージ event_b フィールドを設定してイベントを定義しています。

```
p_ctrl->p_payload->header.event_b.class_code = SF_MESSAGE_EVENT_CLASS_TOUCH;
p_ctrl->p_payload->header.event_b.code = SF_MESSAGE_EVENT_NEW_DATA;
p_ctrl->p_payload->header.event_b.class_instance = p_ctrl->event_class_instance;
```

ここで、p_ctrl はタッチスクリーンパネルのインスタンスです。p_ctrl->p_payload->header はタッチスクリーンパネルからの SF メッセージのインスタンスで、上の構造体に示されている event_b データ型が含まれています。ハードウェアによっては、インスタンスが複数存在する可能性がありますが、タッチスクリーンパネルには 1 つだけ存在するため、インスタンスはゼロに設定されます。

次にアプリケーションは、以下のようにどのハードウェアイベントが GUIX イベントに対応するかを判別します。

```
switch (p_touch_payload->event_type)
    {
    case SF_TOUCH_PANEL_EVENT_DOWN:
        gx_event->gx_event_type = GX_EVENT_PEN_DOWN;
        break;
    case SF_TOUCH_PANEL_EVENT_UP:
        gx_event->gx_event_type = GX_EVENT_PEN_UP;
        break;
    case SF_TOUCH_PANEL_EVENT_HOLD:
    case SF_TOUCH_PANEL_EVENT_MOVE:
        gx_event->gx_event_type = GX_EVENT_PEN_DRAG;
        break;
    case SF_TOUCH_PANEL_EVENT_INVALID:
        send_event = false;
        break;
    default:
        break;
```

ここで p_touch_payload はタッチスクリーンの SF メッセージヘッダー（p_message, など）です。これでアプリケーションは GUIX が理解できる実際のイベントタイプ、GX_EVENT データ型を作成でき、次のフィールドに入力できるようになりました。

```
/** Send event to GUIX */
        gx_event->gx_event_sender = GX_ID_NONE;
        gx_event->gx_event_type = GX_EVENT_PEN_UP;
        gx_event->gx_event_target = 0;
        gx_event->gx_event_display_handle = 0;
        gx_event->gx_event_payload.gx_event_pointdata.gx_point_x = p_touch_paylo
ad->x;
        gx_event->gx_event_payload.gx_event_pointdata.gx_point_y = p_touch_paylo
ad->y;
```

SF_TOUCH_PANEL_EVENT_DOWN,、SF_TOUCH_PANEL_EVENT_UP、およびその他のタッチスクリーンパネルイベントはタッチスクリーンヘッダーファイルで定義されます。また、タッチスクリーンドライバーは、イベントタイプ以外の何らかの value(s) をアプリケーションに通知する必要があるため、ペイロードフィールドのみ定義します。タッ

チスクリーンイベントの場合、タッチスクリーンペイロードには X 座標と Y 座標が含まれます。GX_EVENT にはペイロードのフィールドの共用体が含まれます。この場合、GX_EVENT gx_event_payload フィールドは、X 座標と Y 座標の 2 つの値を保持する GX_POINT gx_event_pointdata を使用します。

これでアプリケーションは、このイベントを GUIX イベントキューに送信し、GUIX は gx_system_event_send API をコールすることでこれを処理できるようになりました。

GUIX はイベントキューを定期的にチェックし、キューに追加されたばかりのイベントを確認します。タッチスクリーンパネルの場合、X、Y 座標を使用して現在ルートウィンドウにアタッチされているウィンドウウィジェットを検出し、そのウィンドウに含まれるすべての子ウィジェットから、この座標を含む Z オーダーが最も高いものを探します。次に、GUIX は通常、子ウィジェットの親イベントハンドラがその後のイベント処理を完了するのを待機します。

メッセージフレームワークの完全な説明は、『*Messaging Framework Module Guide*』（https://www.renesas.com/en-eu/doc/products/renesas-synergy/apn/r11an0096eu0100-synergy-message-fw-mod-guide.pdf）に記載されています。)

画面管理

ルートウィンドウと子ウィンドウ

表示されている各キャンバスにはルートウィンドウが 1 つあります。ルートウィンドウはディスプレイのすべてのウィンドウとウィジェットのコンテナです。

画面のアタッチとデタッチ

新しい画面が親画面の子である場合、アプリケーションは gx_widget_attach を使用して新しい画面をアタッチし、gx_widget_detach を使用して古い画面をデタッチする必要があります（順序は重要ではありません）。親画面に戻った場合、アプリケーションは親画面をアタッチしませんが、親画面に対して gx_window_show をコールします。画面切り替えの単純なアルゴリズムは以下のとおりです。

```
void ToggleScreen(GX_WIDGET *new_win, GX_WIDGET *old_win)
{
    if (!new_win->gx_widget_parent)
    {
        gx_widget_attach(root, new_win);
    }
    else
    {
        gx_widget_show(new_win);
    }
    gx_widget_hide(old_win);
}
```

次のコードでは、window1 から window2 に切り替えるための window1 のボタンウィジェットがクリックされたことを GUIX が検出しました。このため、新しいウィンドウ（window2）をルートウィンドウにアタッチし、古いウィンドウ（window1）をデタッチしています。

```
UINT window1_handler(GX_WINDOW *widget, GX_EVENT *event_ptr)
{

    gx_window_event_process(widget, event_ptr);

    switch (event_ptr->gx_event_type)
    {
        case GX_SIGNAL(ID_WINDOWCHANGER, GX_EVENT_CLICKED):
        if(button_enabled)
        {
```

```
            show_window((GX_WINDOW*)&window2, (GX_WIDGET*)widget, true);
        }
        break;

       default:
        gx_window_event_process(widget, event_ptr);
        break;
    }
    return result;
}

static UINT show_window(GX_WINDOW * p_new, GX_WIDGET * p_widget, bool detach_old
)
{
    UINT result = gx_window_event_process(widget, event_ptr);

    switch (event_ptr->gx_event_type)
    {
      case GX_SIGNAL(ID_WINDOWCHANGER, GX_EVENT_CLICKED):

      if (!p_new->gx_widget_parent)
      {
        err = gx_widget_attach(p_window_root, p_new);
      }
      else
      {
        err = gx_widget_show(p_new);
      }

      gx_system_focus_claim(p_new);

      GX_WIDGET * p_old = p_widget;
      if (p_old && detach_old)
      {
        if (p_old != (GX_WIDGET*)p_new)
        {
            gx_widget_detach(p_old);
        }
      }

      break;

      default:
        gx_window_event_process(widget, event_ptr);
        break;
    }
    return result;
}
```

どちらも画面間を移動する方法としてまったく問題ありませんが、後者の方が複雑です。画面のアタッチとデタッチの順序は重要ではありません。GUIX は表示するウィンドウをダーティとしてマークし、アプリケーションが可視ウィ

ジェットのツリー * の修正を完了した後、そのウィンドウを再描画します。つまり、GUIX は「アタッチ」API コールの一環としてウィンドウを描画するわけではありません。

注 : GUIX は、子ウィジェット（多数）を親ウィジェット（1 つ）にリンクするため、可視オブジェクトのツリー構造リストを保持しています。ウィンドウが別のウィンドウに切り替えられると、ツリーに新しい（表示する）ウィンドウが含められ、古い（非表示にする）ウィンドウが削除されます。*

ウィジェットのアタッチとデタッチ

GUIX は、親に属するか、親にアタッチされた子ウィジェット（ウィンドウはウィジェットタイプの 1 つ）のリンクされたリストを保持しています。子ウィンドウはルートウィンドウにアタッチされます。ウィジェットはアタッチされるとリストの末尾に追加されます。このリストには最初の子と最後の子を示すマーカーがあります（循環リストではありません）。子ウィジェットを親ウィジェットにするためには、そのウィジェットが別の親の子であってはなりません。その場合、最初にその親からデタッチされます。

子ウィジェットは、デタッチされるとリストから除去されます。ただし、削除されるわけではありません。よって、引き続き使用できるため、再作成する必要なく再アタッチすることができます。リストから除去されている間、子ウィジェットの描画はできず、親はありません。ただし、このウィジェットが有効なイベントと制御ブロックを引き続き受信することはできます。

ウィジェットのデタッチと削除

ウィジェットはデタッチされると、GUIX によって維持されるウィジェットのアクティブなリストから除去されます。ただし、削除されるわけではありません。よって、引き続き使用できるため、再作成する必要なく再アタッチすることができます。リストから除去されている間、子ウィジェットの描画はできず、親はありません。ただし、このウィジェットが有効なイベントと制御ブロックを引き続き受信することはできます。

gx_widget_delete API は、ウィジェットにまだ親がある場合は、最初にウィジェットをデタッチしてから制御ブロックをクリアします。制御ブロックが動的に割り当てられた場合は、制御ブロックのメモリが解放されます。gx_widget_delete API がウィジェットに適用された場合、ウィジェットを再表示するには再作成する必要があります。

ウィジェットを作成するケース

GUIX Studio によって、開発者は GUIX API のソースコードから作成するよりもいっそう容易に画面の作成、ウィジェットの名前付け、イベントおよび描画コールバック関数の設定を行うことができます。GUIX Studio では、開発者は、どのウィジェットが起動時に作成されるかを選択することができます（デフォルト設定）。またはウィジェットの [Run Time Allocation] プロパティを選択して、実行時の作成を有効にすることができます。実行時の作成を有効にした場合は、アプリケーションは必要なときにウィジェットを作成する必要があります。有効にしなかった場合は、起動時にすべての「静的」ウィジェットが作成されます。そのため、「静的」ウィジェットを作成するための十分なメモリが必要です。十分なメモリがあれば、起動時にすべてのウィジェットを作成したほうがよいと考えられます。ただし、画面を多数備えているアプリケーションや容量が限られたメモリの場合は、必要に応じて実行時にウィジェットを作成しなければならないことがあります。

動的メモリ割り当てにウィジェットを指定するには、GUIX Studio で [Runtime Allocate] チェックボックスを選択します。GUIX Studio によって作成された仕様ファイルでは、ウィジェット（およびそのウィジェットのすべての子）の「style フラグ」で GX_ STYLE_DYNAMICALLY_ALLOCATED 属性を設定します。アプリケーションは、ウィジェットを作成するときに、gx_system_memory_allocator_set. で指定されたメモリアロケータを使用して、メモリを割り当てます。これには、GUIX サービスを開始する前にアプリケーションでメモリプールを設定する必要があることに注意してください。このことは、このモジュールガイドのプロジェクトで示されています。

実行時の描画

一部の画面は、実行時に必然的に描画が必要になります。このような画面として、リアルタイムで画面上の情報を更新するためのライン描画 API およびテキストライト API があります。動的な文字列作成は sprintf 関数および gx_utility_ltoa 関数を使用して実行されます。これらの関数は、実行時の値を文字列に表示するのに役立ちます。以下に画面の描画に使用する API の一部を例として示します。

```
UINT _gx_canvas_line_draw(GX_VALUE x_start, GX_VALUE y_start, GX_VALUE x_end, GX
_VALUE y_end);

UINT _gx_canvas_text_draw(GX_VALUE x_start, GX_VALUE y_start, GX_CONST GX_CHAR *
```

```
string, INT length);
```

通常、画面に対する描画は、遅延配信と呼ばれる方法によって行われます。この方法では、GUIX がウィンドウまたはウィンドウの一部の再描画が必要なタイミングを内部で管理することで、描画の効率性を高めます。

ただし、アプリケーションで即時にキャンバスに描画する場合は、アプリケーションは何らかの描画の前に gx_canvas_drawing_initiate API をコールする必要があります。描画が完了したら、アプリケーションは gx_canvas_drawing_complete をコールして、GUIX に遅延描画を再開するよう信号を送ります。

メモリ割り当ておよびメモリ割り当て解除が必要な実行時の描画は、モジュールガイドのプロジェクトにある gux_gx_src_mg_ap.c ファイルにおいて、サーモスタットのローテーションで示されています。

SSP 1.3.0/GUIX 5.3.3 では、実行時における画像の「ローテーション」またはその他の再描画を行うには、pixelmap データを圧縮しないでください。ローテーションなど、画像をさまざまな方法で再描画する必要があると予想される場合、GUIX Studio（v5.3.3.7 以降）では pixelmap リソースに対する [Compress Output] チェックボックスの選択を必ず解除します。pixelmap リソースを編集するには、(+) アイコンの選択 -> [Custom] をクリック -> グラフィック項目をダブルクリック - > [Compress Output] を選択解除（デフォルトで選択されています）

の順に実行し、Pixelmaps バーを開きます。

GUIX アプリケーションでのメモリ割り当て

メモリの割り当て

GUIX では動的メモリ割り当てがサポートされています（ヒープメモリなど）。gx_system_memory_allocator_set API によって、アプリケーションはメモリ割り当てとメモリ解放サービスを割り当てることができます。この API は、プログラムの起動時、gx_system_initialize, の後、および動的メモリ割り当てを必要とする GUIX サービスの前にコールされます。

実行時のメモリ割り当ておよびメモリ割り当て解除サービスが必要な GUIX サービスには、次のものがあります。

- GUIX ランタイム環境への外部ストレージからのバイナリリソースのロード
- ソフトウェアランタイム jpeg 画像デコーダ
- ソフトウェアランタイム png 画像デコーダ
- GX_STYLE_TEXT_COPY. でのテキストウィジェットの使用
- 実行時の pixelmap サイズ変更関数およびローテーションユーティリティ関数
- 実行時の画面およびウィジェット制御ブロック割り当て

動的メモリ割り当ては、このモジュールガイドのプロジェクトにおいて pixelmap ローテーションで必要です。

動的割り当てによって、アプリケーションは実行時にフラッシュドライブなどの不揮発性のメモリまたは URL ソースから、フォント、言語および画像などのリソースをインポートできます。ほとんどの GUIX アプリケーションは、動的割り当ては必要がない程度にサイズが小さくなっています。GUIXリソースは、GUIX Studioで作成されたpixelmapを使用して、コンパイル時にロードされ、静的にリンクすることができます。GUIX Studio では、jpeg から pixelmap フォーマットにリソースをデコードします。

ウィジェットの作成：静的と実行時

メモリのサイズが大きく、制御ブロックがすべて静的に割り当てられる場合、たとえば、GUIX Studio でウィジェットの [Runtime Allocated] プロパティが選択されていない場合は、プログラムの起動中にウィジェットを作成したほうがよいと考えられます。GUIX 画面の作成とは、単に仕様ファイルに書き込まれたデータを使用して、初期パラメータを設定することです。これらのどのパラメータも、画面が作成された後、任意の時点で変更できます。

アプリケーションによる RAM の使用効率を高める場合は、GUIX Studio で一部またはすべての画面ウィジェットの [Runtime Allocated] プロパティを設定します。アプリケーションで、制御ブロックの動的割り当てに使用するメモリプールを設定します。これは、メモリ割り当て関数ポインタとメモリ解放関数ポインタを入力として受け取る gx_system_memory_allocator_set API をコールすることで行われます。これらの関数は、一般的にはユーザー定義の関数で、通常は ThreadX tx_byte_allocate API および tx_byte_release API をそれぞれ使用します。ただし、他のメモリ割り当てサービスも使用することができます。

動的ウィジェット割り当ては、GUIX Studio 仕様ファイルで処理されるのが一般的です。仕様ファイルでは、gx_studio_widget_create は、動的割り当てのために割り当てられたこれらのウィジェット制御ブロックを動的に割り当てます。以下の例では、医療画面ウィジェットが style シンボル GX_STYLE_DYNAMICALLY_ALLOCATED:

によって指定された動的割り当てのために指定されています。

```
GX_CONST GX_STUDIO_WIDGET meds_screen_define =
{
    "meds_screen",
    GX_TYPE_TEMPLATE,                          /* widget type                  */
    ID_MEDS_SCREEN,                            /* widget id                    */
    GX_STYLE_BORDER_THIN|GX_STYLE_DYNAMICALLY_ALLOCATED,   /* style flags    */
    GX_STATUS_ACCEPTS_FOCUS,                   /* status flags              */
```

次に、以下のように仕様ファイルのウィジェット作成ハンドラが、ウィジェットを動的に割り当てる必要があるかどうかを確認します。

```
static GX_WIDGET *gx_studio_nested_widget_create(GX_BYTE *control,
            GX_CONST GX_STUDIO_WIDGET *definition, GX_WIDGET *parent)
{
    UINT status = GX_SUCCESS;
    GX_WIDGET *widget = GX_NULL;

    while(definition && status == GX_SUCCESS)
    {
                if (definition->style & GX_STYLE_DYNAMICALLY_ALLOCATED)
                {
                    status = gx_widget_allocate(&widget,
                definition -> control_block_size);
                    ...
```

アプリケーションが実行時に gx_widget_allocate もコールできることに注意してください。

このウィジェットで gx_widget_delete がコールされると、GUIX はアプリケーションによって指定されたメモリ解放関数をコールして、制御ブロックメモリを解放します。数百の画面を備えたアプリケーションと制限のある RAM は、ほとんどの場合、この方法で設定されます。

ウィジェットが動的に割り当てられると、子ウィジェット用の制御ブロックが自動で動的に割り当てられます。子ウィジェットにこのフラグを設定することはできません。またその必要もありません。e^2 studio の任意のレベルでこのフラグを設定すると、動的に割り当てられた親の子もすべて動的に割り当てられます。

複数のキャンバスとレイヤー

Synergy の GUIX は、1 つの「シンプル」キャンバスに制限されます。シンプルキャンバスとは、アプリケーションによって使用されるオフスクリーンの描画領域です。GUIX はシンプルキャンバスに対して直接操作を行いませんが、アプリケーションはシンプルキャンバスを使用して複雑な描画をオフスクリーンバッファにレンダリングしてから、必要に応じてこのオフスクリーンバッファを使用して、表示されているキャンバスをリフレッシュします。Synergy では現在複数のキャンバスはサポートされていません。ほとんどの GUIX アプリケーションは、複数のキャンバスが必要がない程度にシンプルになっています。複数のキャンバスを使用するメリットとしては、1 つ以上のキャンバスを高速メモリに割り当てて、性能要件を満たすことができる点が挙げられます。複数のキャンバスは、フェードインやフェードアウトのような特殊効果にも使用できます。

複数のグラフィックスレイヤー（複数の重なったフレームバッファ）もサポートされていません。

タイマ

gx_system_timer start API は、GUIX サービスが初期化されて、実行されるとウィジェットでコールされます。入力には、アプリケーションによって定義された、このタイマのタイマ ID（0 は使用不可）などがあります。タイマを停止するには、同じウィジェットで gx_system_timer_stop API をコールします。0 以外のタイマ ID が指定されると、GUIX はこのウィジェットにアタッチされたすべてのタイマのなかから、指定された ID と一致する ID のタイマを検索します。タイマ ID に 0 が指定されると、GUIX は指定したウィジェットのすべてのタイマを停止し、デタッチします。

タイマの起動時に、GUIX のタイマの解放リストから使用可能なタイマが割り当てられて、アクティブタイマのリストに追加されます。タイマが停止すると、そのタイマはタイマのアクティブリストから解放リストに戻されます。

タイマが期限切れになると、GUIX はそのウィジェット（タイマのオーナー）のために GX_EVENT_TIMEOUT を送信します。そのウィジェットのイベントハンドラが、タイムアウトイベントを処理し、必要なタスクを実行する必要があります。タイマは、最大で gx_system_timer_start API に指定された回数までリセットされます。

GUIX は、特定の視覚効果とタイムアウト期限切れイベントに ThreadX タイマを使用します。その 2 つの例として、フェードアウトとスプライトアニメーションがあります。

GUIX および GUIX Studio のバージョン

GUIX ライブラリおよび Studio のバージョンは、次のように番号が付けられます。GUIX ライブラリには、major.minor.service_pack のバージョン情報があります。よって、GUIX ライブラリは常に 5.3.3 または 5.4.0 となり、3 つのフィールドで構成されます。Studio バージョン番号は、<GUIX_LIB_VERSION>.Studio Revision です。Studio には、Studio のビルドの一部として GUIX ライブラリが必要であることに注意してください。このようにして、GUIX ウィジェットを Studio のターゲットとなるビューパネル内でレンダリングします。よって、5.3.3 ライブラリに基づいた最初の Studio リリースは、5.3.3.0 です。5.3.3 ライブラリに基づいた最新の Studio リリースは、5.3.3.7 です。次期 5.4.0 ライブラリに基づいた最初の Studio リリースは、5.4.0.0 です。このように、Studio リリースは、GUIX ライブラリのリリースにフィールドが 1 つ追加された形になります。

SSP 1.2.0 および 1.2.1 から SSP 1.3.0 への GUIX プロジェクトの移植

- SSP 1.2.0 または 1.2.1 を使用して以前にビルドした GUIX プロジェクトを移植するには、以下の手順を実行します。
  a) GUIX Studio 5.3.3.7 をインストールします。
  b) GUIX Studio 5.3.3.7 で、1.2.0/1.2.1 で GUIX プロジェクトに使用した gxp プロジェクトファイルを開きます。
  c) [Configure Display] 画面で GUIX ライブラリのバージョンを更新し（メインディスプレイウィンドウを右クリックします）、5.3.3 を選択します。
  d) 新しい仕様ファイルとリソースファイルを生成し、これらが src フォルダまたはサブフォルダに自動的に書き込まれていない場合は、プロジェクトファイルにコピーします。
  e) Synergy コンフィギュレータで SSP 1.3.0 を選択し、[Generate Project Content] をクリックします。
  f) プロジェクトをビルドします。
- 未定義の dlist_start と dlist_indirect を参照しているためにコンパイルエラーが発生する場合は、synergy\ssp\src\framework フォルダ内の tes および sf_tes_2d_draw フォルダを削除します。次に、[Generate Project Content] と [Build project] アイコンをクリックして、プロジェクトファイルを再生成します。
- 誤ったバージョンの GUIX が GUIX Studio で設定されている場合は、以下の未定義の GUIX 関数を参照しているためにコンパイルエラーが発生する可能性があります。
  - _gx_dave2d_glyph_8bit_draw
  - _gx_dave2d_glyph_4bit_draw
  - _gx_dave2d_glyph_1bit_draw

仕様ファイルとリソースファイルのヘッダーを確認し、使用した GUIX Studio のバージョンを確認します。任意の GUIX ソースファイルで関数ヘッダーを確認すると（そのファイルへのアクセス権があることが前提です）、GUIX バージョンが 5.3.3 であるかどうかを確認できます。

GUIX GX_SRC フレームワークモジュールの動作に関する重要な注意事項と制限事項

- Synergy の GUIX は、4bpp（グレースケール）または 1bpp（モノクロ）をサポートしていません。
- SF_EL_GX は、3 つ以上のフレームバッファを持つシステムをサポートしていません。
- SF_EL_GX は、GUIX キャンバスシステムを 1 つだけサポートします。
- SF_EL_GX は、ディスプレイモジュールのグラフィックスレイヤーを 1 つだけ利用します。
- このモジュールを使用する際のその他の制限事項については、SSP の最新のリリースノートを参照してください。

4.3.5.4 アプリケーションへの GUIX GX_SRC フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用して GUIX GX_SRC フレームワークモジュールをアプリケーションに組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

GUIX GX_SRC フレームワークモジュールをアプリケーションに追加するには、[Thread Stack Panel] で *GUIX on gx* コンポーネントの下にある [Add GUIX Source] ブロックをクリックし、[New] を選択します。

GUIX GX_SRC フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
GUIX Source	Threads	New Stack> X-Ware> GUIX> GUIX Source

次の図に示すように、GUIX GX_SRC フレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

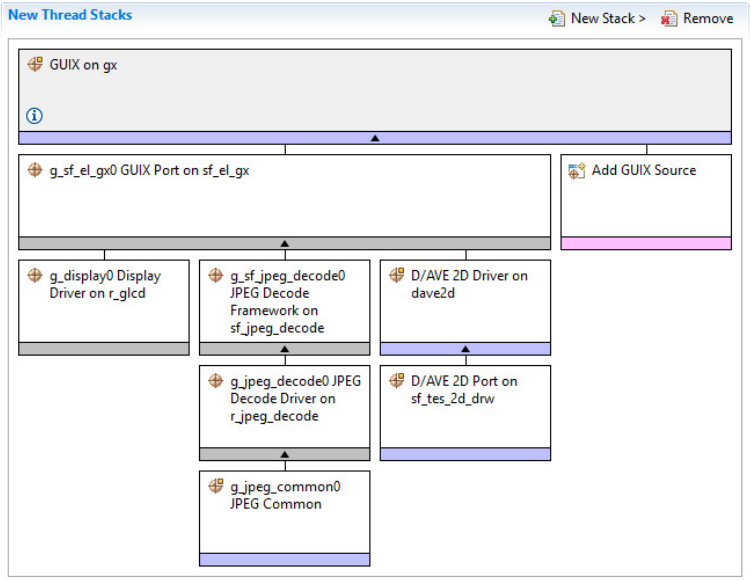


図 315:GUIX GX_SRC フレームワークモジュールのスタック

4.3.5.5 GUIX GX_SRC フレームワークモジュールの構成

ユーザーは必要な動作に合わせて GUIX GX_SRC フレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

GUIX GX_SRC フレームワークの構成設定

ISDE Property	Value	Description
GUIX Stack Size		GUIX internal thread stack size in bytes. Must be greater than zero or empty.

ISDE Property	Value	Description
GUIX System Timer (Milliseconds)		GUIX system timer. Must be a multiple of TX_TIMER_TICKS_PER_SECOND or empty.
GUIX Timer Rate		GUIX timer rate as a multiple of the ThreadX tick interrupt rate. Must be greater than zero or empty.
GUIX Ticks Per Second		Useful for application logic, must correlate with the GX_SYSTEM_TIMER_TICKS definition. This is not used by GUIX but for user applications. Must be greater than zero or empty.
Disable Multithread Support	Yes, No Default: No	If your application has only one thread which utilizes the GUIX API services, say yes to reduce system overhead.
Disable UTF8 Support	Yes, No Default: No	GUIX disables UTF8 support if you say yes.
GUIX Event Queue Size		Size of GUIX Event Queue Size. Must be greater than zero or empty.
GUIX Thread Priority		Priority of GUIX Internal Thread. The value must be between 0 to 31.
GUIX Thread Time Slice	10	Time Slice value of GUIX Internal Thread. The value must be between 0(TX_NO_TIME_SLICE) to 0xFFFFFFFF.
Use User Data Field in GX_WIDGET Structure	Yes, No Default: No	GUIX allows users to use gx_widget_user_data member in GX_WIDGET Structure if you say yes.

ISDE Property	Value	Description
Show linkage warning	Enabled, Disabled Default: Enabled	Select whether or not to show linkage warning.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

GUIX GX_SRC フレームワークモジュールのクロック構成

GUIX GX_SRC モジュールは、論理モジュールであるため、ARM Cortex-M コア SysTick タイマ設定を除くハードウェア設定は不要です。

GUIX GX_SRC フレームワークモジュールのピン構成

GUIX GX_SRC モジュールは、論理モジュールであるため、ピン設定は不要です。

4.3.5.6 アプリケーションでの GUIX GX_SRC フレームワークモジュールの使用

アプリケーションで GUIX フレームワークモジュールを使用する際の一般的な手順は次のとおりです（GUIX ドライバーのインスタンスがg_sf_el_ux0で、LCDのSPIドライバーのインスタンスがg_rspi_lcdc):であると想定しています)。

手順 1. GUIX で動的割り当てに対してメモリプールを作成します。

手順 2. gx_system_initialize 関数で GUIX を初期化します。

手順 3. gx_system_memory_allocator_set API を使用して、GUIX で使用するメモリローテーションサービスおよびメモリ解放サービスを設定します。

手順 4. open API（g_sf_el_ux0.p_api -> open）を使用して、GUIX ドライバーを初期化します。

手順 5. gx_studio_display_configure API を使用して、メインディスプレイを作成および初期化します。この API は、入力パラメータの 1 つに、SF_EL_GX ドライバーインスタンスの open 関数を使用します。

手順 6. SF_EL_GX ドライバーの canvasInit API でキャンバスのメモリアドレスを初期化します。

手順 7. GUIX Studio リソースファイルで定義されたすべてのウィジェットを循環します。
gx_studio_named_widget_create API を使用してルートウィンドウと各ウィジェットを作成します。

手順 8. gx_widget_show API を使用してルートウィンドウを表示します。

手順 9. gx_system_start API で GUIX システムを起動します。

手順 10. SF_EL_GX ドライバーの open API を使用して、SPI ドライバーを開いて LCD を初期化します。

手順 11. ILI9341V_Init 関数で LCD ディスプレイを設定します。

以降の手順では、ハードウェアからのイベントの有無に関するメッセージフレームワークインスタンス g_sf_message0 の確認、GUIX サービスによるイベントの処理、および画像表示の更新を行うループについて説明します。

手順 12. g_sf_message0 インスタンスの pend API を使用して、メッセージフレームワークに対してハードウェアからのメッセージを照会します。

手順 13. メッセージによってタッチイベントが発生したことが示された場合は、ssp_touch_to_guix 関数を使用してタッチイベントを GUIX イベントに処理します。

手順 14. g_sf_message0 インスタンスの bufferRelease API を使用して、画像を再描画するのに必要なメモリを解放します。

手順 15. gx_system_event_send API を使用して、GUIX エンジンにイベントを送信し、更新した画像をレンダリングします。

これらの一般的な手順を、次の図の通常の動作フローに示します。

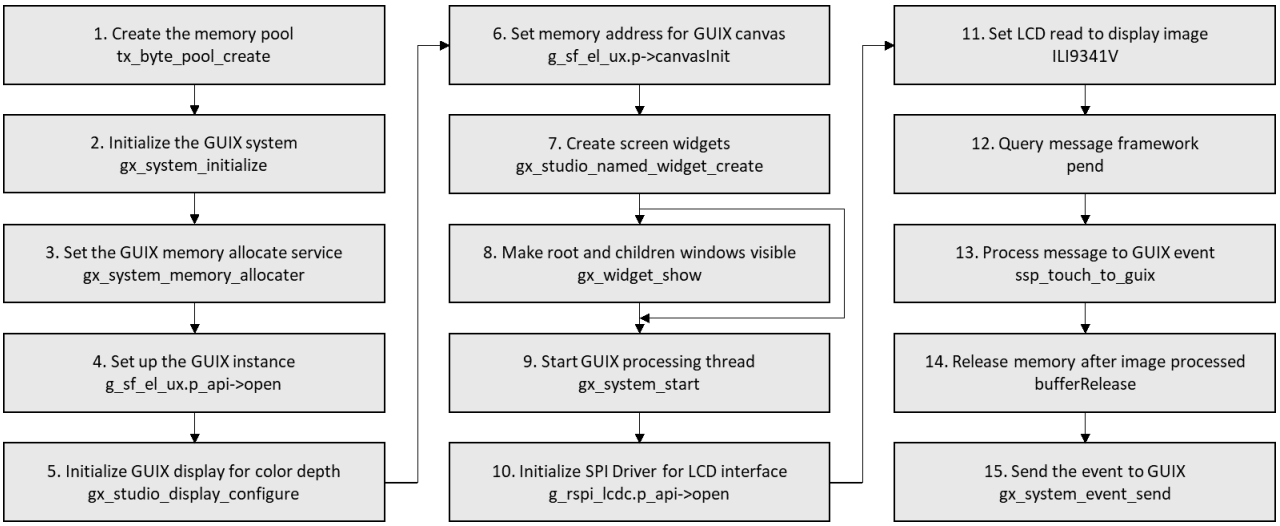


図 316:通常の GUIX GX_SRC フレームワークモジュールアプリケーションのフロー図

GUIX Studio プロジェクトには、このプロジェクトのメインウィンドウのイベント関数で使用するコールバック関数が含まれます。このコールバック thermos_screen_event_handler, は、GUIX によって呼び出されると、画像内のサーモスタットの針を調整する方法を算出し、その情報を GUIX に戻し、ユーザーのタッチイベントに基づいて更新された画像をレンダリングする方法を伝えます。

1) GUIX からの event_ptr の入力に基づいてイベントタイプを判断します。
2) gx_circular_gauge_angle_get API を使用して、GUIX から現在の針の角度データを取得します。
3) gx_window_event_process API を使用して、GUIX にイベントの更新情報を送信します。

GUIX Studio プロジェクトには、メイン画面の「+」および「-」ボタンの描画関数で使用するコールバック関数が含まれています。このコールバック custom_pixelmap_button_draw, は、GUIX によって呼び出されると、押されたボタンに応じて「+」または「-」を追加するボタンを再描画します。この関数自体は重要な機能ではありませんが、描画関数のコールバックが GUIX でどのように動作するのかを示しています。

4.3.6 sf_el_lx_nor 上の Port LevelX フレームワーク

Port LevelX フレームワークは、Express Logic 社の LevelX NOR コンポーネントによって要求されたドライバー API（セクターのリード、セクターのライト、ブロックの消去、消去されたブロックの確認）を実装します。このフレームワークは、LevelX NOR ドライバーの方式を実装するだけでなく、LevelX NOR の初期化を実行するためにフラッシュジオメトリを更新する役割も担います。sf_memory_api 実装を使用して、NOR フラッシュ上で演算を実行します。

4.3.6.1 Port LevelX フレームワークモジュールの特長

- LevelX NOR ドライバー API を実装し、NOR フラッシュメモリデバイス上で演算を実行
- NOR フラッシュジオメトリの設定

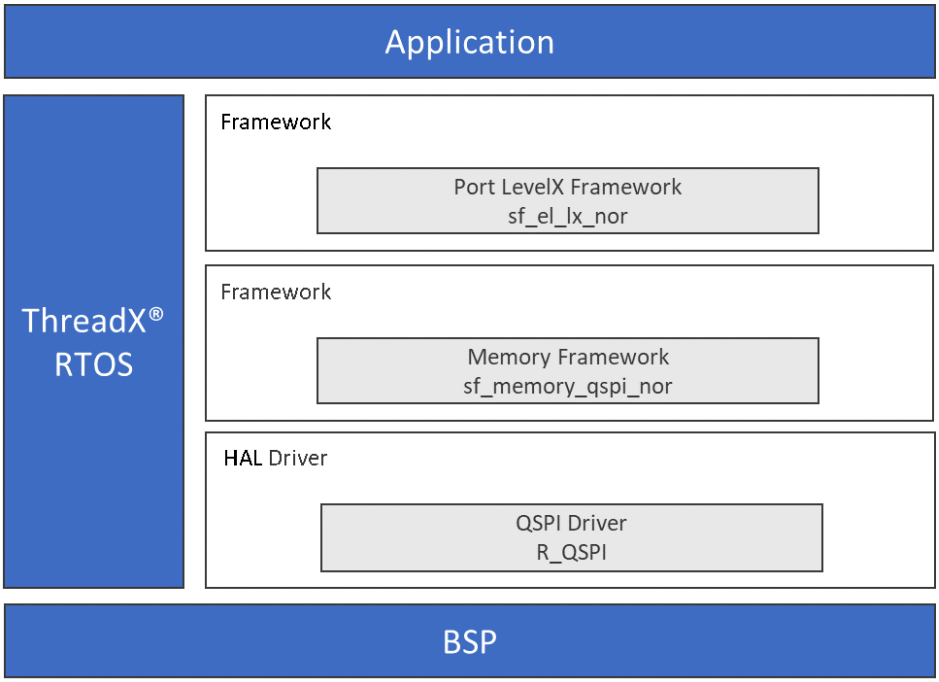


図 317:**Port LevelX フレームワークモジュールのブロック図**

4.3.6.2 Port LevelX フレームワークモジュール API の概要

Port LevelX フレームワークモジュールは、モジュールに対するオープン、リード、ライト、クローズを行う API を定義します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

Port LevelX フレームワークモジュール API の要約

Function Name	Example API Call and Description
SF_EL_LX_NOR_Open	g_sf_el_lx_nor0.p_api->SF_EL_LX_NOR_Open(g_sf_el_lx_nor0.p_ctrl, g_sf_el_lx_nor0.p_cfg); Initializes lower-level driver initialization function.

Function Name	Example API Call and Description
SF_EL_LX_NOR_Read	g_sf_el_lx_nor0.p_api->SF_EL_LX_NOR_Read (g_sf_el_lx_nor0.p_ctrl, p_flash, p_dest, word_count); This is responsible for reading a specific sector in a specific block of the NOR flash. All error checking and correcting logic is the responsibility of this service.
SF_EL_LX_NOR_Write	g_sf_el_lx_nor0.p_api->SF_EL_LX_NOR_Write(g_sf_el_lx_nor0.p_ctrl, p_flash, p_src, word_count); This is responsible for writing a specific sector into a block of the NOR flash. All error checking is the responsibility of this service.
SF_EL_LX_NOR_BlockErase	g_sf_el_lx_nor0.p_api->SF_EL_LX_NOR_BlockErase(g_sf_el_lx_nor0.p_ctrl, block, erase_count); This is responsible for erasing the specified block of the NOR flash.
SF_EL_LX_NOR_BlockErasedVerify	g_sf_el_lx_nor0.p_api->SF_EL_LX_NOR_BlockErasedVerify(g_sf_el_lx_nor0.p_ctrl, block); This is responsible for verifying that the specified block of the NOR flash is erased.
SF_EL_LX_NOR_Close	g_sf_el_lx_nor0.p_api->SF_EL_LX_NOR_Close(g_sf_el_lx_nor0.p_ctrl); This is responsible for closing the driver properly.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_ASSERTION	p_ctrl or p_cfg in NULL.
SSP_ERR_ALREADY_OPEN	Driver is already in OPEN state.
SSP_ERR_INVALID_ARGUMENT	Requested range can't fit in the flash address range.
SSP_ERR_NOT_OPEN	Driver not in OPEN state for writing.
SSP_ERR_NOT_ERASED	The block in not erased properly.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.3.6.3 Port LevelX フレームワークモジュールの動作の概要

Port LevelX フレームワークは単なる抽象インタフェースであり、直接の関数コールの代わりに関数ポインタを使用します。関数は LevelX または FileX と、sf_memory_qspi_nor. などの SSP メモリインタフェース実装の間でコールされます。メモリ適応ドライバー（sf_memory_qspi_nor, など）には Port LevelX フレームワークを使用してアクセスします。これらのドライバーは、データ I/O 操作の実行に必要なデバイス固有のコードを提供します。

Port LevelX フレームワークモジュールの動作に関する重要な注意事項と制限事項

なし。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.6.4 アプリケーションへの Port LevelX フレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに Port LevelX フレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

Port LevelX フレームワークをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。(Port LevelX フレームワークのデフォルト名は g_sf_el_lx_nor0. です。この名前は、対応する [Properties] ウィンドウで変更できます。)

Port LevelX フレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_el_lx_nor0 Port LevelX Framework on sf_el_lx_nor	Threads	New Stack> Framework> LevelX> Port LevelX Framework on sf_el_lx_nor

次の図に示すように、sf_el_lx_nor 上の Port LevelX フレームワークがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

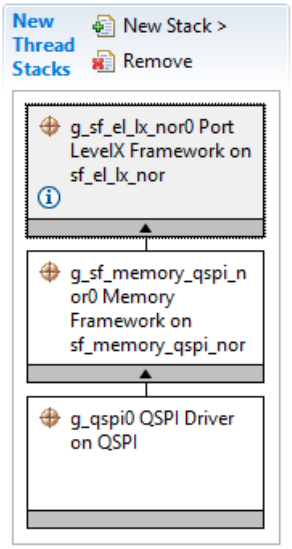


図 318:Port LevelX フレームワークモジュールスタック

4.3.6.5 Port LevelX ポートフレームワークモジュールの構成

ユーザーは必要な動作に合わせて、Port LevelX フレームワークモジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

sf_el_lx_nor での Port LevelX フレームワークモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_sf_el_lx_nor0	Module name.
Event Callback	NULL	Name of the function to call when an event occurs.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

Port LevelX フレームワークのローレベルモジュールの構成設定

通常、ローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

sf_memory_qspi_nor 上のメモリフレームワークの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_sf_memory_qspi_nor0	Module name.
Write of Erase Timeout (in ticks)	30000	Timeout ticks for waiting on write or erase to complete.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_qspi の QSPI HAL モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build.
Name	g_qspi0	Module name.

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

Port LevelX フレームワークモジュールのクロック構成

Port LevelXフレームワークモジュールは、PCLKAをクロックソースとして使用するQSPIペリフェラルを使用します。

実行時にクロック周波数を変更するには、CGC インタフェースを使用します。

Port LevelX フレームワークモジュールのピン構成

Port LevelX フレームワークモジュールを使用するには、必要に応じて QSPI ペリフェラルのポートピンを設定する必要があります。次の表では、ISDE [Configuration] ウィンドウでのピンの選択方法を示します。

sf_memory_qspi_nor 上のメモリフレームワークモジュールのピンの選択

Resource	ISDE Tab	Pin selection Sequence
QSPI	Pins	Select Peripherals > Storage:QSPI QSPI0

4.3.6.6 アプリケーションでの Port LevelX フレームワークモジュールの使用

アプリケーションで Port LevelX フレームワークモジュールを使用する際の一般的な手順は次のとおりです。

1) lx_nor_flash_initialize API 関数を使用してモジュールを初期化します（g_common_init は自動的にこれをコールします）。
2) LevelX NOR API 関数 lx_nor_flash_open を使用して I/O 操作のためにモジュールを開きます（FileX で使用している場合は、fx_media_open は自動的にメディアを開きます）。
3) 必要に応じて、SF_EL_LX_NOR_Read API 関数を使用してメディアを読み取ります（lx_nor_flash_sector_read はこの関数を自動的にコールします）。
4) 必要に応じて、SF_EL_LX_NOR_Write API 関数を使用してメディアに書き込みます（lx_nor_flash_sector_write はこの関数を自動的にコールします）。
5) SF_EL_LX_NOR_BlockErase API 関数を使用して、ブロックを消去できます。

6) SF_EL_LX_NOR_BlockErasedVerify API 関数を使用して、ブロックが消去されているかどうかについて、ブロックを確認できます。

*注 :lx_nor_flash_open のコールが成功したら、リードおよびライトだけでなくすべての LevelX NOR API を使用することができます。*これらの共通の手順を、以下の図に示す一般的な動作フローで説明します。

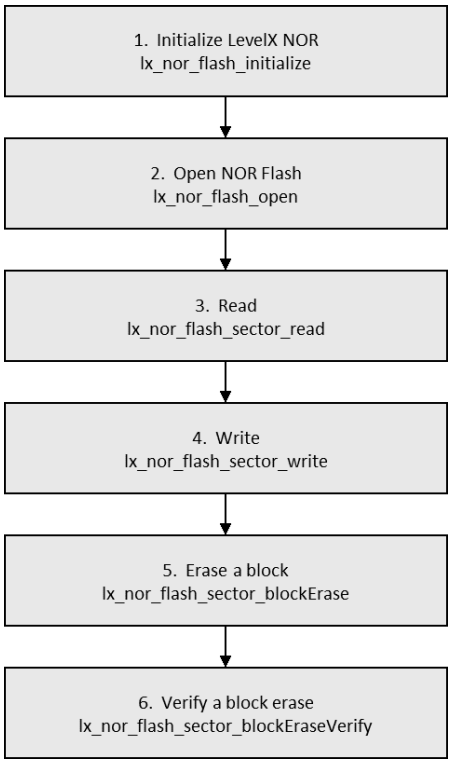


図 319:通常の Port LevelX フレームワークモジュールアプリケーションのフロー図

4.3.7 NetX ポートイーサネット

NetX および NetX Duo 用 Synergy Express Logic NetX ポート ETHER モジュール（sf_el_nx）は、SSP に組み込まれています。このモジュールの機能は、NetX および NetX Duo ソフトウェアと Synergy ハードウェアの間のやり取りを行うことです。このモジュールは、MAC ドライバー、PHY ドライバー、その他のグルーロジックおよびユーティリティ機能などを備えています。

注 : 特に明記されていない限り、NetX プロジェクトと NetX Duo プロジェクトにおけるこのモジュールの動作に違いはありません。

4.3.7.1 NetX ポート ETHER モジュールの特長

- Synergy プラットフォーム用 NetX および NetX Duo のハイレベルインタフェース
- チャネルの選択
- PHY リセットサポート

- 静的 MAC アドレス構成
- 動的 MAC アドレス構成
- 不明パケットの受信に対するコールバックの提供
- 選択可能なイーサネット割り込み優先順位
- リンクステータス監視サポート
- 構成可能な受信 / 送信バッファ記述子の数

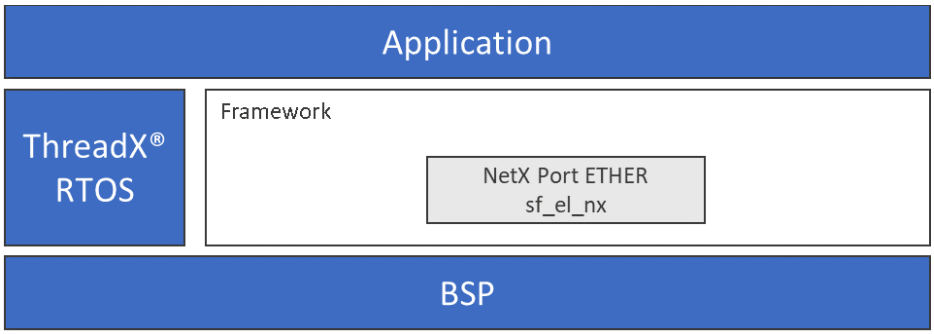


図 320:NetX ポート ETHER モジュールのブロック図

4.3.7.2 NetX ポート ETHER モジュールの API の概要

NetX ポート ETHER モジュールには狭い範囲の API があり、NetX およびモジュール自体によって使用されます。イーサネットドライバーのエントリポイント (nx_ether_driver_eth0, nx_ether_driver_eth1), イーサネット割り込みハンドラ、モジュールによって内部的に使用されるが外部から見ることのできる他の関数などが含まれます。

4.3.7.3 NetX ポート ETHER モジュールの動作の概要

NetX ポート ETHER モジュールは、Renesas Synergy ソフトウェアおよび Synergy イーサネット IP 用の NetX イーサネットドライバーの高性能なリアルタイムの実装です。

注 :NetX は ThreadX の存在を前提にしており、そのスレッド実行、サスペンション、定期タイマ、相互排除の機能に依存します。

NetX の各 IP インスタンスには、nx_ip_create サービスで指定されているデバイスドライバーによって識別されるプライマリインタフェースがあります。ネットワークドライバーは、パケット送信、パケット受信、状態と制御の要求など、NetX のさまざまな要求を処理します。

マルチホームシステムの場合、複数のインタフェース用に IP インスタンスを構成することができ、それぞれのインタフェースに対するこれらのタスクを実行するネットワークドライバーがインタフェースごとに関連付けられます。ネットワークドライバーは、メディアで発生する非同期イベントの処理も行う必要があります。メディアで発生する非同期イベントとしては、パケットの受信、パケットの送信の完了、状態の変化などがあります。NetX には、さまざまなイベントを処理するための複数のアクセス関数を含むネットワークドライバーが用意されています。これらの関数は、ネットワークドライバーの割り込みサービスルーチン部分から呼び出されるように設計されています。IP ネットワークのネットワークドライバーは、受信したすべての ARP パケットを nx_arp_packet_deferred_receive 内部関数に転送する必要があります。すべての RARP パケットは、nx_rarp_packet_deferred_receive 内部関数に転送される必要があります。IP パケットに関しては 2 つのオプションがあります。

IP パケットの高速なディスパッチが必要な場合は、受信 IP パケットは nx_ip_packet_receive に転送されてすぐに処理されます。これにより、NetX による IP パケットの処理性能が大きく向上します。それ以外の場合は、IP パケットは nx_ip_packet_deferred_receive. に転送されます。このサービスは、IP パケットを遅延処理キューに格納します。このキューの IP パケットは内部 IP スレッドによって処理されて、ISR の処理時間が最小限になります。

ネットワークドライバーは、割り込み処理を延期して、IP スレッドのコンテキスト外で実行することもできます。このモードでは、ISR は必要な情報を保存し、内部関数 nx_ip_driver_deferred_processing. を呼び出して、割り込みコントローラにアクノリッジする必要があります。このサービスは、割り込みを発生させる、ネットワークドライバーに対するイベント処理完了のコールバックをスケジュールするように、IP スレッドに通知します。

重要な構成プロパティ設定：

Multi-Channel

Multi-Channel は、イーサネットドライバーが適用されるインタフェースを決定します。デフォルト値の 0 を変更することが必要な場合があります。セクション 5「NetX ポート ETHER の構成」を参照してください。

Channel 0/1 PHY Reset Pin

PHY リセットが、I/O ポートピンによってサポートされています。このプロパティは、上のチャネルプロパティの値に基づきます。

注 :Synergy キットごとに異なる I/O ピンが利用されています。そのため、このプロパティには、デフォルト値からの変更が必要な場合があります。

Static MAC Address Configuration

これらのプロパティは、コンパイル時に該当するチャネルインタフェースのデバイス MAC アドレスを設定します。実行時に MAC アドレスを設定するには、後の [Dynamic MAC Address Configuration] コールバックプロパティの説明を参照してください。

Dynamic MAC Address Configuration

このプロパティは、実行時に MAC アドレスを設定するユーザー定義のコールバック関数を定義します（ネットワークリンクの初期化）。デフォルト値は NULL であり、MAC アドレスはチャネルの MAC アドレスの高ビットおよび低ビットの設定を使用して割り当てられます。

Unknown Ethernet Packet Receive Callback

このプロパティは、ユーザーによる未サポート / カスタムの EtherType パケットの処理を可能にするコールバックを構成します。

このコールバックを 'nx_ether_custom_packet_send' Ethernet API とともに使用すると、カスタムのイーサネットパケットタイプの送受信が可能です。

名前

このプロパティは、NetX ポート ETHER ドライバーのインスタンスの名前を指定します。複数のネットワークインタフェースが構成されている IP インスタンスでは、アプリケーションはドライバーのインスタンスを追加して、一意の名前を指定する必要があります（そうしないと、コンパイルエラーになります）。デフォルトの名前は g_sf_el_nx. です。

イーサネット割り込みの優先順位

このプロパティは、ドライバーの割り込み優先順位を設定します。デフォルト値は無効になっています。ドロップダウンリストでは、MCU ターゲットに基づいて有効なプライオリティと無効なプライオリティが示されます。

Link Status Monitoring

このプロパティでは、ユーザーがリンクステータスの監視方法を選択できます。[PHY Polling] と [PHY Interrupt]（LINKSTA ピンを使用）という 2 つのオプションが使用できます。

[PHY polling] は、オートネゴシエーション用の内部の監視スレッドを利用して、リンクが切断されたとき、または確立されたときにユーザーに通知します。

[PHY Interrupt] では、PHY ステータスピンをイーサネットコントローラの LINKSTA ピンに接続する必要があり、イーサネット割り込みを利用してリンクが切断されたとき、または確立されたときにユーザーに通知します。

ただし、この通知は、IP インスタンスに登録されたコールバックによって行われます。

Number of Receive/Transmit Buffer Descriptors

これらのプロパティは、パケットを受信および送信するためのバッファ記述子（BD）の数を設定します。受信 BD を初期化するときに、ドライバーは各 BD に対してパケットを割り当てます。したがって、受信 BD の数は、パケット割り当て元の IP インスタンスパケットプールを使い切らない値にする必要があります。

NetX ポート ETHER ドライバーは、受信パケットと送信パケットの両方について、パケットチェーンをサポートします（パケットがアプリケーションレイヤーでチェーンされている場合）。IP のデフォルトパケットプールペイロードのサイズを超えるパケットをネットワークで受信した場合、ドライバーは追加のパケットを割り当てて、パケットチェーンとして受信パケットを処理できます。

パラメーター チェック

これは、プロジェクトハードウェアレイヤーの各コンポーネントに対するローレベルのエラーチェックです。デフォルトでは [Default (BSP)] に設定され、これはこのプロパティが BSP の同じプロパティを継承することを意味します。

NetX ポート ETHER モジュールの動作に関する重要な注意事項と制限事項

これらの各トピックの詳細については、『NetX User Guide for the Renesas Synergy ™ Platform』および『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。

注：：NetX ソースプロパティを変更する方法は 2 つあります。ソース要素のソースコードプロパティを直接使用する方法と、ソースコードシンボルを直接定義する方法です。たとえば、物理ネットワークインタフェースの数を変更するには、NetX ソース要素または NetX Duo ソース要素の [Maximum Physical Interfaces] プロパティを設定するか、ソースコードシンボル NX_MAX_PHYSICAL_INTERFACES を直接定義することができます。いずれの場合も、NetX および NetX Duo のソースコンポーネントを組み込み、プロジェクトファイルを生成して、NetX ライブラリを再ビルドする必要もあります。

このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.7.4 アプリケーションへの NetX ポート ETHER モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX ポート ETHER モジュールを組み込む方法について説明します。

注：このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX ポート ETHER フレームワークは、ローレベルモジュールとして使用され、個別のスタックとして追加する用途では使用できません。次の図では、NetX ネットワークドライバーを追加するためにモジュールの選択が必要となる例を示しています。次の図は、モジュールのスタックを完成させる NetX ポート ETHER フレームワークの選択を示しています。

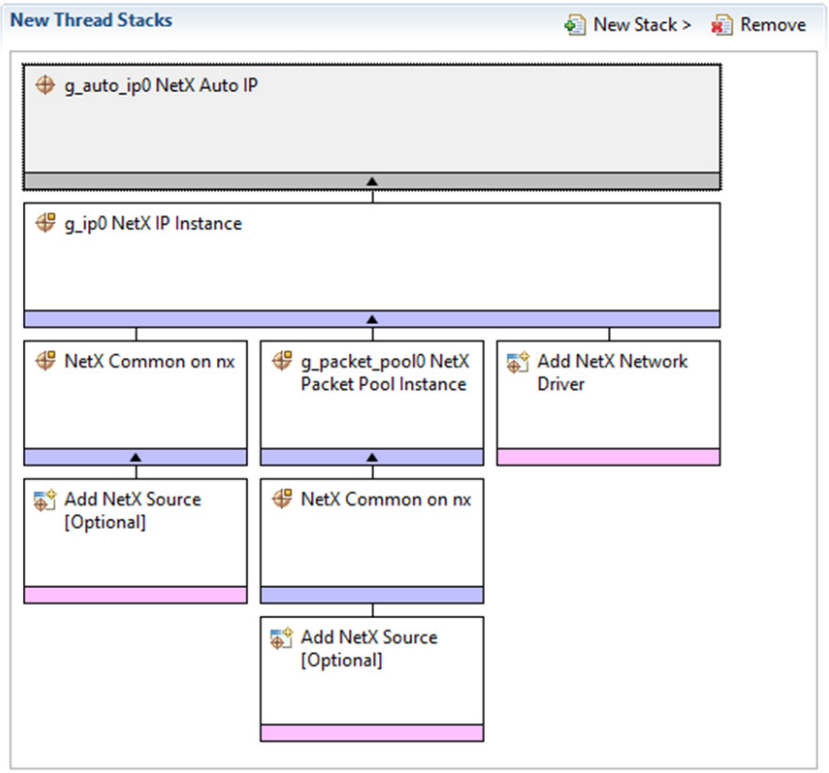


図 321:NetX ポート ETHER モジュールスタック

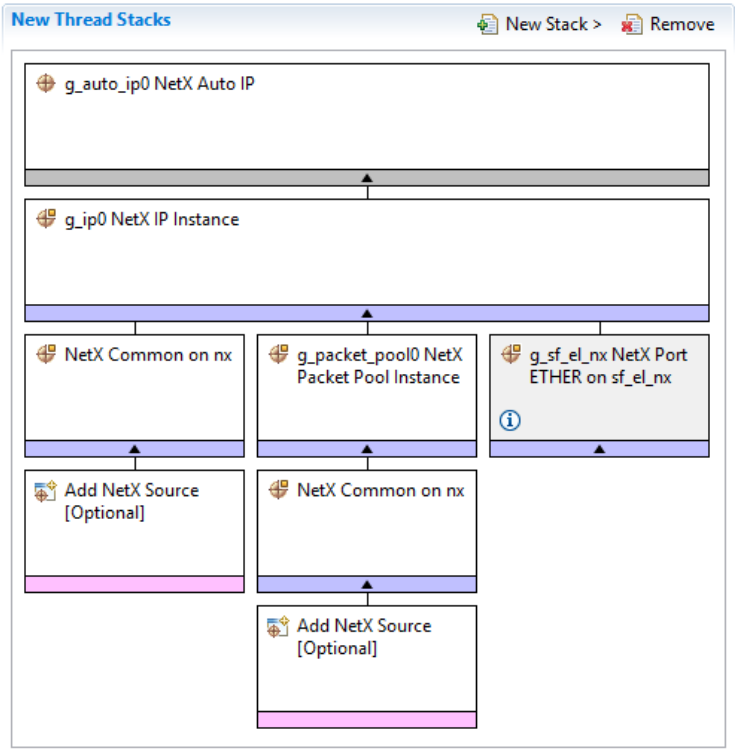


図 322:NetX ポート ETHER モジュールスタック

4.3.7.5 NetX ポート ETHER モジュールの構成

ユーザーは必要な動作に合わせて NetX ポート ETHER モジュールを構成する必要があります。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成テーブルの設定に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX ポート ETHER モジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking
Channel 0 Phy Reset Pin	IOPORT_PORT_09_PIN_03	Channel 0 Phy reset pin selection
Channel 0 MAC Address High Bits	0x00002E09	Channel 0 MAC address high bits selection
Channel 0 MAC Address Low Bits	0x0A0076C7	Channel 0 MAC address low bits selection
Channel 1 Phy Reset Pin	IOPORT_PORT_07_PIN_06	Channel 1 Phy reset pin selection
Channel 1 MAC Address High Bits	0x00002E09	Channel 1 MAC address high bits selection
Channel 1 MAC Address Low Bits	0x0A0076C8	Channel 1 MAC address low bits selection
Number of Receive Buffer Descriptors	8	Number of receive buffer descriptors selection
Number of Transmit Buffer Descriptors	32	Number of transmit buffer descriptors selection
Ethernet Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 lowest - not valid if using ThreadX) Default: Priority 12	Ethernet interrupt priority selection
Link status monitoring method	PHY Interrupt (Uses LINKSTA Pin), PHY Polling Default: PHY Polling	PHY interrupt requires LINKSTA Pin connection to PHY
Name	g_sf_el_nx	Module name
Channel	0	Channel selection

ISDE Property	Value	Description
MAC address change callback	NULL	MAC address change callback selection
Unknown packet receive callback	NULL	Unknown packet receive callback selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU ファミリを使用したプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

4.3.7.6 アプリケーションでの NetX ポート ETHER モジュールの使用

NetX ポート ETHER モジュールは NetX と組み合わせて使用する必要があります。以下は Auto-IP プロトコルの実装に使用するモジュールの使用方法の例です。他のプロトコルでも、使用のフローは同様です。NetX アプリケーションをサポートするために Synergy Software Package（SSP）が自動的に実行する手順は以下のとおりです。

1) システムを nx_system_initialize. で初期化します。
2) nx_packet_pool_create. でパケットプールを作成します。これにより、IP インスタンスとイーサネットドライバーによって使用されるデフォルトの IP パケットプールが作成されます。
3) nx_ip_create. で IP インスタンスを作成します。
4) nx_arp_enable. で ARP を有効にします。
5) nx_ip_link_status_change_notify_set. を使用して、リンクステータス変更のコールバックを有効にします。
6) 作成した IP インスタンス nx_ip_gateway_address_set. のゲートウェイ IP アドレスを設定します。
7) nx_auto_ip_create. で AutoIP インスタンスを作成します。

Auto IP スレッドのタスクをセットアップして実行するには、アプリケーションで以下の手順を直接実行します。

1) nx_ip_interface_status_check API を使用して、ネットワークリンクが有効になっていることを確認します。
2) nx_ip_interface_address_get API を呼び出して、デバイスが IP アドレスを持っていないことを確認します。
3) nx_ip_address_change_notify API をコールして、IP アドレス通知コールバックを設定します。
4) nx_auto_ip_start API を使用して、AutoIP インスタンスを開始します。
5) IP インスタンスに IP アドレスが割り当てられたことを示すフラグを IP アドレス変更コールバックが設定するのを待ちます。
6) nx_auto_ip_stop API を呼び出して、AutoIP タスクを停止します。
7) nx_ip_interface_address_get API を再び呼び出して、IP インスタンスにゼロではない IP アドレスが設定されていることを確認します。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

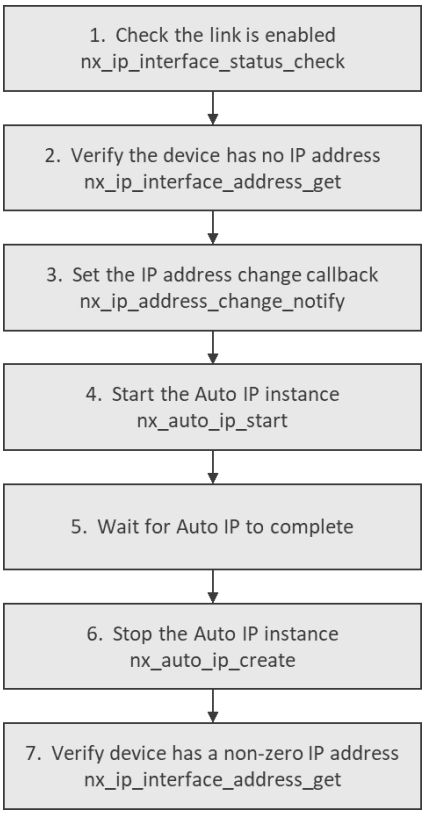


図 323:一般的な NetX ポート ETHER モジュールアプリケーションのフロー図

4.3.8 NetX および NetX Duo ソースモジュールの概要

NetX ™および NetX Duo ™ソースモジュールにより、開発者は NetX の動作を制御する一部の重要なプロパティを変更できます。Synergy コンフィギュレータ環境において、NetX または NetX Duo ソースコンポーネントを追加することで、NetX および NetX Duo ライブラリをカスタマイズしたり、デフォルト設定から値を変更したり、特定の機能を有効化または無効化したりできるようになります。これらのソースコンポーネントを追加しない場合は、ビルド済みの NetX または NetX Duo ライブラリを使用する必要があります。シンプルなソケットプログラムを除くほとんどのプロジェクトでは、通常、開発者は NetX または NetX Duo 環境をカスタマイズする必要があります。ThreadX® ソースコンポーネントは、NetX または NetX Duo ソースコンポーネントを追加するときに自動的に追加されることに留意してください。

NetX または NetX Duo ソースコンポーネントを追加しない場合、Synergy ISDE コンフィギュレータは、NetX および NetX Duo のデフォルト設定を使用して事前にビルドされたライブラリを使用します。

4.3.8.1 NetX および NetX Duo ソースモジュールの API の概要

NetX または NetX Duo ソースモジュールに直接関連付けられている API はありません。このモジュールは、さまざまな NetX または NetX Duo プロパティを構成するために使用されます。

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.3.8.2 NetX および NetX Duo ソースモジュールの動作の概要

NetX または NetX Duo ソースモジュールの使用方法は、他の SSP モジュールの使用方法とは多少異なります。NetX または NetX Duo ソースモジュールはネットワーク操作を構成するために使用するものであり、API 関数、コールバックまたはその他の一般的なモジュールの機能は提供しません。NetX または NetX Duo モジュールの一般的な動作の概要はありません。NetX の動作の詳細については、Synergy Gallery から入手可能な『NetX User's Manual』を参照してください。

NetX および NetX Duo での TraceX の使用

TraceX が ThreadX ソースコンポーネントで有効になっていて、NetX または NetX Duo ソースコンポーネントの追加時に自動的に組み込まれる場合は、ThreadX および NetX、または NetX Duo ライブラリが含まれるプロジェクトを再ビルドする必要があります。再ビルドしないと、イベントをログに記録する TraceX マクロが実行されません。

4.3.8.3 アプリケーションへの NetX および NetX Duo ソースモジュールの組み込み

e^2 studio コンフィギュレータで生成されたネットワークプロジェクトは、自動的にオブジェクト Add NetX Source を組み込みます。

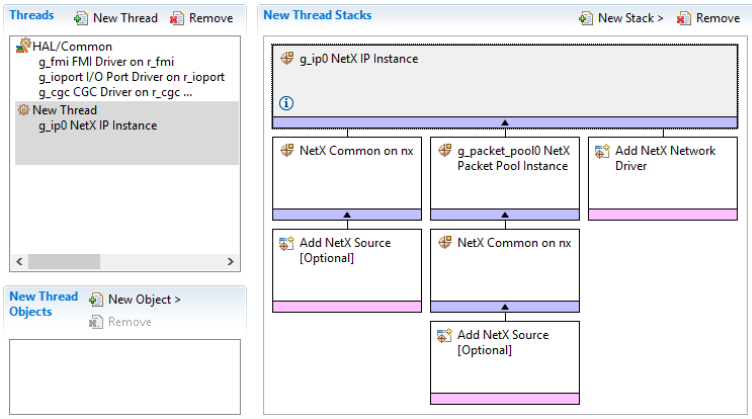


図 324:NetX および NetX Duo ソースモジュールのスタック

NetX または NetX Duo ソースコンポーネントをプロジェクトに追加するには、e2 studio コンフィギュレータで NetX Duo の [Add NetX Source (optional)] または [Add NetX Duo Source (optional)] オブジェクトをクリックし、[New] を選択します。複数の [Add NetX Source] ボックスがある場合は、すべて自動的に更新されます。ThreadX ソースコンポーネントは、自動的に追加されることに留意してください。

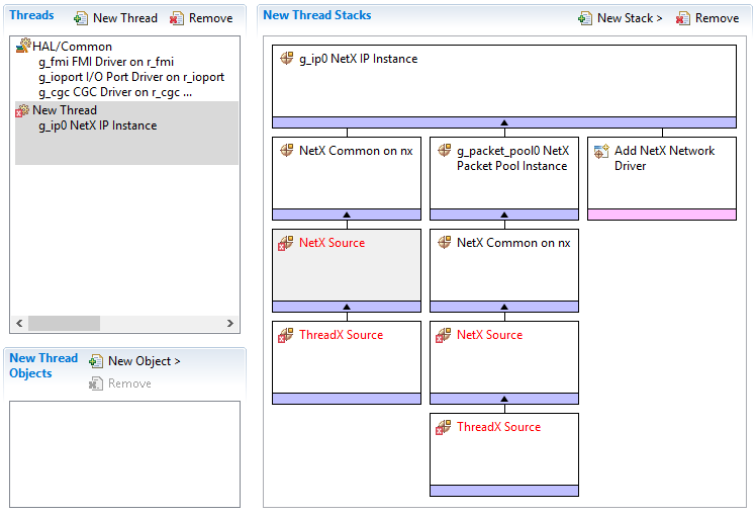


図 325:**ThreadX ソーススタックを使用する NetX および NetX Duo ソース**

4.3.8.4 NetX および NetX Duo ソースモジュールの構成

次のプロパティのリストは、NetX または NetX Duo のプロパティテーブルの順序と一致していない可能性があります。これらは、参照しやすいように ARP、TCP および IGMP という一般的なカテゴリに従ってグループ化されています。

NetX または NetX Duo ソースモジュールの構成可能なプロパティ

このセクションでは、NetX または NetX Duo ソースモジュールで使用できる構成可能なプロパティと、どのような場合にこれらのプロパティをデフォルト値から変更して NetX および NetX Duo の動作をカスタマイズできるかについて説明します。

注 :NetX および NetX Duo プロパティの設定を変更した後、開発者は **[Generate Project Content]** *ボタンをクリックして、ISDE のプロジェクトコンフィギュレータを更新する必要があります。その後、NetX または NetX Duo ライブラリによってプロジェクトを再ビルドする* **必要** *があります。プロジェクトを再ビルドせずプロパティを変更（またはプリプロセッサリストで #define を適用）しても、いずれの変更にも影響しません。Synergy ™ Software Package（SSP）ISDE は、以前ビルドしたライブラリを使用します。*

開発者は、NetX または Net Duo ソースモジュールで使用可能な多数のプロパティの詳細を利用する前に、これらの重要な注意を理解しておく必要があります。

デフォルト設定: ほとんどのデフォルト設定は、プロパティに関連するプロトコルに関する RFC の勧告に基づいています。

[IP Helper Thread Stack Size (bytes)] – デフォルト値は 1024：これは、アプリケーション NetX API コール、周期イベントと延期された ISR イベント、および受信したパケットを処理する IP スレッドタスクのスタックサイズです。デフォルト値よりも大きなスタックサイズが必要になる場合があります。最適なスタックサイズは、経験に基づいてアプリケーションごとに決定されるのが一般的です。

[IP fragmentation] – デフォルト値は有効：フラグメンテーションを有効または無効にするには、[NetX/NetX Duo Source Fragmentation Option] プロパティを使用します。IP インスタンスコンポーネントのこのプロパティは非推奨です。

[IP Helper Thread Priority] – デフォルト値は 3: IP スレッドタスクが動作するプライオリティです。状況によっては、1（最も高い有効なプライオリティ）に上げると最も効率よく動作する可能性があります。たとえば、複数のネットワークアプリケーションスレッドタスク（DNS、HTTP、DHCP）がプロジェクトにある場合、プライオリティが高い（ネットワークアプリケーションスレッドよりも高い）IP ヘルパースレッドは、プライオリティが高いすべてのネットワーク操作に対応できるため、アプリケーションスレッドタスクのレスポンスを効率よく改善することができます。

[Error Checking] – デフォルト値は有効：一般に、開発およびデバッグフェーズでは有効にし、リリースバージョンをビルドするときは無効にします。このプロパティを有効にすると、NetX および NetX Duo によって実際の API をコールする前に入力およびその他のパラメータをチェックするエラーチェックサービスが含められます。チェック対象の例を以下に示します。

- NULL ポインタ入力
- 無効な IP アドレスタイプや 0 に等しい IP アドレスなど無効な非ポインタパラメータ。
- 必須の構成可能なオプションが有効になっていること。たとえば、nx_tcp_socket_receive API を使用するには TCP が有効になっていること。
- NetX または NetX Duo データ構造体 ID が期待される値と一致していること。
  - ip_ptr -> nx_ip_id == NX_IP_ID /* IP インスタンスの構造体を確認
- データ構造体のサイズ、たとえば IP インスタンスのサイズが NetX ライブラリのデータ構造体のサイズと一致していること。

最後の 2 つのチェックは、アプリケーションが使用しているバージョンとは別のバージョンの NetX または NetX Duo ライブラリが使用されていることを検出するために行われます。

[Static Routing] – デフォルト値は無効 特定のルータを介して、特定の宛先にルーティングできるようにします。通常、送信中のパケットは、次のホップとなっている IP インスタンスゲートウェイ / ルータを介してルーティングされます。静的ルーティングが有効な場合は、NetX は静的ルーティングテーブルを確認して、パケットの**ネクストホップ**のアドレスが、IP インスタンスゲートウェイではなく指定したルータを経由するかどうかを判断します。静的ルーティングテーブルは NX_IP_ROUTING_TABLE_SIZE エントリに制限されます。ルーティングテーブルの管理用に、nx_ip_static_route_add などのさまざまな API があります。このプロパティは一般的には使用されませんが、特定の状況で必要となります。

[Physical Header] と [Physical Trailer] – デフォルト値はそれぞれ 16 バイトと 4 バイト：ネットワークメディアのタイプによって、物理（フレームと呼ばれる場合もあります）ヘッダーが決定されます。イーサネットネットワークのヘッダーサイズは 16 バイトです。Wi-Fi のヘッダーは、それよりも長くなります。現在イーサネット用の物理トレーラは使用されていませんが、アプリケーションが使用している物理ネットワークに従って、構成が必要な場合もあります。物理ヘッダーが正しく設定されていない場合は、パケットは組み立てられず、ほとんどの NetX パケットの送受信サービスが正常に機能しません。

[Maximum Listen Requests] – デフォルト値は 10：この値は、TCP 受信待ちキューサイズを定義するために TCP サーバーアプリケーションによって使用されます。TCP サーバーソケットは、ポートをバインドするとき（nx_tcp_server_socket_listen API を使用）に、IP インスタンスで受信待ち要求を要求します。TCP サーバーソケットがそのポートで受信待ちをしないとき（nx_tcp_server_socket_unlisten を使用）は、受信待ち要求は解放され、別のソケットがそのポートで受信を待つことができます。通常、デフォルト値は変更されません。

[Driver Deferred Processing] – デフォルト値は有効：この機能により、IP スレッドタスクは、IP スレッドタスクのコンテキストに対するパケット受信割り込みの処理を延期することができます。有効でない場合は、パケット受信イベントは、割り込みのコンテキストで処理されます。これにより、特定のパケットへのレスポンスは高速になりますが、全体的なシステムの性能は低下する可能性があります。

[Loopback Interface] – デフォルト値は有効：有効な場合、NetX は、それ自体に対してパケットの送受信を行うためのループバックインタフェースを作成します。これは物理インタフェースとしてはカウントされないことに注意してください。アプリケーションは、複数の物理インタフェースを使用している場合、ネットワークインタフェースの数を考慮する必要があります。ループバックインタフェースは、物理ネットワークインタフェースとしてカウントされません（次を参照）。

[Maximum Physical Interfaces] – デフォルト値は 1：デフォルトは 1 つのネットワークインタフェースで、一般的にはプライマリインタフェースと呼ばれます。IP インスタンスがインタフェースのテーブルを保持し、プライマリインタフェースはインデックス 0 の位置にあります。セカンダリインタフェース interface(s) に対しては、[Maximum Physical Interface] の値がセカンダリインタフェース 1 つにつき、1 ずつインクリメントされます。複数のインタフェースを使用する一般的な例としては、ルータがあります。たとえば、一方のインタフェースでローカルネットワークまたはプライベートネットワークに接続し、もう一方のインタフェースでグローバルネットワークまたはパブリックネットワークに接続します。その他の例としては、イーサネットと Wi-Fi のように、2 つの異なるネットワークインタフェースを持つデバイスがあります。

セカンダリインタフェースを IP インスタンスに接続するには、nx_ip_interface_attach API を使用します。インタフェースが API で参照されていない場合は、アクションはプライマリインタフェースで行われます。たとえば、nx_ip_interface_status_checkperforms が特定のインタフェースで動作している間、nx_ip_status_check はプライマリインタフェースで動作します。

注 : 開発者によっては、複数の IP インスタンスを使用して複数の物理ネットワークに対応している場合があります。複数の IP インスタンスを作成しないよう強く推奨します。

[NAT]（NetX Duo でのみ使用可能）– デフォルト値は無効：NAT は、NetX Duo がローカル / プライベートネットワークとパブリック / グローバルネットワークとの間でパケットのマッピングを可能にするプロトコルで、一種のルータとして機能します。このことを行うには、NetX Duo を有効にして、ローカルホストのグローバル IP アドレスと NetX Duo ホストが決定するポートを使用して、ローカルホストからグローバルネットワークにパケットを転送する必要があります。IP インスタンスで NAT が有効な場合は、このようにして NetX Duo がパケットを転送できるようになります。NAT を使用するには、**[Maximum Physical Interfaces]** を 2 に設定しておくことも必要です。SWIOT-5387 には、NAT の設定に関するプロジェクト例があります。

注 :NAT は、NetX Duo でのみ使用できます。

[Fragmentation option] – デフォルト値は有効：パケットデータがデバイスの MTU（最大転送単位）を超えるとフラグメンテーションが発生します。MTU は、イーサネットネットワークでは、ほとんどの場合フレームヘッダーを含めて 1518 バイトとなっています。受信側で対応できることは限られていますが、可能であれば、アプリケーションでは、IP レイヤーでフラグメンテーションを避ける必要があります。

フラグメンテーションで考慮すべき別の問題として、パケットフラグメントの受信に使用するパケットプールを使い切る可能性が挙げられます。NetX が受信するパケットフラグメントが多いと、これらのパケットの受信に使用するパケットプールをすぐに使い切ってしまう可能性があります。この問題は、パケット全体が IP レイヤーによって組み立てられてアプリケーションに転送されるまでは、パケットをパケットプールに戻すことができないために発生します。ボードに十分なメモリがあれば、起動時にパケットプールのパケット数を増やすことで、この問題を解決できます（実行時にはできません）。

注 : このことを、TCP レイヤーで実行されるセグメンテーションと混同しないようにしてください。セグメンテーションの詳細については、TCP MSS Minimum を参照してください。

[Packet Header Pad Size] - デフォルト値は 0（パディングなし）：一部のボードでのハードウェア操作では、メモリを一定のキャッシュラインにアラインする必要があります。そのような場合、NetX または NetX Duo パケットインスタンスのサイズは、ハードウェアの要件に基づいてアラインする必要があります。たとえば、ハードウェアでデータバッファの開始アドレスを 32 バイトにアラインする必要があり、NX_PACKET ヘッダーのサイズが 56 バイトである場合は、[Packet Header Pad Size] を 8 にすることでパケットバッファの開始アドレスを 32 バイトにアラインすることができます。

[Checksums] – デフォルト値は有効：NetX と NetX Duo では、受信（RX）パケットおよび送信（TX）パケットのチェックサムは別々に有効または無効にすることができます。対象となるネットワークプロトコル（UDP、TCP または ICMP）の RFC でゼロチェックサムが許可されていない場合は、チェックサムは無効にしないでください。一般的に、チェックサムは有効にする必要があります。チェックサムを無効にすると、スループットは向上する可能性がありますが、接続の一端でパケットがドロップされる可能性もあります。

[Extended Notify Support] – デフォルト値は無効：有効な場合、NetX は、TCP ソケット接続に関連するさまざまなイベントについて、通常行う以外の通知（nx_tcp_socket_receive_notify および nx_udp_socket_receive_notify API で指定されたコールバックなど）をアプリケーションに対して行います。

[Extended Notify Support] は、NetX および NetX Duo ソケットで必要です。

この機能が有効な API を以下に示します。これらによって、NetX は、NetX による TCP 接続要求の受信時、TCP 接続の完了時、ソケットでの TCP 切断の完了時にアプリケーションに通知し、TCP ソケットの状態を Time-Wait 状態に設定できます。これは、TCP が非ブロッキングコンテキストで動作するために必要です。

```
nx_tcp_socket_syn_received_notify

nx_tcp_establish_notify

nx_tcp_disconnect_complete_notify

nx_tcp_timed_wait_callback
```

[Source Address Check] – デフォルト値は無効：無効な IP アドレスに対する受信パケットをすべてチェックします。具体的には、次のことをチェックします。1）ネットワークマスクによってマスクされた IP アドレスのビットがネットワークマスクの 1 の補数（0xFF）と一致しないこと。2）IP アドレスが 0 ではなく、マスクされていないアド

レスのビットが 0 ではないこと。3）アドレスがタイプ D のアドレス（0xE0000000）ではないこと。NetX において、IP レベルでこの追加の処理が行われると、わずかながら性能の低下が発生します。

[Maximum multicast groups] – デフォルト値は 7：有効なマルチキャストテーブルのサイズを定義し、参加できるマルチキャストグループの数に制限を設定します。一般的には変更しません。

[IGMPV] – デフォルト値は有効：IGMPv2 はデフォルトで有効です。IGMPv1 とは違い、IGMPv2 ではグループ固有の Join クエリ、Leave メッセージ、および（ネットワーク上のすべての IGMP ルータではなく）クエリを転送するルータを決定する方法を使用できます。

[ARP Cache size in Bytes] – デフォルト値は 512：ARP キャッシュのサイズを定義します。このテーブルは、すべての ARP エントリを保持します。テーブルに空きがない場合は、ARP エントリは既存のエントリが「古く」なるまで追加できず（このセクションの [ARP Expiration Rate] の説明を参照）、削除されます。NetX/Duo ソース要素とともに示されている一部の構成可能なオプションは、テーブルに追加される ARP エントリの数に影響する可能性があります。たとえば、このセクションで説明している [ARP Auto ARP Entry] および [ARP Expiration Rate] などです。

ノードがローカルネットワーク上の多数のノードと通信することが予想される場合は、ARP キャッシュサイズの増加が必要な場合があります。

[ARP Auto ARP Entry] – デフォルト値は有効：これにより、NetX は、テーブルに一致する IP アドレスがない ARP パケットを受信した場合に ARP エントリを追加できます。このことは、レスポンスが NetX デバイスに送信された場合でも、NetX デバイスから送信された場合でも可能です。自動 ARP エントリは、データが ARP テーブルにすでに存在する場合に、ARP クエリを送信する必要性を抑えることで NetX の効率性を高めることを目的としています。この機能の短所としては、ARP テーブルに空きがなくなり、新しい ARP クエリを追加できなくなる可能性があることが挙げられます。そのことを防ぐには、この機能を無効にします。無効にすると、要求が NetX によって生成された場合、または NetX デバイスに送信された場合に、NetX は ARP エントリの保存のみを行います。

このオプションは、ほとんどのアプリケーションで必要ありません。

その代わりに、[ARP Auto ARP Entry] を有効のままにして、[ARP Expiration Rate] を 0 以外に設定することで、エントリが使用されない場合に確実に期限切れにする（つまり、エントリを削除する）ことができます。テーブルエントリの期限切れの詳細については、このセクションの **[ARP Expiration Rate]** を参照してください。

[ARP Expiration Rate] – デフォルト値は 0 秒：デフォルトでは、NetX の ARP エントリは **static** です。ARP エントリに期限切れレートは設定されないため、古くなりません。それ以外の場合は、ARP エントリが古くなるに従って、タイムアウト値が減少します。タイムアウト値が 0 になると、ARP エントリは削除されます。ARP エントリに関連する新しいクエリを受信すると、タイムアウト値はリセットされます。IP レイヤーがパケットを送信するためにエントリにアクセスした場合も、タイムアウトはリセットされます。エントリが古くなっている、または削除されていることを判定するために、アプリケーションは nx_arp_info_get API をコールして、古いエントリの統計や、ARP テーブルの管理に関するその他の有益な統計を取得できます。

[ARP Update Rate] – デフォルト値は 10 秒：これは、ARP クエリの再送インターバルです。リトライ回数が [ARP Maximum Retries] に達すると、NetX は IP アドレスと MAC アドレスの物理マッピングの検索を試行しなくなります。この値を小さくしても、悪影響はありません。

[ARP Maximum Queue Depth] – デフォルト値は 4：これは、パケットの MAC アドレスマッピングが NetX にない場合に、そのようなパケットを送信しようとするアプリケーションからの IPv4 パケットの最大数です。NetX が MAC アドレスマッピングの検索を試行している間、これらのパケットはキューに格納されます。キューが一杯の場合は、最も古いパケットが削除されます。この動作が行われているかどうかを確認するために、アプリケーションは e^2 studio の [Expressions] ビューを使用して、IP インスタンスデータ構造体の nx_ip_transmit_resource_errors フィールドをチェックできます。現在、この統計を取得するための API はありません。

この値は、例外的な条件下にある場合を除いて変更する必要はありません。このデフォルト値を使用すると、キューが一杯になることはほとんどありません。宛先 IP アドレスが動作している場合は、NetX が ARP レスポンスを受信し、ARP キューにある対応するパケットが送信されます。宛先 IP が動作していない場合は、そのアドレスに送信するパケットをこれ以上キューに格納する意味はありません。

[ARP Maximum Retries] – デフォルト値は 18：ここには、NetX が IP アドレスマッピングを取得するために、ARP クエリを再送信する回数を指定します。この回数を超えると、再送信を停止します。この値を小さくしても、悪影響はありません。

[ARP Defend by Reply] – デフォルト値は無効：これは、現在の IP アドレスを保持する必要があるホストでの使用を目的としています。

通常、ソース IP アドレスが一致し、MAC アドレスが異なる ARP パケットをホストが取得すると、NetX は ARP 要求をブロードキャストしてその IP アドレスを所有していることを**周知**できます。NetX がこの操作を行えるのは、**[ARP Defend Interval]** プロパティ（このセクションで説明しています）で指定された時間のインターバルの間に、競合する ARP パケットを受信しなかった場合のみです。NetX は、防御返信を送信する場合、タイムアウトをリセットして待機し、次の ARP 競合が発生するとその競合に応答します。そのパケットが、ホストにとって最初の競合する ARP パケットでなく、かつ以前競合した ARP パケットについて記録された時間が **[ARP Defend Interval]** の範囲内である場合は、ホストは即時にこのアドレスの使用を停止する必要があります。この操作は、2 つのホストがともに同じアドレスを防御しようとする無限ループから抜け出せなくなるのを防ぐために必要です。

ARP 競合の詳細については、*「RFC 5227 IPv4 Address Conflicts」の Section 2.4 (b)* を参照してください。

[__ARP Defend by Reply__] が定義されている場合は、**[ARP Defend Interval]** が期限切れになると、以前に送信された ARP 返信以外にも、ARP 返信がブロードキャストされます。この ARP 防御パケットの期限切れまでの待機インターバルには、要件はありません。このプロパティを使用するのは、**[ARP Defend Interval]** タイムアウトが期限切れになると、Windows XP によって、送信済みの ARP 要求パケットが無視されるためです。

[ARP Defend Interval] – デフォルト値は 10 秒：これは、NetX ホストが、競合 ARP パケットを受信した場合に ARP 防御パケットを送信できないインターバルです。10 秒は、RFC 5227 で IPv4 アドレス競合の処理に推奨されているデフォルト値です。NetX は、ARP 防御パケットを送信した後に、タイムアウトをリセットしてこの値に戻します。

[ARP Mac Change Notification] – デフォルト値は無効： NetX が ARP キャッシュテーブルのエントリと一致する MAC アドレスを持つ ARP パケットを受信し、この機能が有効な場合は、NetX は ARP コリジョンハンドラをコールし、アプリケーションがパケットを検査してその処理方法を決定できるようにします。この機能がない場合は、NetX はその ARP テーブルのエントリを新しい MAC アドレスに更新します。この動作は、ARP プロトコルでは通常の動作ですが、**中間者攻撃**、ARP キャッシュポイズニングまたは**スプーフィング**と呼ばれる攻撃によって悪用される可能性があります。この動作により、攻撃者は ARP テーブルの MAC アドレスを改ざんして、パケットをホストから別のホストにリダイレクトできます。NetX には、このような状況に対応する内部のハンドラがあるため、アプリケーションがこの問題に対応する必要がありません。

次のレート設定プロパティについては、NetX における時間設定は、NX_IP_PERIODIC_RATE 設定に基づいています。NX_IP_PERIODIC_RATE は、TX_TIMER_TICKS_PER_SECOND. から直接取得されます。後者のデフォルトは 100 ですが、ユーザーが定義可能で、tx_port.h. に定義することもできます。その場合は、NX_IP_PERIODIC_RATE をユーザー定義の値に設定します。そうでない場合は、NetX はデフォルトで 100 ティック（10 ミリ秒 / ティック）に設定します。

[TCP Fast Timer Rate] – デフォルト値は 10：これにより、NetX で高速タイマが定期的に実行されるインターバルを決定します。これを 10 に設定し、NX_IP_PERIODIC_RATE を 100 に設定すると、高速定期タイマは 100 ミリ秒ごとに実行されます。

このタイマは、ACK パケットとデータパケットをそれぞれ再送信するときのために、遅延された ACK タイムアウト（**[TCP ACK Timer Rate]** を参照）とソケットタイムアウト（**[TCP Transmit Timer Rate]** を参照）をデクリメントするために使用されます。**[TCP Faster Timer Rate]** を増やすと、高速タイマが実行されるインターバルが短くなります。実際には、高速定期タイマを処理する頻度が高くなるほど、余分なオーバーヘッドが発生し性能が低下する可能性があります。

[TCP Retransmit Timer Rate] – デフォルト値は 1（1 秒）：この値は、TCP ソケットタイムアウト値を設定するために使用されます。TCP ソケットは、SYN パケットを送信または受信するときに、またはデータパケットを送信するときに、アクノリッジ（ACK）を待機します。ACK を受信する前にソケットタイムアウトが期限切れになった場合は、**[TCP Maximum Retries]** で指定した最大回数までタイムアウトはリセットされます（パケットは再送信されます）。その後、NetX は接続を閉じます。

一般的には、この値を小さくすることは推奨されません。小さくしても、TCP トランザクションは高速になりません。

[TCP ACK Timer Rate] – デフォルト値は 5：これにより、見つからないデータに対する ACK を NetX が再送信する（つまり ACK プローブの）インターバルが決定されます。NetX の高速定期タイマは、繰り返されるごとにこのタイムアウトをデクリメントします。タイムアウトが期限切れになると、NetX は別の ACK パケットを送信します。データパケットが受信されると、NetX はタイムアウトをリセットします。NetX からデータが送信された場合も、NetX はタイムアウトをリセットします。タイムアウトが期限切れになると、NetX はリトライ回数をインクリメントし、ACK を再送信します。最大リトライ回数に達すると（このセクションで説明している **[TCP Maximum Retries]** を参照）、NetX は接続を閉じます。

5 を設定し、NX_IP_PERIODIC_RATE を 100 に設定すると、200 ミリ秒の遅延されたタイムアウトが発生します。NX_TCP_ACK_TIMER_RATE が大きいほど、ACK タイムアウトは短くなります。この値を増やしても、性能が向上したり、NetX ACK に対するレスポンス時間が短縮されたりすることはありません。短縮されたインターバルの後に、ACK が送信されるだけです。

[TCP Maximum Retries] – デフォルト値は 10：TCP ピアからのレスポンスを受信せずにソケットタイムアウトが期限切れになると、ソケットタイムアウトは **[TCP Maximum Retries]** に等しいリトライ回数になるまでリセットされます。

[TCP Retry Shift] – デフォルト値は 0：これは、再送インターバルに適用されるビットシフトです。デフォルト値 0 は、(TCP ピアからのレスポンスを取得するための) ソケットリトライ間のインターバルを一定に保ちます。これが 1 に設定されている場合は、インターバルは 2 倍になります。つまり、タイムアウト値が 1 ビットシフトされます。ほとんどの場合、この値を変更することはありません。

[TCP Maximum TX Queue] – デフォルト値は 20：これは、NetX が送信および再送信のために TCP ソケットでキューに格納する最大パケット数です。パケットがキューに格納されるのは、ソケットがデータの ACK の受信を待機しているとき、またはデータを送信できる程度に別の端の受信ウィンドウが大きくなるまで待機しているときです。小さなパケットプールや限られたメモリリソースを使用するアプリケーションの場合は、送信キューで待機するパケット数が非常に多くなって他のパケット送信に使用できない場合に、この値を小さくして、パケットプールを使い切らないようにすることができます。

TCP ソケットは、キューに格納できるパケットの最大数に達すると NX_TCP_QUEUE_DEPTH エラーを返します。TCP 受信ウィンドウが小さすぎるためにパケットを送信できない場合には、NX_WINDOW_OVERFLOW エラーを返します。

[TCP Keepalive] – デフォルト値は無効：キープアライブ機能は、確立した状態で、TCP ソケットでタイマを開始してピアに接続します。タイマが期限切れになると、NetX はキープアライブ ACK をピアに送信します。SYN または ACK パケット、NetX デバイスのキープアライブ ACK に応答する ACK パケット、または有効なシーケンス番号の TCP パケットを受信すると、タイマとリトライカウントがリセットされます。

NetX は、キープアライブ ACK の交換を開始するときに、キープアライブ ACK パケットのシーケンス番号を 1 つ減らして、ACK パケットを送信します。このようにして、TCP ピアは 1 バイトのデータが送信されたことを示す ACK とキープアライブ ACK を区別できます。

キープアライブがソケットで有効な場合は、NetX も定期的に別の端がキープアライブ ACK を送信したかどうかをチェックします。

ピアからのレスポンスがない状態でキープアライブタイマが期限切れになると、キープアライブのリトライ回数はインクリメントされます。リトライが [TCP Keepalive Retries] の最大回数に達すると、NetX は接続が切断されているとみなし、接続を終了します。キープアライブを実行しないと、両端ともパケットを送信していない場合に TCP 接続が永久に維持される可能性があります。このような状況では、ソケット接続を閉じる必要があるのか、または開いたままにしておく必要があるのかを判断できません。

[TCP Keepalive Retries] – デフォルト値は 10：キープアライブ ACK へのレスポンスを取得する試行が 10 回行われると、NetX はリセットし、接続を閉じます。

[TCP Keepalive Initial] – デフォルト値は 7200 秒（2 時間）：これは、接続か完了したときに最初のキープアライブが送信されるまで、または以前に送信されたキープアライブパケットからレスポンスを受信するまでのインターバルです。

[TCP Keepalive Retry] – デフォルト値は 75 秒：NetX は、TCP ピアから以前送信されたキープアライブパケットに対するレスポンスを受信していない場合に、別のキープアライブパケットを再送信するまでこのオプションで指定された時間、待機します。

[TCP Window Scaling] – デフォルト値は無効：この機能により、TCP 受信ウィンドウを 65k から、理論上の最大サイズ 1,073,725,440 バイトにまで拡大することができます。NetX TCP クライアントソケットが接続を開始すると、NetX はウィンドウサイズ（TCP ソケットの作成時に設定済み）に基づいて拡大率を計算します。ウィンドウの拡大率の値は、TCP 接続の確立フェーズの間に交換されます。ピアの SYN パケットにウィンドウ拡大オプションがない場合は、ピアがウィンドウの拡大をサポートしていないことを意味します。このような状況では、接続に対してウィンドウの拡大は使用できません。

[TCP Maximum Out of Order Packets] – デフォルト値は無効：有効な場合、このオプションは TCP ソケットの受信キューに格納可能なアウトオブオーダーパケットの最大数を設定します。ソケットの受信キューに最大数のアウトオブオーダーパケットが格納されると、その後に受信するアウトオブオーダーパケットはドロップされます。最終的に、NetX ホストは、データの損失を通知する ACK を送信者に送信し、ドロップされたパケットを再送信してもらう必要があります。この時点で、NetX は受信キューに格納されている残りのすべてのデータをすばやく処理し、パケットを解放できます。

この機能は、以下のシナリオで役立ちます。

パケットの損失やドロップが発生し、送信者がパケットを送信し続けている場合は、TCP ソケットは受信キューに、失われたパケットに続くすべてのパケットを格納する必要があります。TCP ソケットは失われたデータの再送信を待機しているため、どのパケットも解放できません。受信キューの深さに制限がない場合は、パケットプールを使い切り、NetX ホストが実質的に応答しない状態になる可能性があります。この状態は、多くのパケットが TCP ピアによって送信される**バースト**データ送信で発生する可能性があります。パケットをドロップすると、結果として多くのパケットが TCP 受信キューに格納される可能性があります。

[TCP MSS Minimum] – デフォルト値は 128：MSS（最大セグメントサイズ）は、フラグメンテーションを必要としない TCP データの最大サイズです。この**最小** MSS とは、NetX TCP ソケットが TCP ピアから受け入れる最も小さい MSS です。TCP ヘッダーオプションデータから解析された MSS がこの値よりも小さい場合は、接続はドロップされます。
このプロパティは、MSS の値が小さい接続を、アプリケーションで拒否する場合に使用することを目的としています。

[TCP ACK every N Packets] – デフォルトは無効：この機能を有効にするには、正の数を入力します。NetX は、再送信されたデータではなく、新しいデータを含む 1 つおきのパケットに対して ACK を送信します。これは、通常ネットワークトラフィックとパケット処理を最小限にするには最適の設定で、TCP ピアのアドバタイズウィンドウを最新に保ちます。**[TCP Immediate ACK]** が有効な場合、この値は上書きされ、自動的に 1 が設定されることに注意してください。この値を増やすと、別の端が、アドバタイズウィンドウが追加のデータを送信するのに十分なサイズになったことを認識するまで **n 番目の ACK** で待機しなくてはならない可能性が高まります。このことにより、ネットワークスループットが低下する可能性があります。最適な設定は、テストに基づいて決定するのが一般的です。

このパケット N 個あたりの ACK という比率は、TCP の**遅延 ACK** というロジックと関連しています。Telnet などのアプリケーションの場合、N を大きくすると、数多くの 1 バイトパケットを受信する Telnet サーバーで処理効率が大幅に高まる可能性があります。

[TCP Immediate ACK] – デフォルト値は無効：有効な場合、NetX は、受信した新しいデータのすべてのパケットに対して ACK を送信します。このプロパティは、遅延 ACK の使用を希望しない場合に役立ちます。

[Reset Disconnect] – デフォルト値は有効：有効な場合、これにより、アプリケーションは RST パケットを送信しないで、ゼロの待機オプションを使用して TCP ソケットで切断をコールできます。アプリケーションは単に FIN パケットを送信することで、切断を開始してソケットをクローズすることを通知します。アプリケーションは TCP ピアからの ACK または FIN ACK を待機しません。これは、次回のクライアント要求のためにソケットを解放したいサーバーなど、ソケットのクローズを待機したくないホストに対して役立ちます。RST パケットでは問題の発生を通知するのが一般的ですが、そのような通知を行わずにこのことを実現できます。

[RX Size Checking] – デフォルト値は有効：有効な場合、NetX は、受信したパケットに、パケットを処理しているレイヤーに応じて、少なくとも TCP、UDP、IGMP、ICMP といった IP レイヤーヘッダーまたはトランスポートレイヤーヘッダーを格納する十分な領域があるかどうかをチェックします。たとえば、IP レイヤーでは、NetX は、パケットに少なくとも IP ヘッダーを格納するのに十分な領域があるかどうかをチェックします。TCP レイヤーでは、NetX は、パケットに少なくとも TCP ヘッダーを格納するのに十分な領域があるかどうかをチェックします。

パケットが NetX のスタックの上位に移動するに従い、パケット**プリペンド**ポインタとパケット長が各ネットワークレイヤーに合わせて調整されます。パケットは、IP レイヤーから TCP レイヤーに渡されるときに、プリペンドポインタを TCP ヘッダーの先頭に移動し、パケット長のサイズから IP レイヤーのサイズを減算します。同じように、HTTP パケットなどの場合、TCP レイヤーからアプリケーションレイヤーに移動すると、プリペンドポインタは TCP ヘッダーを越えてアプリケーションデータまたはアプリケーションヘッダーに移動されます。パケット長から TCP ヘッダーのサイズが減算されます。

パケットは内部でこのように処理されるので、アプリケーション側でポインタとデータサイズを調整する必要はありません。アプリケーションまたは NetX プロトコルは、通常 nx_tcp_socket_receive API としてソケットコールを作成します。ソケットの受信キューに待機しているキューがある場合は、アプリケーションはパケットを受信し、データの位置と大きさを認識します。アプリケーションは the_nx_packet_length_getAPI（推奨）を使用してパケットデータのサイズを取得するか、NX_PACKET インスタンスの nx_packet_length フィールドに直接アクセスできます。
有効な場合、NetX はその内部動作についての統計を保持します。これらは、IP、TCP、ARP およびパケット（プール）などのコンポーネントレベルの統計です。TCP と UDP については、すべての TCP/UDP 送信の統計とソケットごとの統計があります。これらの統計を無効にすると、処理時間をわずかに短縮できます。

ここに挙げた統計情報の価値は、プログラムのフローを停止または中断したり、冗長なデバッグコードを作成したりすることなく、問題の診断やネットワーク性能の最適化を行うことができるという点にあります。

例 :

nx_packet_pool_info_get.c

アプリケーションがパケットの送信または伝送を行っていない可能性がある場合は、nx_packet_pool_empty_requests 統計をチェックします。この統計は、パケットプールにパケットがないことが原因で nx_packet_allocatefails が失敗するたびにインクリメントされます。これは、nx_packet_allocate が、異なる値を返す可能性がある void 関数または NetX プロトコルによってコールされる場合に役立ちます。

nx_ip_info_get.c

アプリケーションがデータを受信していないときに、サードパーティのパケットトレースでパケットが表示される場合は、nx_ip_total_packets_received 統計をチェックし、データが最低限 IP レベルまで転送されているかどうかを確認します。同様に、パケットトレースで NetX デバイスからのパケットが確認できない場合は、nx_ip_total_packets_sent 統計をチェックします。

以下は、NetX 統計用の API のリストの一部です。

[IP info]:ドライバーからのパケットの受信とトランスポートレイヤーからドライバーへのパケットの転送を行う IP レイヤーのレベルの統計です。

```
UINT  _nx_ip_info_get(NX_IP *ip_ptr, ULONG *ip_total_packets_sent,
    ULONG *ip_total_bytes_sent, ULONG *ip_total_packets_received,
    ULONG *ip_total_bytes_received, ULONG *ip_invalid_packets,
    ULONG *ip_receive_packets_dropped, ULONG *ip_receive_checksum_errors,
    ULONG *ip_send_packets_dropped, ULONG *ip_total_fragments_sent,
    ULONG *ip_total_fragments_received)
```

nx_ip_interface_info_get.c は同じですが、指定されたインタフェースに固有です。

[ARP Info]:送信済み / 受信済み ARP パケットの統計、および ARP テーブルの統計 :

```
UINT  _nx_arp_info_get(NX_IP *ip_ptr, ULONG *arp_requests_sent,
            ULONG *arp_requests_received,
                            ULONG *arp_responses_sent, ULONG *arp_responses_rece
ived,
                            ULONG *arp_dynamic_entries, ULONG *arp_static_entrie
s,
                            ULONG *arp_aged_entries, ULONG *arp_invalid_messages
)
```

[Packet Pool Info]:使用可能なパケット、空のパケット(プール)要求、および無効なパケット解放の統計です。無効なパケットプールまたはパケットポインタが指定されます。

```
INT  _nx_packet_pool_info_get(NX_PACKET_POOL *pool_ptr, ULONG *total_packets,
            ULONG *free_packets,
                            ULONG *empty_pool_requests, ULONG *empty_pool_suspen
sions,
                            ULONG *invalid_packet_releases)
```

[TCP Info]:送信済み / 受信済み TCP パケットの合計数、接続数と切断数、およびドロップ済み / 再送信済みパケット数の統計です。ソケット固有の API には、受信ウィンドウサイズなどがあります。

```
UINT  _nx_tcp_info_get(NX_IP *ip_ptr, ULONG *tcp_packets_sent, ULONG *tcp_bytes_
sent,
                            ULONG *tcp_packets_received, ULONG *tcp_bytes_receiv
ed,
```

```
                        ULONG *tcp_invalid_packets,
            ULONG *tcp_receive_packets_dropped,
                        ULONG *tcp_checksum_errors, ULONG *tcp_connections,
                        ULONG *tcp_disconnections, ULONG *tcp_connections_droppe
d,
                        ULONG *tcp_retransmit_packets)

UINT  _nx_tcp_socket_info_get(NX_TCP_SOCKET *socket_ptr, ULONG *tcp_packets_sent
,
            ULONG *tcp_bytes_sent,
                        ULONG *tcp_packets_received, ULONG *tcp_bytes_received,
                        ULONG *tcp_retransmit_packets, ULONG *tcp_packets_queued
,
                      ULONG *tcp_checksum_errors, ULONG *tcp_socket_state,
                      ULONG *tcp_transmit_queue_depth,
            ULONG *tcp_transmit_window,
                      ULONG *tcp_receive_window)
```

[UDP Info]：送信済み / 受信済み UDP パケット、ドロップ済みパケット、および送信または受信される不適切な形式のパケットの合計数に関する統計です。ソケット固有の API には、受信済みの無効なパケットのカウントは含まれません。

```
UINT  _nx_udp_info_get (NX_IP *ip_ptr, ULONG *udp_packets_sent, ULONG *udp_bytes
_sent,
                        ULONG *udp_packets_received, ULONG *udp_bytes_received,
                        ULONG *udp_invalid_packets,
            ULONG *udp_receive_packets_dropped,
                        ULONG *udp_checksum_errors)

UINT  _nx_udp_socket_info_get (NX_UDP_SOCKET *socket_ptr, ULONG *udp_packets_sen
t,
            ULONG *udp_bytes_sent, ULONG *udp_packets_received,
            ULONG *udp_bytes_received, ULONG *udp_packets_queued,
                        ULONG *udp_receive_packets_dropped,
            ULONG *udp_checksum_errors)
```

[ICMP Info]：未サポートの ICMP メッセージ、タイムアウトした ping 要求、および ping 要求へのレスポンスなど、制御メッセージの送信に関する統計です。

```
UINT  _nx_icmp_info_get(NX_IP *ip_ptr, ULONG *pings_sent, ULONG *ping_timeouts,
                        ULONG *ping_threads_suspended,
            ULONG *ping_responses_received,
                        ULONG *icmp_checksum_errors,
            ULONG *icmp_unhandled_messages)
```

[IGMP Info]：参加済みマルチキャストグループ、受信済み IGMP クエリ、および送信済み IGMP レポートの統計です。

```
UINT  _nx_igmp_info_get(NX_IP *ip_ptr, ULONG *igmp_reports_sent,
                ULONG *igmp_queries_received,
                         ULONG *igmp_checksum_errors, ULONG *current_groups_join
```

```
ed)
```

4.3.9 Express Logic 社の NetX の概要

Express Logic 社の NetX は、TCP/IP 標準をリアルタイムに実装できるパフォーマンスに優れたツールです。組み込み ThreadX ベースアプリケーション専用に設計されています。Renesas SSP には、Express Logic 社の NetX のネットワークスタック（nx）が組み込まれています。Express Logic 社の NetX インタフェースを使用すると、SSP プロジェクトで NetX の機能にアクセスする手段を利用できます。

NetX（API リファレンスを含む）の詳細については、Web サイトで提供されている X-Ware コンポーネントドキュメントパックの一部として入手可能な『NetX ユーザーガイド』を参照してください。
https://www.renesas.com/en-us/products/synergy/software/ssp.html.

4.3.9.1 Express Logic 社の NetX と TraceX の使用

ThreadX ソースコンポーネントで TraceX を有効にしていると、NetX ソースコンポーネントを追加するときに TraceX が自動的に組み込まれます。TraceX を有効にしたときは、ThreadX および NetX ライブラリを含むプロジェクトを再ビルドする必要があります。そうしないと、イベントをログに記録する TraceX マクロが実行されません。

4.3.9.2 Express Logic 社の NetX プロトコルモジュール

Express Logic 社の SSP 用 NetX インタフェースでは、定評のある幅広いネットワークプロトコル関数がサポートされています。このドキュメントでは、使用可能なプロトコル関数に対応するモジュール概要セクションを掲載しています。これらの概要では、モジュールがアプリケーションに必要な関数を提供しているかどうかを判断するための十分な情報を提供しています。

完全なアプリケーションプロジェクト（AP）も使用可能で、Renesas Synergy ハードウェアキットを対象とした典型的な NetX プロトコルアプリケーション向けに実際に動作するプロジェクトを確認できます。AP は Web サイトから入手できます。
https://www.renesas.com/en-us/products/synergy/gallery/solutions-category.html?applicationType=appproject&heading=Application%20Projects

4.3.9.3 Express Logic 社の NetX の構成に関する注意

プロトコルモジュールがプロジェクトに追加されると、アプリケーションコードのビルド済みライブラリが追加されます。各プロトコルコンポーネントには、保護ソースファイルを含む、末尾が「'_src'」の類似のコンポーネントがあります。ビルド済みのライブラリモジュールのほかに、「'_src'」コンポーネントを追加することが可能です。ビルド済みライブラリモジュールなしで、「'_src'」コンポーネントを追加しないでください。

NetX ソースモジュールをプロジェクトに追加した場合、[Properties] ウィンドウでは、NetX ソースライブラリの高度な構成を行うことができます。構成オプションをハイライトすると、そのオプションの説明が e^2 studio GUI の左下隅に表示されます。構成オプションが空の場合は、デフォルト値が使用されます。構成オプションのデフォルト値は、ssp/inc/framework/el/nx/nx_port.h. に定義されます。詳細については、『*NetX ユーザーガイド*』の「構成オプション」の章を参照してください。

詳細な NetX および NetX Duo ソースモジュールの動作の概要は、このドキュメントに記載されています。この概要では、ソースモジュール内で使用できる構成可能なネットワーク操作について説明しています。NetX または NetX Duo のソース構成設定の変更を試行する前に読んで、理解しておくことを強くお勧めします。

4.3.9.4 Express Logic 社の NetX の制限

- Express Logic 社の NetX は、Express Logic 社の NetX Duo と同じアプリケーションでは使用できません。アプリケーションごとにいずれか一方のみを使用できます。
- NetX BSD を GCC コンパイラで使用する場合、コンパイルエラーを避けるためにマクロ _POSIX_SOURCE を定義してビルドします。

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.9.5 Express Logic 社の NetX がサポートするデバイス

- サポートされている MCU デバイスの完全なリストについては、関連する SSP のリリースノートを参照してください。

4.3.10 Express Logic 社の NetX Duo の概要

Express Logic 社の NetX Duo は、TCP/IP 標準をリアルタイムに実装できるパフォーマンスに優れたツールです。組み込み ThreadX ベースアプリケーション専用に設計されています。Renesas SSP には、Express Logic 社の NetX Duo のネットワークスタック（nxd）が組み込まれています。Express Logic 社の NetX Duo インタフェースを使用すると、SSP プロジェクトで NetX Duo の機能にアクセスする手段を利用できます。

NetX Duo（API リファレンスを含む）の詳細については、次で提供されている X-Ware コンポーネントドキュメントパックの一部として入手可能な『*NetX Duo ユーザーガイド*』を参照してください。
https://www.renesas.com/en-us/products/synergy/software/ssp.html .

4.3.10.1 Express Logic 社の NetX Duo と TraceX の使用

TraceX が ThreadX ソースコンポーネントで有効になっていて、NetX Duo ソースコンポーネントの追加時に自動的に組み込まれる場合は、ThreadX および NetX Duo ライブラリが含まれるプロジェクトを再ビルドする必要があります。再ビルドしないと、イベントをログに記録する TraceX マクロが実行されません。

4.3.10.2 Express Logic 社の NetX Duo プロトコルモジュール

Express Logic 社の SSP 用 NetX Duo インタフェースでは、定評のある幅広いネットワークプロトコル関数がサポートされています。このドキュメントでは、使用可能なプロトコル関数に対応するモジュール概要セクションを掲載しています。これらの概要では、モジュールがアプリケーションに必要な関数を提供しているかどうかを判断するための十分な情報を提供しています。

完全なアプリケーションプロジェクト（AP）も使用可能で、Renesas Synergy ハードウェアキットを対象とした典型的な NetX プロトコルアプリケーション向けに実際に動作するプロジェクトを確認できます。AP は Web サイトから入手できます。
https://www.renesas.com/en-us/products/synergy/gallery/solutions-category.html?applicationType=appproject&heading=Application%20Projects

4.3.10.3 Express Logic 社の NetX Duo の制限

- Express Logic 社の NetX Duo は、Express Logic 社の NetX と同じアプリケーションでは使用できません。アプリケーションごとにいずれか一方のみを使用できます。
- NetX Duo BSD を GCC コンパイラで使用する場合、コンパイルエラーを避けるためにマクロ _POSIX_SOURCE を定義してビルドします。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.10.4 Express Logic 社の NetX Duo がサポートするデバイス

- サポートされている MCU デバイスの完全なリストについては、関連する SSP のリリースノートを参照してください。

4.3.11 NetX/NetX Duo Auto IP

Auto IP プロトコルは、動的ホスト構成プロトコル（DHCP）とは違い、サーバーを必要とせずにローカルネットワーク上で IPv4 アドレスを動的に構成するよう設計されています。Auto IP はアドレス解決プロトコル（ARP）を使用して自動的に IP アドレスを割り当て、169.254.1.0 から 169.254.254.255 までの範囲でアドレスを割り当てます。

注 :NetX Duo ™ Auto IP モジュールは、内部処理を除けば、Auto IP セッションの応用、設定、実行が NetX ™ Auto IP モジュールと同じです。

4.3.11.1 NetX/NetX Duo Auto IP モジュールの特長

- RFC3927 および関連 RFC に準拠しています。
- アドレス競合があるかどうかの確認に ARP プローブを使用します。
- NetX のコリジョンハンドラ通知を使用して、使用中のアドレスを検出します。
- IP インスタンスに有効な Auto IP アドレスを登録します。
- また、以下を行うためのハイレベル API を提供します。
  - Auto IP インスタンスの作成と削除
  - Auto IP スレッドタスクの開始と停止
  - Auto IP を実行するネットワークインタフェースの指定

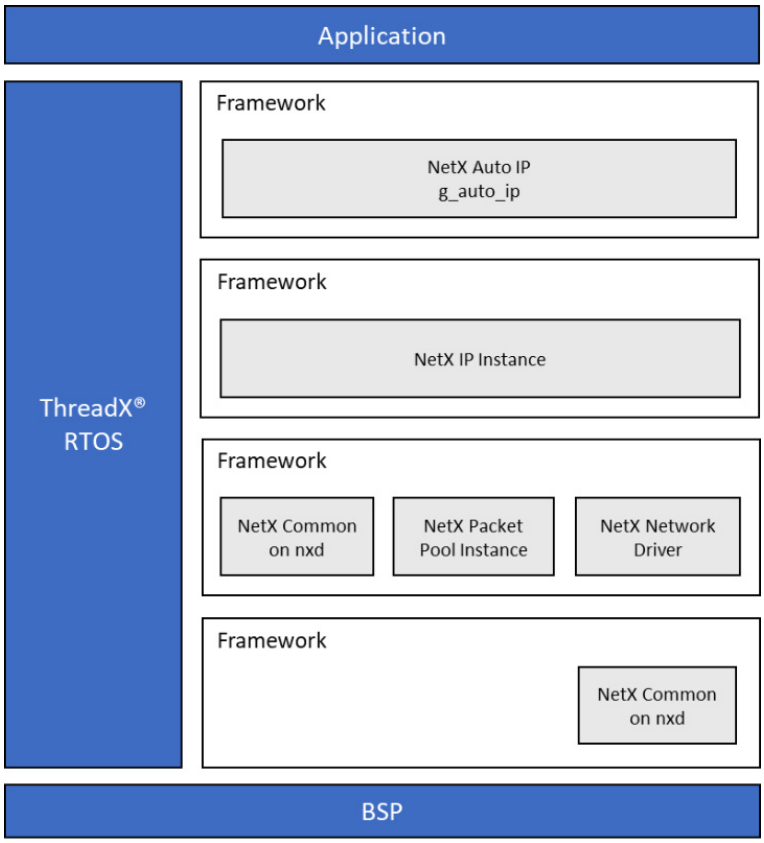


図 326:**NetX/NetX Duo Auto IP モジュールのブロック図**

注 : 上の図で、NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.11.2 NetX/NetX Duo Auto IP モジュールの API の概要

NetX Auto IP では、アドレスの作成、削除、取得および設定のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX/NetX Duo Auto IP モジュールの API の要約

Function Name	Example API Call and Description
nx_auto_ip_create	nx_auto_ip_create(&g_auto_ip0, "AutoIP 0", &g_ip0, stack_pointer, stack_size, priority); Create an Auto IP instance
nx_auto_ip_delete	nx_auto_ip_delete(&g_auto_ip_0); Delete Auto IP instance.
nx_auto_ip_get_address	nx_auto_ip_get_address(&g_auto_ip_0, &local_address); Get current Auto IP address.
nx_auto_ip_set_interface	nx_auto_ip_set_interface(&g_auto_ip_0, interface_index); Set network interface needing an Auto IP address.
nx_auto_ip_start	nx_auto_ip_start(&g_auto_ip_0, IP_ADDRESS(0,0,0,0)); Start Auto IP processing. If the address input is NULL. NetX Auto IP randomly assigns an address in the Auto IP address range.

Function Name	Example API Call and Description
nx_auto_ip_stop	nx_auto_ip_stop(&g_auto_ip_0); Stop Auto IP processing.
nx_dhcp_server_stop	nx_dhcp_server_stop(&dhcp_server); Stop DHCP server processing.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

ステータス戻り値

Name	Description
NX_SUCCESS	Successful AutoIP function
NX_AUTO_IP_ERROR	Error creating components of Auto IP instance
NX_PTR_ERROR*	Invalid pointer input
NX_CALLER_ERROR*	Invalid caller of this service
NX_AUTO_IP_NO_LOCAL	No Auto IP address registered with the NetX IP instance.
NX_AUTO_IP_BAD_INTERFACE_INDEX	Invalid network interface

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。注 : これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX と NetX Duo でのエラーチェックサービスの詳細については、それぞれ『NetX User Guide for the Renesas Synergy ™ Platform』または『NetX Duo User's Guide for the Renesas Synergy ™ Platform』を参照してください。*

4.3.11.3 NetX/NetX Duo Auto IP モジュールの動作の概要

NetX Auto IP プロトコルは、最初に 169.254.1.0 から 169.254.254.255 までの Auto IP IPv4 アドレスの範囲でランダムなアドレスを選択します。または、アプリケーションで開始 IP アドレスを nx_auto_ip_start サービスに指定して、そのアドレスを強制的に使用することができます。この方法は、以前に Auto IP アドレスが使用されていた場合に役立ちます。

Auto IP アドレスが選択されると、NetX Auto IP は、選択したアドレスを探すための一連の ARP プローブを送信します。ARP プローブは、送信元アドレスを 0.0.0.0 に、宛先アドレスを目的の Auto IP アドレスに設定した ARP 要求

メッセージで構成されます。このような一連の ARP プローブが送信されます（実際の数は、NetX Auto IP インスタンスのプロパティを送信するために ARP プローブによって設定されます）。別のネットワークノードがこのプローブに応答するか、同じアドレスに対する同一のプローブを送信すると、Auto IP IPv4 アドレスの範囲内で新しい Auto IP アドレスがランダムに選択され、プローブ処理が繰り返されます。

ARP プローブが送信されてレスポンスがない場合は、NetX Auto IP は選択されたアドレスを対象とする多数の ARP 通知（数は [Number of ARP announces] プロパティで設定）を発行します。ARP 通知は、選択した Auto IP アドレスに設定された ARP メッセージに記載されている送信元アドレスと宛先アドレスの両方を備えた ARP 要求メッセージで構成されます。別のネットワークノードが通知されたメッセージに応答するか、同じアドレスに対する同一の通知を送信した場合には、Auto IP IPv4 アドレスの範囲内で新しい Auto IP アドレスがランダムに選択され、プローブ処理がやり直されます。競合が検出されずにプローブと通知が完了すると、選択された Auto IP アドレスは有効とみなされ、そのアドレスは IP インスタンスに登録されます。

NetX Auto IP は Auto IP で生成された IP アドレスを NetX IP インスタンスの成功したプローブおよび通知の処理に登録します。NetX の nx_ip_address_change_notify コールバックを使用して Auto IP アプリケーションにアドレスの変更を通知できます。または、Auto IP アプリケーションは nx_ip_status_check を使用して、有効な IP アドレスが割り当てられたタイミングを確認できます。有効なアドレスが割り当てられたら、アプリケーションは nx_auto_ip_stop サービスを使用して、Auto IP タスクを停止する必要があります。アドレス変更コールバックは、Auto IP スレッドタスクがサスペンドされるとアプリケーションにアドレスの変更を通知します。Auto IP を使って明示的なアドレスの変更をしていないのにアドレスが変わる理由としては、他のノードとの間における Auto IP アドレスの競合と、DHCP アドレス解決による Auto IP アドレスの置換が考えられます。

NetX/NetX Duo Auto IP モジュールの動作に関する重要な注意事項と制限事項

- NetX DHCP クライアントと NetX Auto IP は、ともに有効な IP アドレスをホストに設定するために使用できます。通常、DHCP クライアントはサーバーとの通信を試行します。DHCP クライアントに応答するサーバーがない場合、クライアントはサスペンド状態となり、Auto IP タスクが開始されます。一般的には、Auto IP は、使用可能な DHCP サーバーがない場合にローカルアドレスを割り当てます。DHCP クライアントは、後から DHCP サーバーに要求をブロードキャストできます。このプロセスが成功すると、Auto IP のローカルアドレスが自動的に上書きされます。
- IP アドレスが変更された場合は、アプリケーションは既存のソケット接続をクローズする必要があります。
- NetX DHCP を Auto IP とともに使用する場合は、作成された DHCP スレッドのプライオリティは、Auto IP スレッドよりも高くする必要があります。
- NetX Auto IP には、以前使用されていた IP アドレスを保持するメカニズムはありません。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.11.4 アプリケーションへの NetX/NetX Duo Auto IP モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX/NetX Duo Auto IP モジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo Auto IP モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo Auto IP モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_auto_ip0 NetX Auto IP	Threads	New Stack> X-Ware> NetX> Protocols> NetX Auto IP
g_auto_ip0 NetX Duo Auto IP	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo Auto IP

次の図に示すように、NetX/NetX Duo Auto IP モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

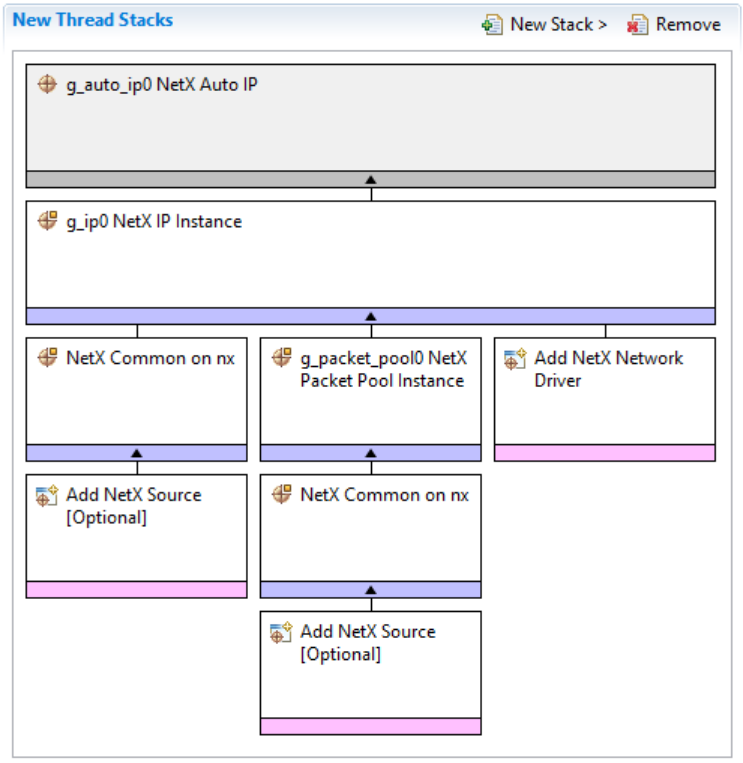


図 327:NetX/NetX Duo Auto IP モジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.11.5 NetX/NetX Duo Auto IP モジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo Auto IP モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo Auto IP モジュールの構成設定

ISDE Property	Value	Description
Wait before sending first probe (seconds)	1	Wait before sending first probe selection
ARP probes to send	3	ARP probes to send selection
Minimum wait between probes (seconds)	1	Minimum wait between probes selection
Maximum wait between probes (seconds)	2	Maximum wait between probes selection
Maximum conflicts before increasing processing delay	10	Maximum conflicts before increasing processing delay selection
Wait extend after maximum conflicts (seconds)	60	Wait extend after maximum conflicts selection

ISDE Property	Value	Description
Wait before announcement (seconds)	2	Wait before announcement selection
Number of ARP announces	2	Number of ARP announces selection
Wait between announces (seconds)	2	Wait between announces selection
Wait between defense announces (seconds)	10	Wait between defense announces selection
Name	g_auto_ip0	Module name
Internal thread stack size (bytes)	2048	Internal thread stack size selection
Internal thread priority	3	Internal thread priority selection
Name of generated initialization function	auto_ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

スタックモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、イーサネットポートに対して異なるアドレスを選択するときに役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo Auto IP のローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	0,0,0,0	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection

ISDE Property	Value	Description
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

*注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo Auto IP モジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo Auto IP モジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I^2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.11.6 アプリケーションでの NetX/NetX Duo Auto IP モジュールの使用

一般的なアプリケーションでは、IP インスタンスが作成済みで、ARP が有効となっていることを前提としています。このIPインスタンスを作成した後、アプリケーションでNetX Auto IPを使用する際の一般的な手順は次のとおりです。

1) tx_thread_sleep API によって IP スレッドタスクとネットワークドライバーが初期化される時間（2 ～ 3 秒）を見込んでおきます。
2) nx_ip_address_change_notify API を使用して、アドレス変更通知を設定します [オプション]。
3) nx_auto_ip_start API を使用して、AutoIP インスタンスを開始します。

4) nx_ip_status_check API または nx_auto_ip_get_address API のいずれかを使用して、IP インスタンスの有効なアドレスをチェックします。nx_ip_status_check API のデフォルト値はプライマリアドレスです。セカンダリインタフェースで Auto IP を実行する場合は、nx_ip_interface_status_check. を使用します。nx_auto_ip_get_address API は、プライマリアドレスまたはセカンダリアドレスのいずれかの Auto IP に利用できます。
5) 有効なローカル IP アドレスが割り当てられたら、nx_auto_ip_stop API を使用して、Auto IP スレッドタスクを停止します。

次の図は、一般的な手順を示した通常の動作フロー図です。

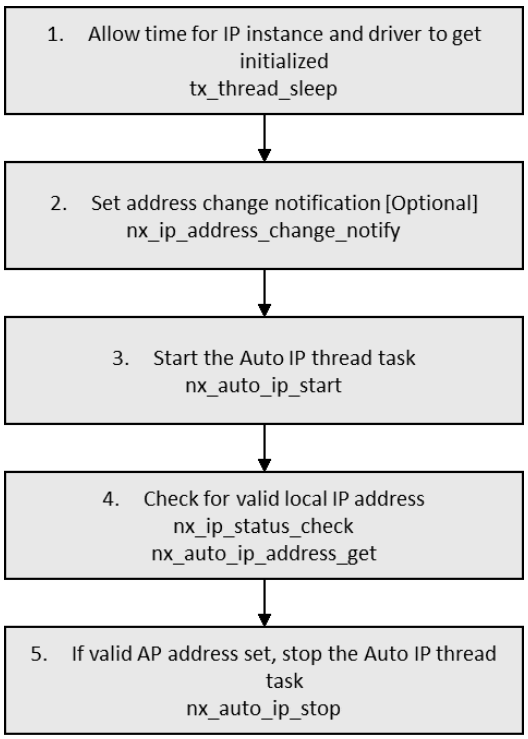


図 328:一般的な NetX/NetX Duo Auto IP モジュールアプリケーションのフロー図

4.3.12 NetX/NetX Duo BSD サポート

BSD ソケット API 準拠ラッパー（NetX ™ BSD）は、NetX ™サービスを使用して、基本 BSD ソケット API コールのサブセット（いくつかの制限があります）をサポートします。

注 :NetX Duo ™ BSD サポートモジュールは、内部処理を除けば、BSD サポートセッションの応用、設定、実行が NetX ™ BSD サポートモジュールと同じです。

4.3.12.1 NetX/NetX Duo BSD サポートモジュールの特長

- NetX BSD サポートモジュールは BSD 4.3 に準拠しています。

- 以下を行うためのハイレベル API を提供します。
  - ソケットの作成と削除
  - ソケットオプションの設定
  - TCP 接続の要求と接続要求の受信待ち
  - パケットの送信と受信
  - 未加工パケットのサポート **
  - PPP over Ethernet のサポート **

  **NetX Duo BSD のみ

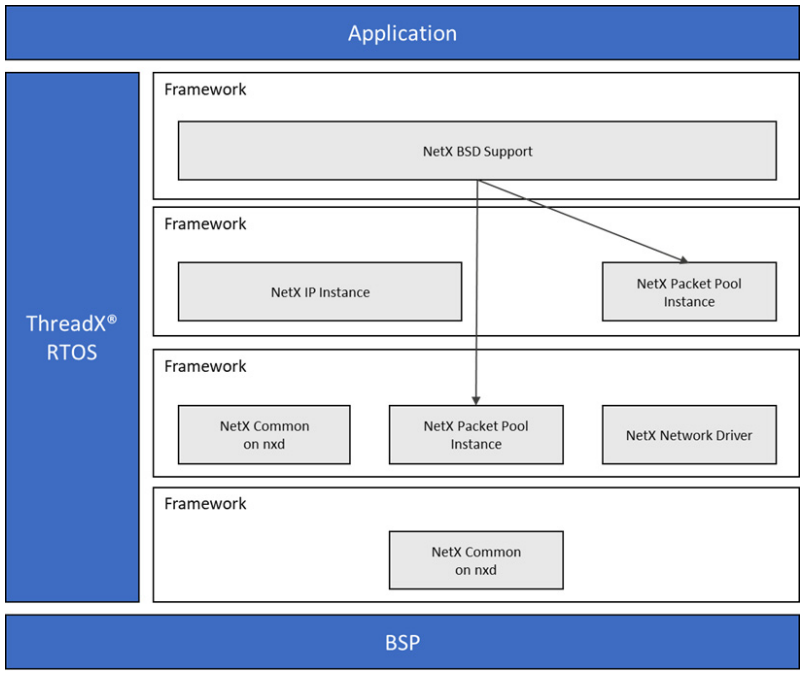


図 329:NetX/NetX Duo BSD サポートモジュールのブロック図

注 : 上の図で、NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.12.2 NetX/NetX Duo BSD サポートモジュールの API の概要

NetX BSD サポートモジュールでは、接続、バインド、受信待ち、送信、受信のための標準 BSD API 関数が提供されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。NetX および NetX Duo BSD によって実装されるサービスは nx_, で始まり、この表の最後に示されています。

NetX/NetX Duo BSD サポートモジュールの API の要約

Function Name	Example API Call and Description
accept	accept(INT sockID, struct sockaddr *ClientAddress, INT *addressLength); TCP server socket waits to make a TCP connection
bsd_initialize	bsd_initialize(NX_IP *default_ip, NX_PACKET_POOL *default_pool, CHAR *bsd_thread_stack_area, ULONG bsd_thread_stack_size, UINT bsd_thread_priority); Sets up BSD Support Module to use NetX and BSD services. (Called by the NetX BSD Support Module automatically.)
bind	bind(INT sockID, struct sockaddr *localAddress, INT addressLength); Bind TCP or UDP socket to a local port
connect	connect(INT sockID, struct sockaddr *remoteAddress, INT addressLength); Connect to a TCP peer; if the remoteAddress indicates raw or UDP socket, then if the address is NULL this dis-associates the peer from this socket.
fcntl	fcntl(INT sock_ID, UINT flag_type, UINT f_options); Sets socket options for the specified socket
freeaddrinfo	freeaddrinfo(struct addrinfo *res); Releases memory allocated by the getaddrinfo service

Function Name	Example API Call and Description
getnameinfo	getnameinfo(const struct sockaddr *sa, socklen_t salen, char *host, size_t hostlen, char *serv, size_t servlen, int flags); Converts a socket address to a corresponding host and service string.
getpeername	getpeername(INT sockID, struct sockaddr *remoteAddress, INT *addressLength); Return remote peer's IP address and port.
getsockname	getsockname(INT sockID, struct sockaddr *localAddress, INT *addressLength); Return the socket's primary IP address and source port. In NetX Duo, this would be the address as index 1 in the IP interface table.
getsockopt	getsockopt(INT sockID, INT option_level, INT option_name, VOID *option_value, INT *option_length); Return the status of the specified socket option
getaddrinfo	getaddrinfo(const CHAR *node, const CHAR *service, const struct addrinfo *hints, struct addrinfo **res); Fills in the addrinfo for the indicated node (host) based on hints in the addrinfo input.
ioctl	ioctl(INT sockID, INT command, INT *result); Carry out the command on the specified socket. Only two options supported FIONREAD (extract number of bytes on socket queue) and FIONBIO (enable/disable non blocking as per the result flag).

Function Name	Example API Call and Description
inet_addr	inet_addr(const CHAR *buffer); Convert an IP address from a string buffer to a number.
inet_ntoa	inet_ntoa(struct in_addr address_to_convert); Convert an IP address to a string.
inet_aton	inet_aton(const CHAR * address_buffer_ptr, struct in_addr *addr); Converts hexadecimal characters into an ASCII IP address representation
inet_pton	inet_pton(INT af, const CHAR *src, VOID *dst); Converts an IP address from string to numeric format
inet_ntop	inet_ntop(INT af, const VOID *src, CHAR *dst, socklen_t size); Converts an IP address from numeric format to string presentation.
listen	listen(INT sockID, INT backlog); Bind a TCP server socket to a local port on which to listen for connection requests.
recvfrom	recvfrom(INT sockID, CHAR *buffer, INT buffersize, INT flags,struct sockaddr *fromAddr, INT *fromAddrLen); Receive up to the specified number of bytes on the specified socket (either UDP or TCP)

Function Name	Example API Call and Description
recv	recv(INT sockID, VOID *rcvBuffer, INT bufferLength, INT flags) Copies up to a specified number of bytes received on the socket into specified location. The given socket can be UDP or TCP, but must be in the connected state.
send	send(INT sockID, const CHAR *msg, INT msgLength, INT flags) Send the specified buffer out on the socket; the actual number of bytes sent is returned in msglength. Does not support raw sockets.
sendto	sendto(INT sockID, CHAR *msg, INT msgLength, INT flags, struct sockaddr *destAddr, INT destAddrLen); Send the specified buffer out on the socket; the actual number of bytes sent is returned in msglength. The socket must be in a connected state. Supports raw sockets (NetX Duo BSD only)
select	select(INT nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout); Check all sockets specified in the fd_set inputs to be checked for read request (incoming packets), write request (outgoing packets), and exception request input.
soc_close	soc_close(INT sockID); Close the specified socket.
socket	socket(INT protocolFamily, INT type, INT protocol); Creates and endpoint for communication and returns a file descriptor for the socket.

Function Name	Example API Call and Description
setsockopt	setsockopt(INT sockID, INT option_level, INT option_name, const VOID *option_value, INT option_length); Enable the input socket option on the socket.
fcntl	fcntl(INT sock_ID, UINT flag_type, UINT f_options); Sets flag options for the specified socket.
nx_bsd_raw_packet_info_extract**	nx_bsd_raw_packet_info_extract(NX_PACKET *packet_ptr, NXD_ADDRESS *nxd_address, UINT *interface_index); Extracts source IP address and interface index of the IP address in the IP interface table
nx_bsd_socket_set_inherited_settings	nx_bsd_socket_set_inherited_settings(UINT master_sock_id, UINT secondary_sock_id); Apply the socket options of the specified master socket to the specified child secondary socket; requires NX_BSD_INHERIT_LISTENER_SOCKET_SETTINGSbe defined. If it is not, this function has no effect.
nx_bsd_set_service_list	nx_bsd_set_service_list(struct NX_BSD_SERVICE_LIST *serv_list_ptr, ULONG serv_list_len); Define the service list used bygetaddrinfowith the specified service list.

*注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。注 :** この API は、NetX Duo FTP クライアントでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

4.3.12.3 NetX/NetX Duo BSD サポートモジュールの動作の概要

NetX および BSD サービスを利用するために、NetX BSD サポートモジュールは自動的に IP インスタンスを作成し、内部 BSD スレッドスタックのメモリ空間を作成します。パケットプールには、IP のデフォルトパケットプール（g_packet_pool0）を使用できます。または、[Add NetX Packet Pool]（あるいは [Add NetX Duo Packet Pool]）-> [New] の順にクリックして、BSD パケットの送信に別のパケットプール g_packet_pool1, を使用します。メモリ空間用の定義パラメータは、NetX BSD サポートスタック要素の internal_thread_stack_sizeand および internal_thread_priorityproperties プロパティです。

NetX BSD サービスを使用する前に、アプリケーションは、ローカルとピアがホストする 1 つ以上の sockaddr_in インスタンスを作成する必要があります。サーバーアプリケーションは、それ自体のためだけに sock_addr を作成する必要があります（これらは IPv4 アドレシングのみに制限されます）。IPv6 アドレス（NetX Duo BSD が必要です）の場合は、アプリケーションは 1 つ以上の sockaddr_in6instances. インスタンスを作成します。ソケットサービスを使用して、タイプが TCP または UDP のソケットが作成されます。プロトコルは、IPv4 では AF_INET、IPv6 では AF_INET6 にする必要があります。

未加工パケット（NetX Duo BSD のみ）の場合は、ソケットは AF_PACKET. のタイプにする必要があります。

オプションで、アプリケーションは fnctl および ioctl サービスを使用して、非ブロッキングなどのソケットオプションを設定できます。

TCP および UDP クライアントの場合は、ソケットは bind サービスを使用して、ローカルソースポートをバインドする必要があります。sockaddr_in データインスタンスのポートの値が 0 の場合は、NetX にローカルポートを割り当てるよう通知します。TCP ソケットの場合は、クライアントソケットは connect サービスを使用して TCP サーバーに接続します。

UDP および TCP ソケットは、ともにデータを送受信できます。BSD はストリーミングベースのプロトコルであるため、ユーザー定義のバッファでアプリケーションに対してデータの送受信が行われます。内部的には、NetX はパケットを使用して、パケットプールからアプリケーションにデータの送信と伝送を透過的に行います。受信したパケットまたは NetX が送信できなかった割り当て済みのパケットを解放する必要はありません。

クライアントソケットを閉じるために、アプリケーションは soc_close. をコールします。soc_close. を使用してソケットをクローズするときに、ソケットをアンバインドする必要はありません。

TCP サーバーの場合は、アプリケーションが listen サービスを使用して、指定したマスタソケットが受信待ちを行うローカルポートを選択します。マスタソケットは、select サービスを使用して、接続要求をチェックします。要求が検出されたら、マスタソケットは accept サービスを呼び出し、2 番目のソケットを割り当てて接続を処理します。このようにして、BSD は複数の接続を同時に維持できます。

サーバーソケットを閉じるために、アプリケーションは soc_close. をコールします。NetX TCP ソケットとは違い、NetX BSD では TCP ソケット上で nx_tcp_server_socket_unlisten API または nx_tcp_server_socket_unaccept API をコールする必要はありません。

内部 BSD スレッドの排除

デフォルトでは、BSD は内部スレッドを利用して、一部の処理を実行します。メモリの制約が厳しいシステムでは、NX_BSD_TIMEOUT_PROCESS_IN_TIMER を定義して、NetX BSD をビルドできます。これにより、内部 BSD スレッドは不要となり、代わりに内部タイマを使用して同じ処理を実行します。このようにして、内部の BSD スレッド制御ブロックおよびスタックに必要なメモリを不要にします。タイマ処理が大幅に増大し、BSD 処理が必要以上に高いプライオリティで実行される可能性もあります。

BSD ソケットを ThreadX タイマコンテキストで実行するよう構成するには、プロジェクトに NX_BSD_TIMEOUT_PROCESS_IN_TIMER を定義します。BSD タスクをタイマコンテキストで実行するよう BSD レイヤーが構成されている場合は、BSD スタック要素の以下のプロパティが無視されます。

- internal_thread_stack_size
- internal_thread_priority

DNS サポートを使用した NetX BSD

NX_BSD_ENABLE_DNS が定義されている場合は、NetX BSD は DNS クエリを送信して、ホスト名またはホストの IP の情報を取得できます。この機能には、事前に NetX DNS クライアントインスタンスが作成されていることが必要です。この DNS インスタンスに対する BSD のリンクは、外部 NX_DNS *_nx_dns_instance_ptr. を経由します。BSD ア

プリケーションがアドレスを指定して getnameinfo をコールするか、ホスト名を指定して getaddrinfo をコールすると、NetX BSD はさまざまな NetX DNS クライアントサービスをコールして、ホスト名または IP アドレスをそれぞれ取得します。アプリケーションにおける DNS クライアントの設定の詳細については、このドキュメントの最後に掲載されている「参考文献」セクションの説明に従って入手可能な Renesas Synergy プラットフォーム用の『*NetX DNS Client User Guide*』を参照してください。

未加工ソケットのサポート（NetX Duo BSD のみ）

NetX Duo BSD で未加工ソケットを使用するには、プロジェクトに NX_ENABLE_IP_RAW_PACKET_FILTERdefined を定義して NetX Duo ライブラリをコンパイルする必要があります。デフォルトでは、これは定義されていません。定義するには、[Synergy Configuration] ペインのプロジェクト -> [Properties] -> [Cross ARM C Compiler] -> [Preprocessor] の順に右クリックし、(**+**) アイコンをクリックしてこれを定義済みのシンボルのリストに追加します。

アプリケーションは、NetX Duo BSD サービスを使用する前に、nx_ip_raw_packet_enable サービスをコールして事前に作成した IP インスタンスの未加工ソケット処理を有効にする必要があります。NetX Duo BSD で未加工ソケットを作成するために、アプリケーションは socket サービスを使用して、以下のようにプロトコルファミリ、ソケットタイプおよびプロトコルを指定します。

sock_1 = socket(INT protocolFamily, INT socket_type, INT プロトコル)

NetX BSD は、protocolFamily に対して、IPv4 ソケットでは AF_INET、IPv6 ソケットでは AF_INET6、未加工ソケットでは AF_PACKET という値をサポートします。socket_type は、SOCK_RAW. に設定する必要があります。プロトコルはアプリケーションに固有です。

未加工パケットの送受信および未加工ソケットのクローズを行うために、アプリケーションは sendto、recvfrom、select、andsoc_close. などの UDP と同じ BSD サービスを使用するのが一般的です。未加工ソケットは、accept および listen BSD サービスのどちらもサポートしません。

- デフォルトでは、受信する IPv4 未加工データには IPv4 ヘッダーが含まれます。一方、受信する IPv6 未加工データには IPv6 ヘッダーは含まれません。
- デフォルトでは、未加工 IPv6 パケットまたは未加工 IPv4 パケットのいずれかを送信する場合、BSD ラッパーレイヤーによって、データの送信前に IPv6 ヘッダーまたは IPv4 ヘッダーが追加されます。

NetX Duo BSD は、includingIP_RAW_RX_NO_HEADER,、IP_HDRINCL、IP_RAW_IPV6_HDRINCL. などの追加の未加工ソケットオプションをサポートします。IP_RAW_RX_NO_HEADER が設定されている場合は、IPv4 ヘッダーが削除され、受信するデータと、レポートされるメッセージ長に IPv4 ヘッダーが含まれません。IPv6 ソケットの場合、デフォルトでは受信する未加工ソケットには IPv6 ヘッダーは含まれず、IP_RAW_RX_NO_HEADER オプションを設定した場合と同じ処理となります。アプリケーションは、setsockopt サービスを使用して IP_RAW_RX_NO_HEADER オプションをクリアできます。IP_RAW_RX_NO_HEADERoption がクリアされると、受信する IPv6 未加工データとレポートされるメッセージ長に IPv6 ヘッダーが含まれます。このオプションは、IPv4 と IPv6 の送信データには影響しません。

IP_HDRINCL が設定されている場合は、データの送信時にアプリケーションに IPv4 ヘッダーが含まれます。このオプションは、IPv6 での送信には影響せず、デフォルトでは定義されません。IP_RAW_IPV6_HDRINCL が設定されている場合は、データの送信時にアプリケーションに IPv6 ヘッダーが含まれます。このオプションは、IPv4 での送信には影響せず、デフォルトでは定義されません。

IP_HDRINCLand および IP_RAW_IPV6_HDRINCL は、IPv4 と IPv6 での受信には影響しません。

注 :BSD 4.3 ソケット仕様では、カーネルは未加工パケットを各ソケット受信バッファにコピーする必要があると規定されています。一方、NetX Duo BSD では、同じプロトコルを共有する複数のソケットが作成された場合は、その動作は定義されていません。

PPPoE の未加工パケットのサポート（NetX Duo BSD のみ）

PPPoE の未加工パケットのサポートを有効にするには、NX_BSD_RAW_PPPOE_SUPPORT をプロジェクトに定義する必要があります。定義するには、[Synergy Configuration] ペインのプロジェクト -> [Properties] -> [Cross ARM C Compiler] -> [Preprocessor] の順に右クリックし、(**+**) アイコンをクリックしてこれを定義済みのシンボルのリストに追加します。これについては、NetX Duo ライブラリを再ビルドする必要はありません。

次のコマンドで BSD ソケットを作成して、PPPoE の未加工パケットを処理します。

sockfd = socket(AF_PACKET, SOCK_RAW, プロトコル);

次のとおり、現在の BSD の実装では、AF_PACKET ファミリの 2 つのプロトコルタイプのみがサポートされています。

- ETHERTYPE_PPPOE_DISC:：PPPoE ディスカバリパケット。MAC データフレームでは、ディスカバリパケットにイーサネットフレームタイプ 0x8863 が格納されます。
- ETHERTYPE_PPPOE_SESS:：PPP セッションパケット。MAC データフレームでは、セッションパケットにイーサネットフレームタイプ 0x8864 が格納されます。

データタイプ sockaddr_ll（ll はリンクレイヤーを表します）は、PPPoE フレームの送受信時のパラメータを指定するために使用されます。

構造体 sockaddr_ll は、次のように宣言されます。

```
struct sockaddr_ll
{
USHORT  sll_family; /* Must be AF_PACKET */
USHORT  sll_protocol; /* LL frame type */
INT     sll_ifindex; /* Interface Index. */
USHORT  sll_hatype; /* Header type */
UCHAR   sll_pkttype; /* Packet type */
UCHAR   sll_halen; /* Length of address */
UCHAR   sll_addr[8]; /* Physical layer address */
};
```

構造体のすべてのフィールドが sendto() または recvfrom() によって使用されるとは限らないことに注意してください。PPPoE パケットの送受信を目的とした sockaddr_ll の設定方法については、以下の説明を参照してください。

指定されたプロトコルに関係なく、AF_PACKET ファミリに作成されたソケットは、PPPoE ディスカバリパケットまたは PPP セッションパケットを送信するために使用できます。PPPoE パケットの送信時に、アプリケーションは、MAC ヘッダー（宛先 MAC アドレス、送信元 MAC アドレス、およびフレームタイプ）など、適切なフォーマットの PPPoE フレームを格納するバッファを準備する必要があります。バッファのサイズには、14 バイトのイーサネットヘッダーが含まれます。

thesockaddr_ll 構造体では、sll_ifindex を使用して、このパケットの送信に使用される物理インタフェースが示されます。構造体の他のフィールドは使用されません。使用されないフィールドに設定された値は、BSD 内部プロセスで無視されます。

次のコードブロックは、PPPoE パケットの送信方法を示しています。

```
struct sockaddr_ll peer_addr;
/* Transmit the PPPoE frame using the primary network interface. */
peer_addr.sll_ifindex = 0;
n = sendto(sockfd, frame, frame_size, 0, (struct
sockaddr*)&peer_addr, sizeof(peer_addr));
```

戻り値は、送信されたバイト数を示します。PPPoE パケットはメッセージベースなので、送信が成功した場合、送信されたバイト数は 14 バイトのイーサネットヘッダーを含むパケットのサイズと一致します。

PPPoE パケットは、recvfrom() を使用して受信できます。受信バッファには、イーサネットの MTU サイズを持つメッセージを格納できるだけの大きさが必要です。受信された PPPoE パケットには、14 バイトのイーサネットヘッダーが含まれています。PPPoE パケットの受信時に PPPoE ディスカバリパケットを受信できるのは、プロトコル ETHERTYPE_PPPOE_DISC. で作成されたソケットに限られます。同様に、PPP セッションパケットを受信できるのは、プロトコル ETHERTYPE_PPPOE_SESS. で作成されたソケットに限られます。同じプロトコルタイプのソケットが複数作成されている場合、到着した PPPoE パケットは最初に作成されたソケットに転送されます。最初に作成されたソケットが閉じられている場合は、その次に作成されたソケットがパケットの受信に使用されます。

PPPoE パケットが受信されると、sockaddr_ll 構造体内の以下のフィールドが有効になります。

- sll_family:：BSD 内部で AF_PACKET に設定されます
- sll_ifindex:：パケットの受信元インタフェースを示します
- sll_protocol:：受信されたパケットのタイプとして、ETHERTYPE_PPPOE_DISC または ETHERTYPE_PPPOE_SESS に設定されます

NetX/NetX Duo BSD サポートモジュールの動作に関する重要な注意事項と制限事項

NetX BSD のビルド要件

- NetX ソース要素を構成に追加し、[Extended Notify Support] プロパティを有効に設定します。
- [ThreadX Source] 要素で次のプロパティを定義します。
  - TX_THREAD_EXTENSION_2 int bsd_errno;By デフォルトでは、すべての EXTENSION マクロが未定義です。bsd_errno は、TX_THREAD_EXTENSION_1 または TX_THREAD_EXTENSION_0. として定義することもできます。
- システムによっては、ネイティブ BSD での型定義と競合する可能性があります。その場合は、プロジェクトのプリプロセッサ定義に _POSIX_SOURCE を含めます。これを行うには、プロジェクトの最上位レベルを右クリックし、[Properties]->[C/C++ Build]->[Settings]->[Cross ARM C Compiler]（このプロジェクトプラットフォームを使用している場合）->[Preprocessor] の順に選択します。

NetX BSD のソケットオプション

NetX BSD ソケットオプションは、入力 option_level. の 1 つとなる setsockopt サービスを使用することにより、実行時にソケット単位で有効（または無効）にすることができます。option_level. には 2 つの設定があります。1 つ目のタイプは、ソケットレベルオプションのための SOL_SOCKET です。サポートされているオプションのリストは、次のとおりです。

- SO_BROADCASTIf：設定すると、NetX ソケットとの間でブロードキャストパケットの送受信が可能になります。これは NetX Duo のデフォルト動作です。すべてのソケットにこの機能があります。
- SO_ERRORUsed：getsockoptservice を使用して、指定されたソケットでの前回のソケット操作によるソケットのステータスを取得するために使用されます。すべてのソケットにこの機能があります。
- SO_KEEPALIVEIf：設定すると、TCP Keep Alive 機能が有効になります。そのためには、NX_TCP_ENABLE_KEEPALIVE を定義（NetX および NetX Duo 共通スタック要素で [TCP Keepalive] プロパティを有効に設定）して NetX Duo ライブラリをビルドする必要があります。デフォルトでは、この機能は無効になっています。
- SO_RCVTIMEOThis：NetX Duo BSD ソケットでパケットを受信する際の待機オプションを秒単位で設定します。デフォルト値は NX_WAIT_FOREVER（0xFFFFFFFF）、または非ブロッキングが有効な場合は NX_NO_WAIT（0x0）です。
- SO_RCVBUFThis：TCP ソケットのウィンドウサイズを設定します。デフォルト値 NX_BSD_TCP_WINDOW, は、BSD TCP ソケットの場合 64k に設定されます。65535 よりも大きいサイズを設定するためには、NX_TCP_ENABLE_WINDOW_SCALINGbe を定義（NetX および NetX Duo 共通スタック要素で [TCP Keepalive] プロパティを有効に設定）して NetX Duo ライブラリをビルドする必要があります。
- SO_REUSEADDRIf：設定すると、複数のソケットを 1 つのポートにマップできます。これの一般的な用途は TCP サーバーソケットです。これは NetX ソケットのデフォルト動作です。

*注 :SO_ERROR オプションでは、bsd_errno が定義されていることが必要です。bsd_errno, を定義するには、nx 要素で NetX 共通の下に NetX ソーススタックを追加します。または、NetX Duo を使用している場合、nxd 要素で NetX Duo 共通の下に NetX ソーススタックを追加します。その後、ThreadX ソーススタック要素を NetX Duo ソースに追加します。プロパティのリストをスクロールダウンし、TX_THREAD_EXTENSION マクロのいずれか（0 ～ 2）を選択します。値は次のように設定します。*int bsd_errno;

これについては、上記の「NetX BSD のビルド要件」セクションで説明しています。

もう 1 つのオプションタイプ IP_PROTO は、IP レイヤーに実装されるオプション用で、すべてのソケットに影響します。実行時の IP レベルオプションのリストを以下に示します。

IP_MULTICAST_TTL

これにより時刻が UDP ソケットに対してライブに設定されます。デフォルト値は、ソケットが作成された時点の NX_IP_TIME_TO_LIVE（0x80）です。この値は、ソケットサービスをコールする前にこのソケットオプションを指定して setsockopt をコールすることによって上書きできます。

- IP_ADD_MEMBERSHIPIf このオプションを設定すると、指定された IGMP グループに BSD ソケット（UDP ソケットのみに適用）が参加します。
- IP_DROP_MEMBERSHIPIf このオプションを設定すると、指定された IGMP グループから BSD ソケット（UDP ソケットのみに適用）が離脱します。

以下のオプションは、NetX Duo BSD でのみサポートされます。

- IP_HDRINCLIf このオプションを設定すると、コール元のアプリケーションは IP ヘッダーを付加する必要があります。また、オプションで、BSD で作成された未加工 IPv4 ソケットで送信するデータにアプリケーションヘッダーを付加できます。このオプションを使用するには、IP タスクで未加工ソケット処理を有効にする必要があります。詳細については、前の「未加工ソケットのサポート」セクションを参照してください。
- IP_RAW_IPV6_HDRINCLIf このオプションを設定すると、コール元のアプリケーションは IPv6 ヘッダーを付加する必要があります。また、オプションで、BSD で作成された未加工 IPv6 ソケットで送信するデータにアプリケーションヘッダーを付加できます。このオプションを使用するには、IP タスクで未加工ソケット処理を有効にする必要があります。詳細については、前の「未加工ソケットのサポート」セクションを参照してください。
- IP_RAW_RX_NO_HEADERIf オフにすると、BSD で作成された未加工 IPv6 ソケットで受信するデータに IPv6 ヘッダーが含まれます。デフォルトでは、BSD の未加工 IPv6 ソケットから IPv6 ヘッダーが削除され、パケット長に IPv6 ヘッダーは含まれません。設定すると、IPv4 タイプの BSD の未加工ソケットで受信するデータから IPv4 ヘッダーが削除されます。デフォルトでは、BSD の未加工 IPv4 ソケットに IPv4 ヘッダーが含まれ、パケット長に IPv4 ヘッダーが含まれます。このオプションは、IPv4 と IPv6 のどちらでも送信データには影響しません。未加工パケットのサポートの有効に関する詳細については、前の「未加工ソケットのサポート」セクションを参照してください。

オプション設定を取得するには、option_level を再度 SOL_SOCKET（ソケットレベルオプション）または IP_PROTOfor（IP レベルオプション）に設定し、オプション名を指定して getsockopt をコールします。

実行時のソケットレベルオプションの詳細は、概要セクションで説明されている『*NetX ™ BSD 4.3 Socket API Wrapper for NetX User's Guide* for the Renesas Synergy ™ Platform』および『*NetX Duo ™ BSD 4.3 Socket API Wrapper for NetX Duo User's Guide* for the Renesas Synergy ™ Platform』ドキュメントで確認できます。

- send、recv、sendto、recvfrom の各コールでは、MSG_DONTWAIT フラグと MSG_PEEK フラグのみがサポートされています。
- NetX BSD ソケットレベルオプションは以下に限定されています。
  - SO_BROADCAST
  - SO_ERROR
  - SO_KEEPALIVE
  - SO_RCVTIMEO
  - SO_RCVBUF
  - SO_REUSEADDR
- NetX BSD IP レベルオプションは以下に限定されています。
  - IP_MULTICAST_TTL
  - IP_RAW_IPV6_HDRINCL（NetX Duo BSD のみ）
  - IP_ADD_MEMBERSHIP
  - IP_DROP_MEMBERSHIP
  - IP_HDRINCL（NetX Duo BSD のみ。未加工ソケットを有効にする必要があります）

– IP_RAW_RX_NO_HEADER（NetX Duo BSD のみ。未加工ソケットを有効にする必要があります）

- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.12.4 アプリケーションへの NetX/NetX Duo BSD サポートモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo BSD サポートモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo BSD サポートモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo BSD サポートモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
NetX BSD Support	Threads	New Stack> X-Ware> NetX> Protocols> NetX BSD Support
NetX Duo BSD Support	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo BSD Support

次の図に示すように、NetX/NetX Duo BSD サポートモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

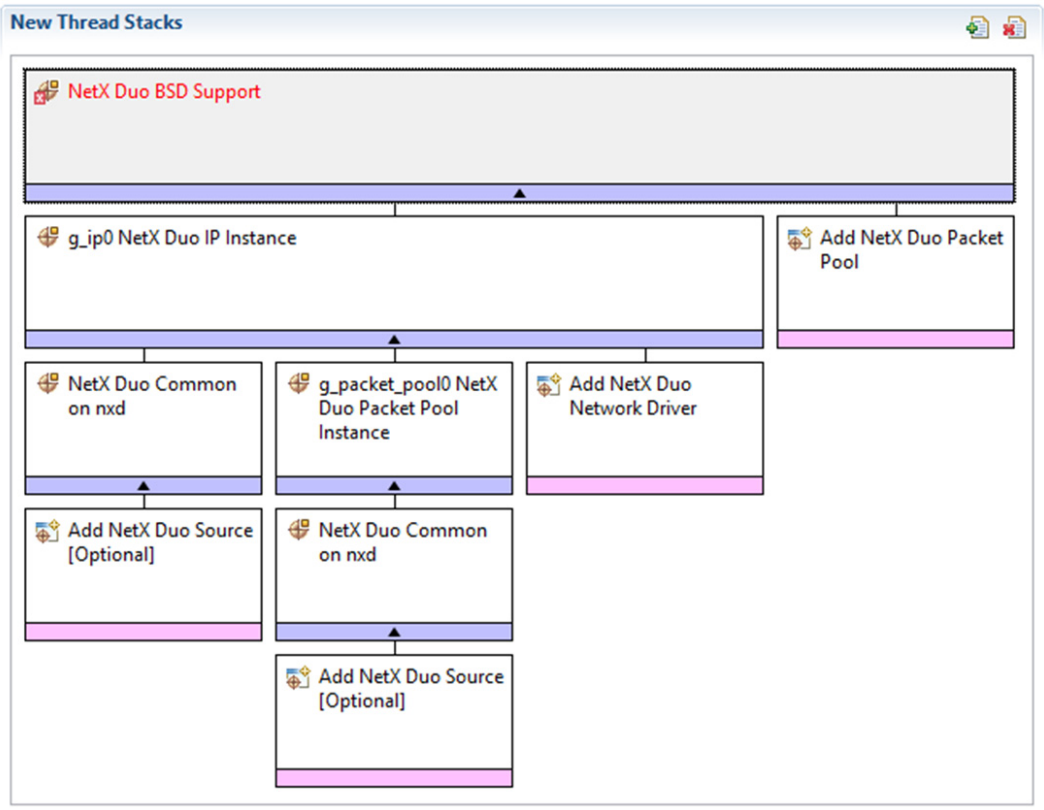


図 330:NetX/NetX Duo BSD サポートモジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.12.5 NetX/NetX Duo BSD サポートモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo BSD サポートモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発

生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo BSD サポートモジュールの構成設定

ISDE Property	Value	Description
NetX BSD Warning	Enable, Disable Default: Enable	NetX BSD warning selection
Internal thread stack size(bytes)	2048	Internal thread stack size selection
Internal thread priority	3	Internal thread priority selection
Name of generated initialization function	nx_bsd_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、異なる MAC アドレスまたは IP アドレスの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo BSD サポートのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
Default Gateway Address (use commas for separation)	0,0,0,0	Default gateway address selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection

ISDE Property	Value	Description
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function
Link status change callback	NULL	Link status change callback selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo BSD サポートモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo BSD サポートモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.12.6 アプリケーションでの NetX/NetX Duo BSD サポートモジュールの使用

アプリケーションで NetX および NetX Duo BSD サポートモジュールを使用する際の一般的な手順は次のとおりです。

NetX BSD クライアントの場合：

1) IP インスタンスの IP アドレスが有効な場合は、nx_ip_status_check API をポーリングします。
2) socket API を使用して、ソケットを作成します。
3) クライアントとサーバーの sockaddr_in を作成して、それぞれの IP アドレスとポートを定義します。
4) bind API を使用して、ローカルソースポートにバインドします。

5) connect API を使用して、サーバーに接続します。
6) getpeername および getsockname サービスを使用して、接続情報を取得します（オプション）。
7) send API を使用して、サーバーにパケットを送信します。
8) recv API を使用して、パケットを受信します。
9) soc_close API を使用して、ソケットを閉じます。

NetX BSD サーバーの場合：

1) IP インスタンスの IP アドレスが有効な場合は、nx_ip_status_check API をポーリングします。
2) socket API を使用して、マスタソケットを作成します。
3) sockaddr_in を作成して、サーバーの IP アドレスとポートを定義します。
4) bind API を使用して、ソケットをポートにバインドします。
5) listen API を使用して、クライアント要求を受信待ちするローカルソースポートを割り当てます。
6) select API を使用して、ソケット要求（リード、ライト、例外）を確認します。
7) accept API を使用して、クライアント要求を受け取り、2 番目のソケットに接続を渡します。
8) recv API を使用して、パケットを受信します。
9) send API を使用して、パケットを送信します。
10) soc_close API を使用して、ソケットを閉じます。

次の図は、一般的な手順を示した通常の動作フロー図です。

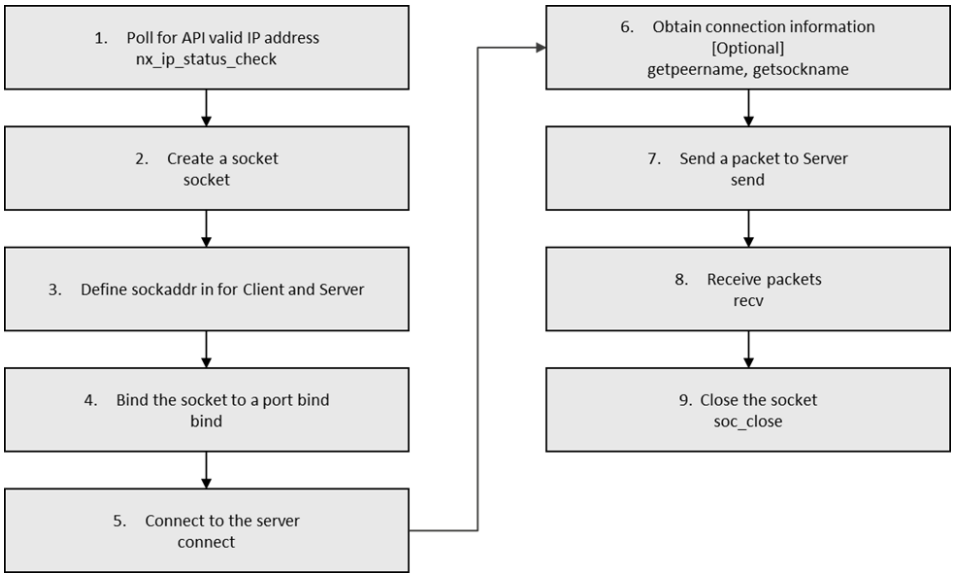


図 331:一般的な NetX BSD クライアントモジュールアプリケーションのフロー図

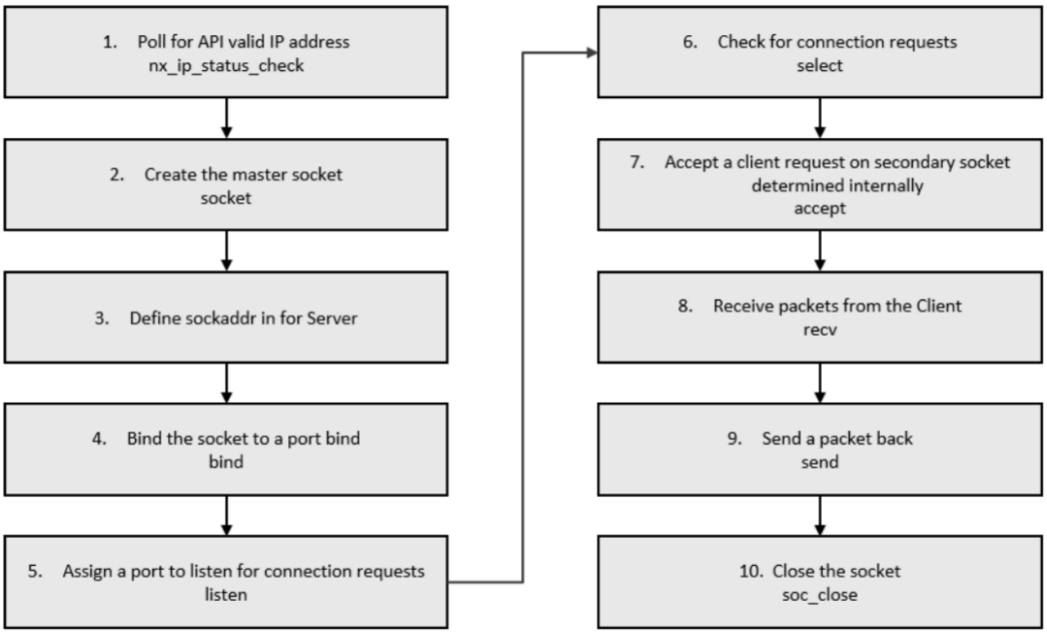


図 332:一般的な NetX BSD サーバーモジュールアプリケーションのフロー図

NetX Duo BSD クライアント

1) IP インスタンスの IP アドレスが有効な場合は、nx_ip_status_check API をポーリングします。
2) socket API を使用して、ソケットを作成します。
3) sockaddr_in（IPv6 の場合は sockaddr_in6）を作成して、クライアントとサーバーの IP（または IPv6）アドレスとポートを定義します。
4) bind API を使用して、ローカルソースポートにバインドします。
5) connect API を使用して、サーバーに接続します。
6) getpeername および getsockname サービスを使用して、接続情報を取得します（オプション）。
7) send API を使用して、パケットを送信します。
8) recv および send API を使用して、パケットを受信します。
9) soc_close API を使用して、ソケットを閉じます。

NetX Duo BSD サーバー

1) IP インスタンスの IP アドレスが有効な場合は、nx_ip_status_check API をポーリングします。
2) socket API を使用して、マスタソケットを作成します。
3) sockaddr_in（IPv6 の場合は sockaddr_in6）を作成して、サーバーの IP（または IPv6）アドレスとポートを定義します。ioctl API と setsockopt API を使用して、ソケットオプションを設定します。
4) bind API を使用して、ソケットをポートにバインドします。
5) listen API を使用して、クライアント要求を受信待ちするローカルソースポートを割り当てます。

6) select API を使用して、ソケット要求（リード、ライト、例外）を確認します。
7) accept API を使用して、クライアント要求を受け取り、2 番目のソケットに接続を渡します。
8) recv API を使用して、パケットを受信します。
9) send API を使用して、パケットを送信します。
10) soc_close API を使用して、ソケットを閉じます。

次の図は、一般的な手順を示した通常の動作フロー図です。

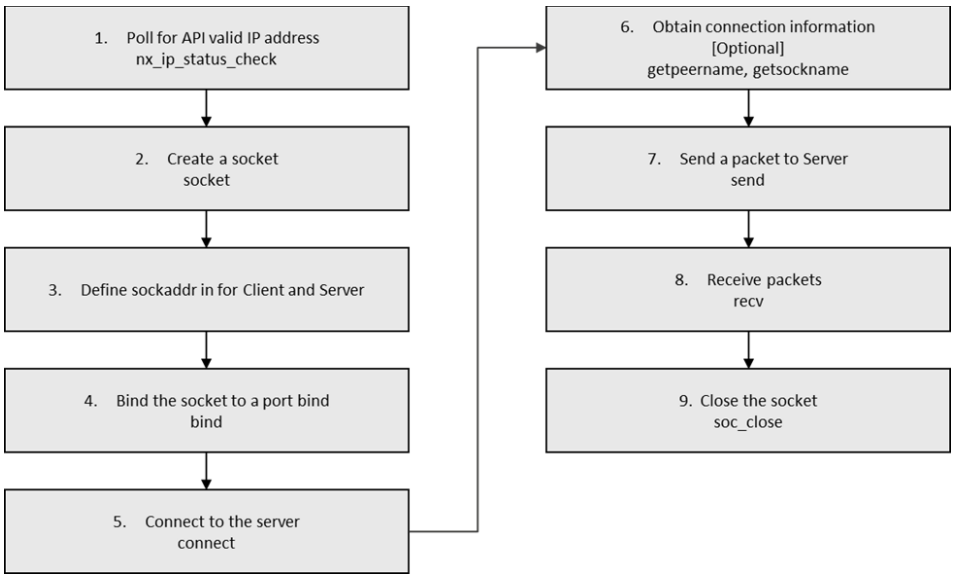


図 333:一般的な NetX Duo BSD クライアントモジュールアプリケーションのフロー図

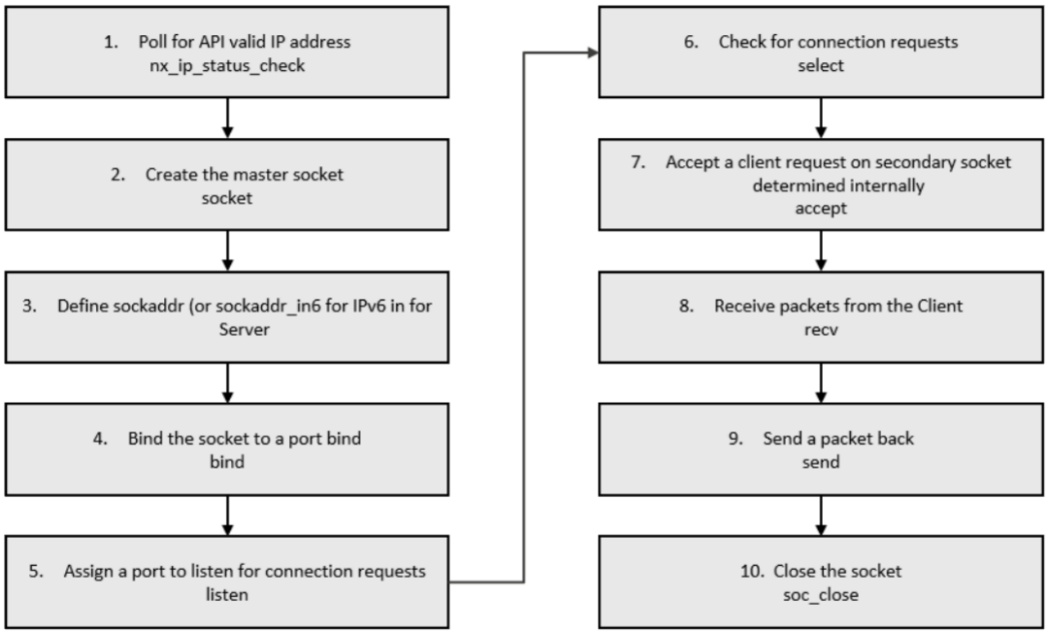


図 334:一般的な NetX Duo BSD サーバーモジュールアプリケーションのフロー図

4.3.13 NetX/NetX Duo DHCP クライアント

動的ホスト構成プロトコル（DHCP）は、IP アドレスとネットワークパラメータを取得するために使用されます。DHCP は、(静的なアドレス構成に限定されている) BOOTP の基本的な機能を拡張し、クライアントに IP アドレスを一定時間「リースする」ことによって完全に動的な IP アドレス割り当てを実現するように設計されています。DHCP は、BOOTP のように静的な IP アドレス割り当てを行うように構成することもできます。アプリケーションの IP アドレスは、NetX ™コンポーネントに提供されるパラメータの 1 つです。静的に、またはユーザー構成を通じて割り当てられた IP アドレスがアプリケーションにとって既知である場合、IP アドレスを提供することで問題が発生することはありません。アプリケーションが割り当てられる IP アドレスを認識しない場合や、認識する必要がない場合、IP アドレスをゼロとして NetX を初期化します。これにより、NetX に追加された DHCP クライアントコンポーネントは IP アドレスを動的に取得できます。

DHCP プロトコルは IPv4 に限定されているため、IPv6 ネットワークでは使用されません。そのため、DHCPv6 プロトコルが DHCPv6 サーバーからのグローバル IPv6 アドレスの動的な割り当てに使用されます。このガイドでは IPv4 バージョンの DHCP についてのみ説明しますが、その内容は NetX™ と NetX™ Duo に当てはまります。NetX と NetX Duo での使用方法の違いについては、注で明確に示します。このドキュメントでは表記を簡略化するために、NetX および NetX Duo DHCP for IPv4 を合わせて NetX DHCPv4 と表記します。

4.3.13.1 NetX/NetX Duo DHCP クライアントモジュールの特長

- NetX/NetX Duo DHCP クライアントモジュールは、RFC 2132、RFC 2131、および関連する RFC に準拠しています。
- このモジュールは、以下を行うためのハイレベル API を提供します。
- DHCP クライアントインスタンスの作成および削除
- DHCP クライアントの起動、停止、再初期化（DHCP クライアントプロトコルを再起動するため）
- サーバーへの特定の IP アドレスの要求

- DHCP クライアントを実行するネットワークインタフェースの指定
- アプリケーションが作成したパケットプールの DHCP クライアントへの提供

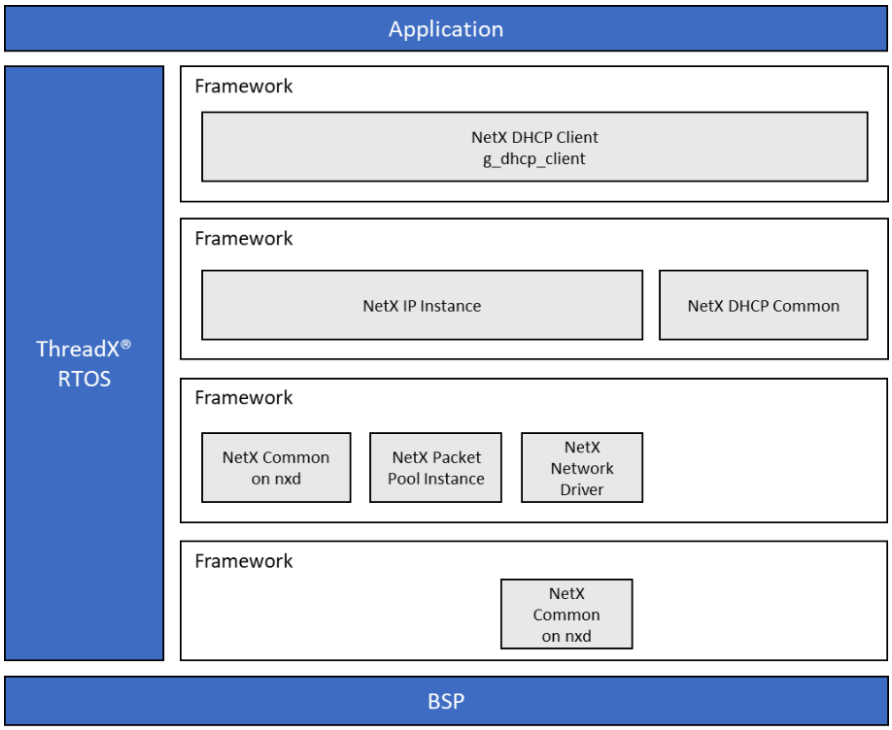


図 335:NetX/NetX Duo DHCP クライアントモジュールのブロック図

注:上の図で、NetX(または NetX Duo)ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.13.2 NetX/NetX Duo DHCP クライアントモジュールの API の概要

NetX/NetX Duo DHCP クライアントモジュールでは、DHCP クライアントの作成と起動のための API が定義されています。内部的には、DHCP クライアントが DHCP サーバーとのすべての通信を処理して IP アドレスを取得します。次の表には、主な API 関数のリスト、API 関数コールの例、各 API 関数の簡単な説明が記載されています。これ以外の関数コールについては、表の下の注で説明しているとおり『NetX DHCP Client User's Manual』で説明されています。ステータス戻り値の表は API 要約表の後にあります。

NetX/NetX Duo DHCP クライアントモジュールの API の要約

Function Name	Example API Call and Description
nx_dhcp_create	nx_dhcp_create(&my_dhcp, &my_ip, "My DHCP"); Create a DHCP instance.
nx_dhcp_clear_broadcast_flag	nx_dhcp_clear_broadcast_flag(&my_dhcp, NX_TRUE); Clear broadcast flag on Client messages.
nx_dhcp_delete	nx_dhcp_delete(&my_dhcp); Delete a DHCP instance.
nx_dhcp_decline	nx_dhcp_decline(&my_dhcp); Send Decline message to server.
nx_dhcp_force_renew	nx_dhcp_force_renew(&my_dhcp); Handle Server force renew message.

nx_dhcp_packet_pool_set	nx_packet_pool_create(&dhcp_pool, "DHCP Client Packet Pool", NX_DHCP_PACKET_PAYLOAD, pointer, (15 * NX_DHCP_PACKET_PAYLOAD));
	
	nx_dhcp_create(&dhcp_0, &ip_0, "janetsdhcp1");
	
	nx_dhcp_packet_pool_set(&my_dhcp, packet_pool_ptr);
	
	Set the DHCP Client packet pool. By default, the DHCP Client creates its own packet pool.

Function Name	Example API Call and Description
nx_dhcp_release	nx_dhcp_release(&my_dhcp); Send Release message to server.
nx_dhcp_reinitialize	nx_dhcp_reinitialize(&my_dhcp); Clear DHCP client network parameters and clear IP address and gateway registered with the IP instance.
nx_dhcp_request_client_ip	nx_dhcp_request_client_ip(&my_dhcp, IP(192,168,0,6), NX_TRUE); Request a specific IP address.
nx_dhcp_send_request	nx_dhcp_send_request(&my_dhcp, NX_DHCP_TYPE_INFORMREQUEST); Send DHCP message to server (only INFORM_REQUEST is allowed).
nx_dhcp_server_address_get	nx_dhcp_server_address_get(&dhcp_0, &server_address); Retrieve DHCP Client's DHCP server address.
nx_dhcp_set_interface_index	nx_dhcp_set_interface_index(&my_dhcp, 1); Specify the network interface to run DHCP Client.
nx_dhcp_start	nx_dhcp_start(&my_dhcp); Start DHCP processing.
nx_dhcp_state_change_notify	nx_dhcp_state_change_notify(&my_dhcp, my_state_change); Notify application of DHCP state change.

Function Name	Example API Call and Description
nx_dhcp_stop	nx_dhcp_stop(&my_dhcp); Stop DHCP processing.
nx_dhcp_user_option_retrieve	nx_dhcp_user_option_retrieve(&my_dhcp,NX_DHCP_OPTION_DNS_SVR, dns_ip_string, &size); Retrieve the specified DHCP option.
nx_dhcp_user_option_convert	nx_dhcp_user_option_convert(dns_ip_string); Convert four bytes to ULONG.
The following services require that Persistent client state be enabled	
nx_dhcp_suspend	nx_dhcp_suspend(&g_dhcp_client0); Suspend the DHCP Client thread.
nx_dhcp_resume	nx_dhcp_resume (&g_dhcp_client0); Resume the DHCP Client thread.
nx_dhcp_client_update_time_remaining	nx_dhcp_client_update_time_remaining(*g_dhcp_client0, 1000) This updates the time remaining on the IP lease by the input time in timer ticks, such as the time interval while the DHCP Client thread was suspended.
nx_dhcp_client_create_record	nx_dhcp_client_create_record(&g_dhcp_client0) This fills in a client record structure associated with the DHCP Client based on Client lease data.

Function Name	Example API Call and Description
nx_dhcp_client_restore_record	nx_dhcp_client_restore_record(&g_dhcp_client, client_record_ptr, time_elapsed) The Client record points to data to restore to the DHCP Client itself, and time elapsed is subtracted from the DHCP Client time remaining on its lease.

注 : これ以外の API 関数、関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作および定義の詳細については、関連する『Express Logic User's Manual』を参照してください。これは、下記の URL の Renesas Synergy SSP Web ページの一番下にある「X-Ware ™ Component Documents for Renesas Synergy」に含まれています。https://www.renesas.com/en-us/products/synergy/software/ssp.html.

ステータス戻り値

Name	Description
NX_SUCCESS	Successful API call.
NX_PTR_ERROR**	Invalid pointer input.
NX_THREADS_ONLY_CALLER_CHECKING**	Invalid caller of this service.
NX_INVALID_INTERFACE	NetX is not enabled on the input interface
NX_NOT_ENABLED	Not enabled to set the DHCP Client packet pool.
NX_DHCP_NOT_STARTED	DHCP Client not started.
NX_DHCP_NOT_BOUND	The IP address has not been leased so the current operation is not allowed.
NX_DHCP_INVALID_MESSAGE	Illegal message type to send.
NX_DHCP_BAD_INTERFACE_INDEX**	An invalid network interface supplied
NX_DHCP_UNKNOWN_OPTION	Unknown DHCP option to extract from DHCP server response
NX_DHCP_INVALID_IP_REQUEST**	Invalid address for the DHCP Client to request

Name	Description
NX_DHCP_INVALID_PAYLOAD	Packet pool for the DHCP Client has insufficient payload
NX_DHCP_ALREADY_STARTED	DHCP Client thread task has already started
NX_DHCP_PARSE_ERROR	Unable to parse requested option from Server response
NX_DHCP_DEST_TO_SMALL	Supplied buffer too small to hold the requested option data for user requesting option data

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

** これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX と NetX Duo でのエラーチェックサービスの詳細については、それぞれ『*NetX User Guide* for the Renesas Synergy ™ Platform』または『*NetX Duo User's Guide* for the Renesas Synergy ™ Platform』を参照してください。

4.3.13.3 NetX/NetX Duo DHCP クライアントモジュールの動作の概要

DHCP クライアントモジュールは、IP アドレスの取得、IP インスタンスへの IP アドレスの登録、および IP アドレスのリースが期限切れになる前の IP アドレスの更新の詳細をすべて処理します。

NetX IP インスタンスが作成されます。このインスタンスは IP アドレスがゼロであり、ユーザーデータグラムプロトコル（UDP）とアドレス解決プロトコル（ARP）がそれぞれ有効にされています。逆アドレス解決プロトコル（RARP）を有効にすることはできません。DHCP クライアントが作成されます。それにより、DHCP メッセージを送受信するための UDP ソケットが作成されます。デフォルトでは、[*Minimum packet payload size*] および [*Number of packets in packet pool*] の設定に基づいて、DHCP クライアントにより独自のパケットプールが作成されます（次の表を参照してください）。[*Minimum Client packet payload size*] は、DHCP データ、IP、UDP ヘッダー、および物理フレームヘッダーを格納できるだけの大きさである必要があります。

- イーサネットネットワークの場合、この最小ペイロードは 592 バイトで、これが [*Minimum Client packet payload size*] のデフォルト設定です。
- 他のネットワークタイプ（Wi-Fi など）の場合、フレームヘッダーサイズはこれよりも大きくなり、最小サイズもそれに合わせて大きくする必要があります。

パケットプールが作成されると、パケットペイロードが必要な最小ペイロードサイズよりも小さくないか、DHCP クライアントによって検証されます。

DHCP クライアントは、起動前に nx_dhcp_request_client_ip サービスを使用して特定の IP アドレスを要求し、ゼロでない IP アドレスを渡すことができます。通常、この要求は、それまでにデバイスに割り当てられていた IP アドレスがあり、同じ IP アドレスを維持する場合に役立ちます。注：サーバーはこの要求に応える義務はありません。

DHCP クライアントが起動すると、DHCP ポート（デフォルトでは 68）にソケットがバインドされ、このソケットを通じてパケットの送受信が開始されます。クライアントに IP アドレスが割り当てられると、クライアントは自動的にその IP アドレスを NetX に登録します。サーバーはネットワークマスクおよびネットワークゲートウェイを提供し、DHCP クライアントモジュールはその情報に基づいて NetX を更新します。

サーバーがクライアントに IP アドレスを割り当てるとき、DNS サーバーや NTP サーバーなど、他のネットワーク情報も提供することがあります。アプリケーションは nx_dhcp_user_option_retrieve サービスを使用することにより、これらの値を取得できます。

DHCP クライアントでは、IP リースの残り時間が追跡されます。更新日時になると、自動的に DHCP サーバーに更新要求が送信されます。サーバーが既にネットワーク上にない場合、または応答しない場合、クライアントはネットワーク上のすべての DHCP サーバーにブロードキャスト要求を送信します。更新も再バインドもされないままリースの有効期限が切れた場合、クライアントは NX_DHCP_STATE_INIT 状態に戻されます。そのデバイスは、引き続き同じ IP アドレスを使用できる場合があります。その後、DHCP サーバーが使用可能になってデバイスが IP アドレスを要求できるようになったら、以前の IP アドレスを使用することはできません。

トラフィックの多いネットワークでは、他の DHCP クライアントホスト向けの非特定の DHCP ブロードキャストパケットによって、DHCP クライアントソケットのキューがいっぱいになる可能性があります。DHCP クライアントソケットの受信キューがいっぱいになると、そのデバイス宛のパケットがドロップされる可能性があります。この問題を回避するために、DHCP クライアントは継続的にソケット内の非特定のブロードキャストパケットをクリアします。

DHCP クライアントモジュールは、nx_dhcp_user_option_add_callback_set API 関数を使用することで、コールバックを登録して DHCP プロトコルにオプションを追加できます。登録されたコールバックを使用して、DHCP オプション 60（[Vendor Class Identifier]）、DHCP オプション 61（[Client Identifier]）などのオプションを DHCP 要求に追加できます。DHCP モジュールでは複数のオプションをつなげることができます。これらのオプションは、通常のペイロードに収まるサイズである必要があります。Synergy コンフィギュレータには、デフォルトの DHCP オプション 60 追加コールバックが用意されています。デフォルトコールバックを有効にするには、コンフィギュレータで [DHCP Option addition] プロパティと [DHCP Option addition function] プロパティを [Enable] に設定する必要があります。

NetX/NetX Duo DHCP クライアントモジュールの動作に関する重要な注意事項と制限事項

開発時に、DHCP クライアントモジュールでパケットプールを作成するのではなく、以前に作成したパケットプールを使用することが考えられます。これを行うには、[*Use application packet pool*] オプションを有効にしてから、nx_dhcp_packet_pool_set サービスを使用して DHCP クライアントのパケットプールを設定します。

パケットペイロードが必要な最小パケットサイズよりも小さくないか、DHCP クライアントによって検証されます。

クライアントに提供する IP アドレスは、ローカルネットワークでの「一意性」をテストする必要があります。これは、DHCP プロトコルではサーバーにこの確認が求められていないためです。DHCP クライアントによる確認を構成するには、[*Send ARP probe*] オプションを有効にします。

DHCP クライアントは、割り当てられた IP アドレスを使用した一連の ARP「プローブ」をネットワークに対して送信します。この ARP 要求 / プローブに応答するホストがあれば、DHCP クライアントはサーバーに対して自動的に DECLINE メッセージを送信し、DHCP プロトコルを再起動して別の IP アドレスを要求します。そうでない場合、DHCP クライアントはバインドされた状態に進みます。DHCP プロトコルでのクライアントの状態は、以下のとおりです。

NX_DHCP_STATE_NOT_STARTED

NX_DHCP_STATE_INIT

NX_DHCP_STATE_SELECTING

NX_DHCP_STATE_REQUESTING

NX_DHCP_STATE_BOUND

NX_DHCP_STATE_RENEWING

NX_DHCP_STATE_REBINDING

注 :ARP プローブを有効にすると、NetX DHCP クライアントは NX_DHCP_STATE_BOUND 状態の前に NX_DHCP_STATE_ADDRESS_PROBING という一時的な状態になります。

アプリケーションは次の 2 つの方法のうちどちらかを使用して、DHCP クライアントが完了した（IP アドレスを取得した）ことを検出できます。1 つ目は、NX_IP_ADDRESS_RESOLVED オプションを指定して nx_ip_status_check サービスをコールする方法です。もう 1 つは、DHCP クライアントの状態が変化するとアプリケーションに通知する _nx_dhcp_state_change_notify サービスを使用する方法です。DHCP クライアントがバインドされた状態（state == NX_DHCP_STATE_BOUND）になると、このクライアントは有効な IP アドレスを取得したことになります。

DHCP クライアントのスレッドタスクを停止する必要がある場合は、nx_dhcp_stop サービスをコールします。クライアントを再起動するには、まず nx_dhcp_reinitialize サービスをコールして、DHCP クライアントのデータをクリアすると共に NetX に登録されているネットワークパラメータもクリアします。その後、nx_dhcp_start をコールして DHCP クライアントを再起動します。

- DHCP クライアントでは、INFORM_REQUEST メッセージはサポートされません。アプリケーションは nx_dhcp_send_request サービスを使用してこのメッセージを送信できますが、サーバーからのデータが抽出されて DHCP クライアントに保存されることはありません。
- _nx_dhcp_user_option_retrieve でサポートされるオプションは、以下に限られます。

NX_DHCP_OPTION_SUBNET_MASK
NX_DHCP_OPTION_TIME_OFFSET
NX_DHCP_OPTION_GATEWAYS
NX_DHCP_OPTION_TIMESVR
NX_DHCP_OPTION_DNS_SVR
NX_DHCP_OPTION_NTP_SVR
NX_DHCP_OPTION_DHCP_LEASE
NX_DHCP_OPTION_DHCP_SERVER
NX_DHCP_OPTION_RENEWAL
NX_DHCP_OPTION_REBIND

- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.13.4 アプリケーションへの NetX/NetX Duo DHCP クライアントモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo DHCP クライアントモジュールのいずれか、または両方を組み込む方法について説明します。

*注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。*NetX/NetX Duo DHCP クライアントモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo DHCP クライアントモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_dhcp_client0 NetX DHCP Client	Threads	**New Stack> X-Ware> NetX> Protocols> NetX DHCP Client**
g_dhcp_client0 NetX Duo DHCP IPv4 Client	Threads	**New Stack> X-Ware> NetX Duo> Protocols> NetX Duo DHCP IPv4 Client**

次の図に示すように、NetX/NetX Duo DHCP クライアントモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。

（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

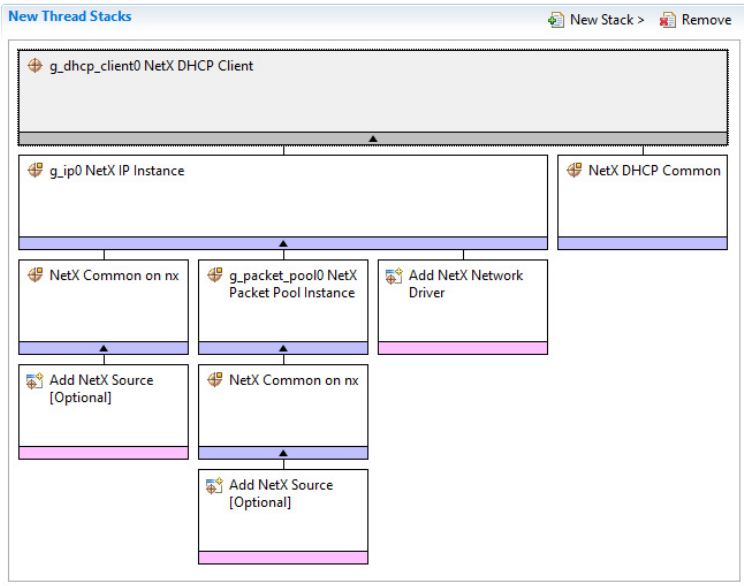


図 336:NetX/NetX Duo DHCP クライアントモジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.13.5 NetX/NetX Duo DHCP クライアントモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo DHCP クライアントモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo DHCP クライアントモジュールの構成設定

ISDE Property	Value	Description
Internal thread priority	3	Internal thread priority selection
Internal thread stack size (bytes)	NetX Default: 2048 NetX Duo Default: 4096	Internal thread stack size (bytes) selection
Timeout between DHCP messages processed (seconds)	1	Timeout between DHCP messages processed (seconds) selection
Use BOOTP	Enable, Disable Default: Disable	Use BOOTP selection
Send ARP probe	Enable, Disable Default: Disable	Send ARP probe selection
ARP probe wait time (seconds)	1	ARP probe wait time selection
Minimum ARP probe wait time (seconds)	1	Minimum ARP probe wait time selection
Minimum ARP probe wait time (seconds)	2	Minimum ARP probe wait time selection
ARP probe count	2	ARP probe count selection
Maximum retransmission timeout (seconds)	64	Maximum retransmission timeout (seconds) selection
Minimum renew timeout (seconds)	60	Minimum renew timeout (seconds) selection
Minimum retransmission timeout (seconds)	4	Minimum retransmission timeout (seconds) selection

ISDE Property	Value	Description
Client packet payload size (bytes)	592	Client packet payload size (bytes) selection
Number of packets in internal packet pool	5	Number of packets in internal packet pool selection
Persistent client state	Enable, Disable Default: Disable	Persistent client state selection
Use application packet pool	Enable, Disable Default: Disable	Use application packet pool selection
Maximum message size support	Enable, Disable Default: Disable	Maximum message size support selection
DHCP options buffer size (bytes)	312	DHCP options buffer size (bytes) selection
Maximum DHCP client state record on an interface	1	Maximum DHCP client state record on an interface selection
Wait before restarting the configuration process (seconds)	10	Wait before restarting the configuration process selection
Name	g_dhcp_client0	Module name
Name of generated initialization function	dhcp_client_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection
*DHCP Option addition	Enable, Disable Default: Enable	Enable or Disable feature to add DHCP Option to DHCP message

ISDE Property	Value	Description
*DHCP Option addition function	Enable, Disable Default: Enable	Enable or Disable Option Addition function
*Name of the DHCP option addition function	dhcp_user_option_add_client0	Name for the option add function provided by the user

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注 : NetX 実装でのみ使用可能な設定。*スタックモジュールのデフォルト以外の設定が望ましいこともあります。たとえば、異なるイーサネットインタフェースピンやリセットの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo DHCP クライアントのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	0,0,0,0	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection

ISDE Property	Value	Description
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

ISDE Property	Value	Description
Link status change callback	Default: NULL	Name of user defined callback function if needed; otherwise set as NULL.

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo DHCP 共通インスタンスの構成設定

ISDE Property	Value	Description
Type of Service for UDP requests	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost Default: Normal	Type of service UDP requests selection
Fragmentation option	Don't fragment, Fragment okay Default: Don't fragment	Fragmentation option selection
Time to live	128	Time to live selection
Packet Queue depth	5	Packet queue depth selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection

ISDE Property	Value	Description
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo DHCP クライアントモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo DHCP クライアントモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I^2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select **Peripherals > Connectivity:ETHERC > ETHERC1.RMII**

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.13.6 アプリケーションでの NetX/NetX Duo DHCP クライアントモジュールの使用

次の例は、実際に使用できる有効な IP、ARP、UDP が確立済みで、リンクが実行されているシステムを前提にしています。また、DHCP クライアントを起動する前に DHCP クライアントの DHCP 機能を設定します（特定の IP アドレスの要求、ブロードキャストフラグのクリア、DHCP クライアントが実行されるインタフェースの設定、またはユーザーオプションを構成するコールバックの設定）（オプション）。

一般的なアプリケーションで NetX/NetX Duo DHCP クライアントモジュールを使用する際の手順は次のとおりです。

1) DHCP クライアントを起動する前に DHCP クライアントの DHCP 機能を設定します（特定の IP アドレスの要求、ブロードキャストフラグのクリア、DHCP クライアントが実行されるインタフェースの設定）（オプション）。
2) nx_dhcp_start API を使用して DHCP を開始します。
3) nx_ip_status_check（NetX ライブラリサービスコール）をコールして IP アドレスの解決を待つか、DHCP クライアント状態変更コールバック関数で、バインドされた状態を確認します。
4) 有効な IP アドレスがリースされ、アプリケーションは NetX サービスを使用してパケットの送受信を開始できるようになります。
5) （DHCP クライアントのスレッドタスクが引き続き実行されている限り）DHCP クライアントは IP リースの残り時間に基づいて、IP リースの更新を自動的に要求します（オプション）。
6) DHCP クライアントのスレッドタスクを停止するために、nx_dhcp_stop API をコールします。
7) DHCP クライアントを再起動するために、nx_dhcp_reinitialize API をコールしてから *nx_dhcp_start API* をコールします（オプション）。
8) 既存の DHCP クライアント設定を追加または変更します（オプション）。

次の図は、一般的な手順を示した通常の動作フロー図です。

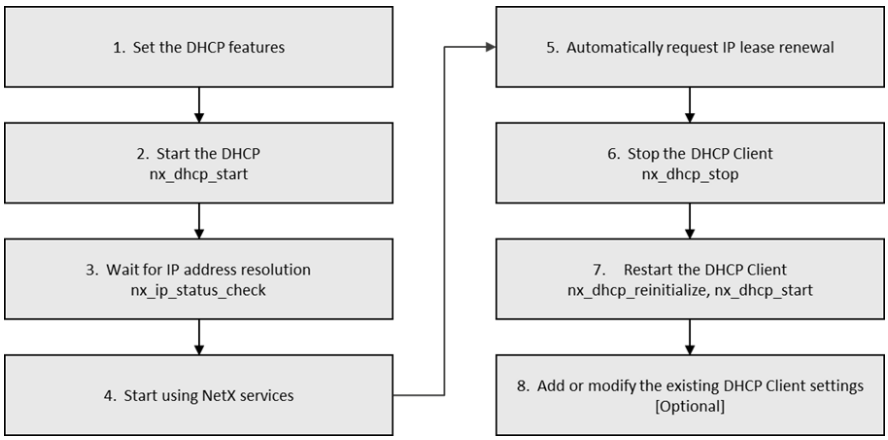


図 337:一般的な NetX/NetX Duo DHCP クライアントモジュールアプリケーションのフロー図

4.3.14 NetX/NetX Duo DHCP サーバー

動的ホスト構成プロトコル（DHCP）は、クライアントに IP アドレスを一定時間リースすることによって、DHCP サーバー割り当てと動的な IP アドレス割り当てを完全に自動化するように設計されています。

DHCP プロトコルは IPv4 に限定されているため、IPv6 ネットワークでは使用されません。そのため、DHCPv6 プロトコルが DHCPv6 サーバーからのグローバル IPv6 アドレスの動的な割り当てに使用されます。このガイドでは IPv4 バージョンの DHCP についてのみ説明しますが、その内容は NetX ™と NetX ™ Duo に当てはまります。NetX と NetX Duo での使用方法の違いについては、注で明確に示します。このドキュメントでは表記を簡略化するために、NetX および NetX Duo DHCP for IPv4 を合わせて NetX DHCPv4 と表記します。

4.3.14.1 NetX/NetX Duo DHCP サーバーモジュールの特長

- NetX DHCP は、RFC 2132、RFC 2131、および関連する RFC に準拠しています。
- また、以下を行うためのハイレベル API を提供します。
- DHCPv4 サーバーインスタンスの作成と削除
- DHCPv4 サーバーがクライアントにメッセージを送信するためのネットワークパラメータの設定
- 割り当て可能な IP アドレスのプールの作成
  - DHCP サーバータスクスレッドの開始と停止

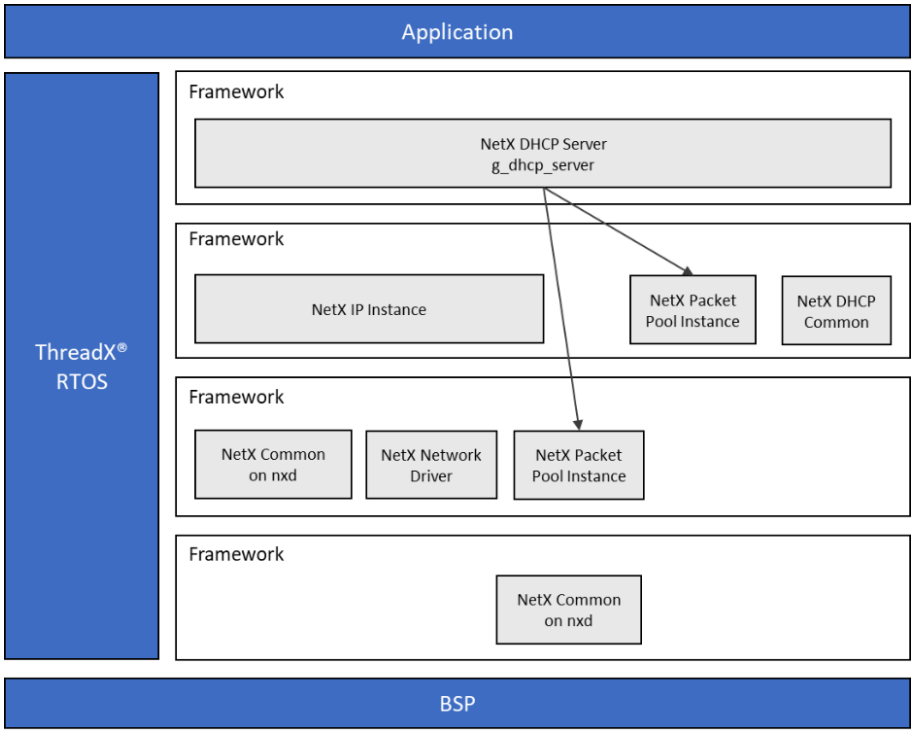


図 338:NetX/NetX Duo DHCP サーバーモジュールのブロック図

注:上の図で、NetX(または NetX Duo)ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.14.2 NetX/NetX Duo DHCP サーバーモジュールの API の概要

NetX DHCP サーバーでは、サーバーの作成、削除、移動、起動、停止、割り当て可能な IP アドレスのプールの作成、およびクライアントのネットワーク情報の設定のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX/NetX Duo DHCP サーバーモジュールの API の要約

Function Name	Example API Call and Description
nx_dhcp_server_create	nx_dhcp_server_create(&dhcp_server, &server_ip, pointer, DEMO_SERVER_STACK_SIZE, SERVER_IP_ADDRESS_LIST, "DHCP server", &server_pool); Create a DHCP Server instance.
nx_dhcp_create_server_ip_address_list	nx_dhcp_create_server_ip_list (&dhcp_server, iface_index, START_IP_ADDRESS_LIST, END_IP_ADDRESS_LIST, &addresses_added); Create pool of available IP addresses to assign to DHCP Clients on the specified network index.
nx_dhcp_clear_client_record	nx_dhcp_clear_client_record (&dhcp_server, &dhcp_client_ptr); Remove Client record in the Server database.
nx_dhcp_set_interface_network_parameters	nx_dhcp_set_interface_network_parameters(&dhcp_server, iface_index, NX_DHCP_SUBNET_MASK, NX_DHCP_DEFAULT_GATEWAY, NX_DHCP_DNS_SERVER); Set DHCP options for adding critical network parameters on specified interface in messages to Clients.

Function Name	Example API Call and Description
nx_dhcp_server_delete	nx_dhcp_server_delete(&dhcp_server); Delete a DHCP Server instance.
nx_dhcp_server_start	nx_dhcp_server_start(&dhcp_server); Start or resume DHCP Server processing.
nx_dhcp_server_stop	nx_dhcp_server_stop(&dhcp_server); Stop DHCP server processing.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

ステータス戻り値

Name	Description
NX_SUCCESS	Successful DHCP call
NX_PTR_ERROR*	Invalid pointer input
NX_DHCP_PARAMETER_ERROR	Invalid non-pointer input
NX_DHCP_INADEQUATE_PACKET_POOL_PAYLOAD	Packet payload too small error
NX_DHCP_NO_SERVER_OPTION_LIST	Missing option list; cannot create Server
NX_DHCP_SERVER_BAD_INTERFACE_INDEX	Index does not match addresses
NX_DHCP_INVALID_IP_ADDRESS	Invalid IP address or network interface for creating Server address list
NX_DHCP_INVALID_IP_ADDRESS_LIST	Illogical start/end IP addresses for Server list
NX_DHCP_INVALID_NETWORK_PARAMETERS	Invalid network parameters for DHCP messages to Client
NX_DHCP_SERVER_ALREADY_STARTED	The DHCP instance has already been started

Name	Description
NX_DHCP_SERVER_NOT_STARTED	DHCP Server not started
NX_CALLER_ERROR*	Invalid caller of service

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。注 : これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX と NetX Duo でのエラーチェックサービスの詳細については、それぞれ『NetX User Guide for the Renesas Synergy ™ Platform』または『NetX Duo User's Guide for the Renesas Synergy ™ Platform』を参照してください。*

4.3.14.3 NetX/NetX Duo DHCP サーバーモジュールの動作の概要

DHCP サーバーでは、DHCP クライアントからの要求の受け取りとレスポンスの送信に UDP プロトコルが使用されます。DHCP サーバーは、IP インスタンスの作成、ドライバーの初期化、UDP ソケットの作成、およびクライアント要求を受信するための既知の DHCP ポート 67 へのバインドの詳細をすべて処理します。

パケットプールが作成されると、DHCP サーバーに割り当てられます。DHCP サーバーは IP インスタンスが使用するパケットプール（IP のデフォルトパケットプール）を共有できますが、モジュールでサーバー用のパケットプールを別途作成することもできます。パケットペイロードは、DHCP データ、IP、UDP ヘッダー、および物理フレームヘッダーを格納できるだけの大きさである必要があります。DHCP データのサイズは [Size of the BOOT Buffer (bytes)] プロパティで設定され、デフォルトは 548 バイトです。

DHCP サーバーを起動する前に、アプリケーションは割り当て可能な IP アドレスのプールを作成する必要があります。それには、nx_dhcp_create_server_ip_address_list サービスをコールします。このサービスは、開始 IP アドレスと終了 IP アドレスを入力として受け取ります。サーバーは、アドレスがローカルネットワークのアドレスであるかどうかを検証します。DHCP サーバーサービスはインタフェースに固有で、IP アドレスリストの取り込みやネットワークパラメータの設定などを行います。DHCP サーバーの実行が想定されているネットワークインタフェースは、プライマリインタフェース（インデックス値がゼロ）です。開始 IP アドレスから順番に、IP アドレスのテーブルが埋められます。Theaddresses_added ポインタ入力は追加されるアドレスの数を返します。この値は、このテーブルのサイズ以下です。IP アドレステーブルのサイズは [Maximum size of an IP addresses list] プロパティで定義され、デフォルトは 20 です。このテーブルは、DHCP サーバーが DHCP クライアントの要求を受け取るネットワークインタフェースごとに存在します。

DHCP サーバーのクライアントレコードテーブルには各クライアントのレコード（DISCOVER 要求）が保存されます。レコードは、クライアントが割り当てられた IP アドレスを保持する限り存続します。クライアントが更新に失敗した場合、またはバインドされた（IP アドレスが割り当てられた）状態になる前に DHCP プロトコルに応答しなかった場合、レコードは削除されます。サーバーが DHCP 要求を受け取るすべてのネットワークインタフェースのすべてのクライアントレコードが 1 つのテーブルに格納されます。テーブルのサイズは [Size of client record table (units)] プロパティで設定され、デフォルトは 50 です。

DHCP サーバーが実行中になり、クライアントレコードが作成され、IP アドレスが割り当てられると、各クライアント IP リースの残り時間が定期的に確認されます。IP リースの長さは [Client IP address lease time (seconds)] プロパティで設定されます。デフォルト値の 0xFFFFFFFF は、実質的に永続的なリースを意味します。期限のあるリースを割り当てるには、標準的な時間を設定します。たとえば、10 日間（0x0d5930、874,800 秒）に設定します。割り当てられた IP リースの残り時間を DHCP サーバーが確認するインターバルは、1000 秒に設定されています。リースの有効期限が切れると、サーバーはクライアントレコードをクライアントレコードテーブルから削除し、その IP アドレスを割り当て可能な IP アドレスのプールに戻します。クライアントにメッセージは送信されません。クライアントはリースの有効期限が切れる前に更新または再バインドの要求を開始しておく必要があります。これが行われていない場合、クライアントがネットワークを離脱した可能性があります。

DHCP サーバーでは、各クライアントセッションの非アクティブタイムアウトも管理されます。クライアントがパケットを送信すると、そのクライアントの非アクティブタイムアウトはリセットされます。DHCP サーバーが残り時間を確認するインターバルは [Fast periodic timer interval to check valid sessions (ticks)] で設定され、デフォルトは 10 です。このセッションタイムアウトの場合、この値に 1 秒あたりのティック数の比率をかけて、10 秒というセッションタイムアウトが算出されます。クライアントレコードのセッションタイムアウトが切れると、そのクライアン

トの IP アドレスは割り当て可能な IP アドレスのプールに戻され、クライアントレコードはクリアされます。クライアントにメッセージは送信されません。

NetX/NetX Duo DHCP サーバーモジュールの動作に関する重要な注意事項と制限事項

- DHCP サーバーからクライアントに提供される重要なネットワークパラメータのオプションは、[Server option list] プロパティで定義され、デフォルト値は 1 3 6 です。これらは順番に、[Subnet Mask]、[Router/Gateway address]、[DNS Server IP address] のオプションコードです。オプションの数は [Server option list size] プロパティで設定され、デフォルトは 3 です。
- [DHCP Type]（オプション 53）と [DHCP Server Identifier]（オプション 54）は、サーバーが DHCP クライアントに提供する必要がある DHCP パラメータです。
- DHCP サーバーから提供されるオプションの選択肢は、[Subnet Mask]（オプション 1）、[Router/Gateway address]（オプション 3）、[DNS Server IP address]（オプション 6）の一部またはすべてに限定されています。したがって、[Server option list size] を 3 より大きい値に設定しても意味はありません。オプションのリストを 1、3、6 以外のオプションに設定しても意味はありません。
- NetX DHCP サーバーでは、割り当て可能な IP アドレスがネットワークの他の場所で使用されていないことの検証は行われません。割り当てられた IP アドレスの一意性はクライアントが確認することが想定されています。
- NetX DHCP サーバーでは、FORCE RENEW メッセージはサポートされません。
- NetX DHCP サーバーではネットワーク外からの DHCP 要求はサポートされないため、DHCP ヘッダーのリレーエージェントフィールドは Null のままです。
- DHCP サーバーでは、割り当てられた IP リースの残り時間が正しく更新されません。低速定期タイマのインターバルは 1000 ティックに設定されています。この値は内部では秒単位に変換されるため、サーバーが IP リースのタイムアウトを確認する実際のインターバルは、NetX デバイスでの 1 秒あたりティック数が 100 の場合、約 1000 * 100 です。クライアントのリース時間をデフォルト値の 0xFFFFFFFF のままにした場合、クライアントが解放するまでの永続的なリースとなり、このバグの影響を受けません。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.14.4 アプリケーションへの NetX/NetX Duo DHCP サーバーモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo DHCP サーバーモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo DHCP サーバーモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo DHCP サーバーモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_dhcp_Server0 NetX DHCP Server	Threads	New Stack> X-Ware> NetX> Protocols> NetX DHCP Server

Resource	ISDE Tab	Stacks Selection Sequence
g_dhcp_Server0 NetX Duo DHCP IPv4 Server	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo DHCP IPv4 Server

次の図に示すように、NetX/NetX Duo DHCP サーバーモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

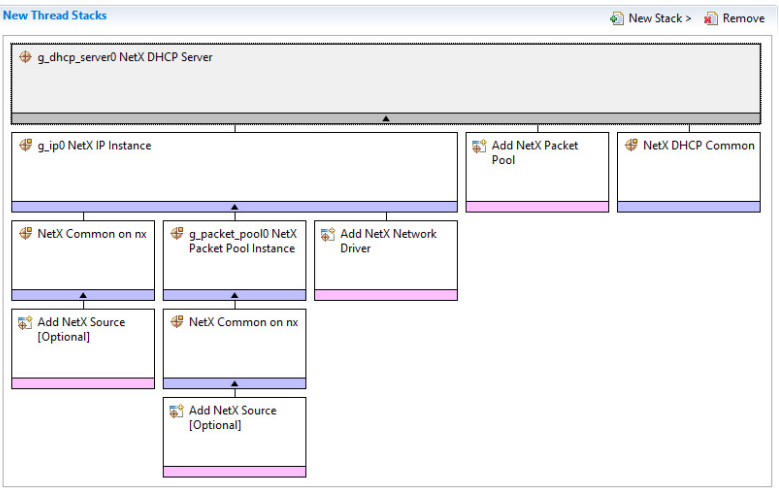


図 339:NetX/NetX Duo DHCP サーバーモジュールのスタック

上記のスタックで、NetX ネットワークドライバー (NetX Duo スタックの場合は NetX Duo ネットワークドライバー) はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.14.5 NetX/NetX Duo DHCP サーバーモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo DHCP サーバーモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注：次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo DHCP サーバーモジュールの構成設定

ISDE Property	Value	Description
Internal thread priority	1	Internal thread priority selection
Packet allocate timeout (seconds)	2	Packet allocate timeout selection
Fast periodic timer interval to check valid sessions (ticks)	10	Fast periodic timer interval to check valid sessions selection
DHCP Client Session timeout - multiple of Fast periodic interval (seconds)	10	DHCP Client session timeout selection
Client IP address default lease time (seconds)	0xFFFFFFFF	Client IP address lease time selection
Slow periodic timer inverval to check IP lease expiration (seconds)	1000	Slow periodic timer inverval to check IP lease expiration selection
Size of the array to contain options in client request (units)	12	Size of the array containing current requested options selection
Server option list (optional - use space for separation	1 3 6	Module server option list selection
Server option list size (optional)	3	Server option list size selection

ISDE Property	Value	Description
Size of the server host main buffer (bytes)	32	Size of the server host main buffer selection
Size of the current client hostname buffer (byte)	32	Size of the current client hostname buffer selection
Maximum size of an IP addresses list (units)	20	Maximum size of an IP addresses list selection
Size of the client record table (units)	50	Size of the client record table selection
Size of the BOOT buffer (bytes)	548	Size of the BOOT buffer selection
Name	g_dhcp_server0	Module name
Internal thread stack size (bytes)	NetX Default: 2048 NetX Duo Default: 4096	Internal thread stack size selection
Name of generated initialization function	nx_dhcp_server_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

スタックモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、異なるイーサネットインタフェースピンやリセットの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo DHCP サーバーのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	0,0,0,0	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection

ISDE Property	Value	Description
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo DHCP 共通インスタンスの構成設定

ISDE Property	Value	Description
Type of Service for UDP requests	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost Default: Normal	Type of service UDP requests selection
Fragmentation option	Don't fragment, Fragment okay Default: Don't fragment	Fragmentation option selection
Time to live	128	Time to live selection
Packet Queue depth	5	Packet queue depth selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo DHCP サーバーモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo DHCP サーバーモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I^2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin

Property	Value	Description
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.14.6 アプリケーションでの NetX/NetX Duo DHCP サーバーモジュールの使用

一般的なアプリケーションで NetX/NetX Duo DHCP サーバーモジュールを使用する際の手順は次のとおりです。

1) nx_dhcp_create_server_ip_address_list API を使用して、割り当て可能な IP アドレスのプールを作成します。
2) nx_dhcp_set_interface_network_parameters API を使用して、サーバーから返されるネットワークパラメータを設定します。
3) nx_dhcp_server_start API で DHCPv4 サーバーを起動します。

次の図は、一般的な手順を示した通常の動作フロー図です。

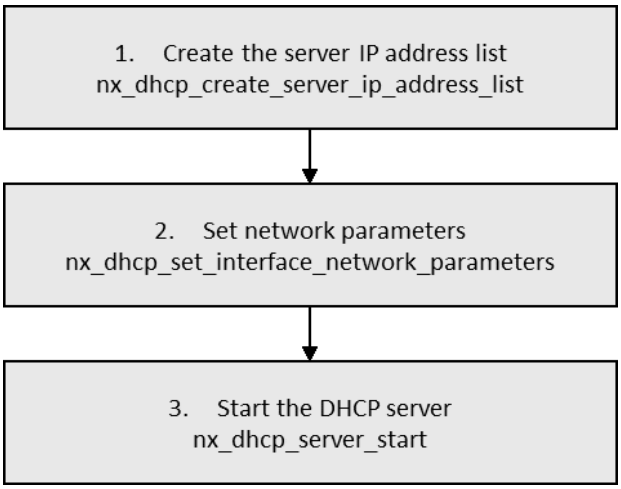


図 340:一般的な NetX/NetX Duo DHCP サーバーモジュールアプリケーションのフロー図

4.3.15 NetX Duo DHCP IPv6 クライアント

動的ホスト構成プロトコル（DHCP）は、IP アドレスとネットワークパラメータを取得するために使用されます。DHCP は、(静的なアドレス構成に限定されている) BOOTP の基本的な機能を拡張し、クライアントに IP アドレスを一定時間「リースする」ことによって完全に動的な IP アドレス割り当てを実現するように設計されています。DHCP

は、BOOTP のように静的な IP アドレス割り当てを行うように構成することもできます。アプリケーションの IP アドレスは、NetX ™コンポーネントに提供されるパラメータの 1 つです。静的に、またはユーザー構成を通じて割り当てられた IP アドレスがアプリケーションにとって既知である場合、IP アドレスを提供することで問題が発生することはありません。アプリケーションが割り当てられる IP アドレスを認識しない場合や、認識する必要がない場合、IP アドレスをゼロとして NetX を初期化します。これにより、NetX に追加された DHCP クライアントコンポーネントは IP アドレスを動的に取得できます。

このドキュメントでは、NetX Duo DHCPv6 クライアントの API についてと、この API を使用して IPv6 アドレスを取得する方法を説明しています。IPv6 ネットワークでは、DHCPv6 を（DHCP の代わりに）使用することで、DHCPv6 サーバーからグローバル IPv6 アドレスの動的な割り当てを行います。DHCPv6 の機能は多くが同じですが、いくつかの機能強化も行われています。

4.3.15.1 NetX Duo DHCP IPv6 クライアントモジュールの特長

- NetX Duo DHCPv6 クライアントは、RFC 3315、RFC 3646、および関連する RFC に準拠しています。
- また、以下を行うためのハイレベル API を提供します。
  - DHCPv6 クライアントインスタンスの作成と削除
  - DHCPv6 クライアントの起動と停止
  - メッセージの送信と処理
  - DHCPv6 クライアントからの DHCPv6 データの取得

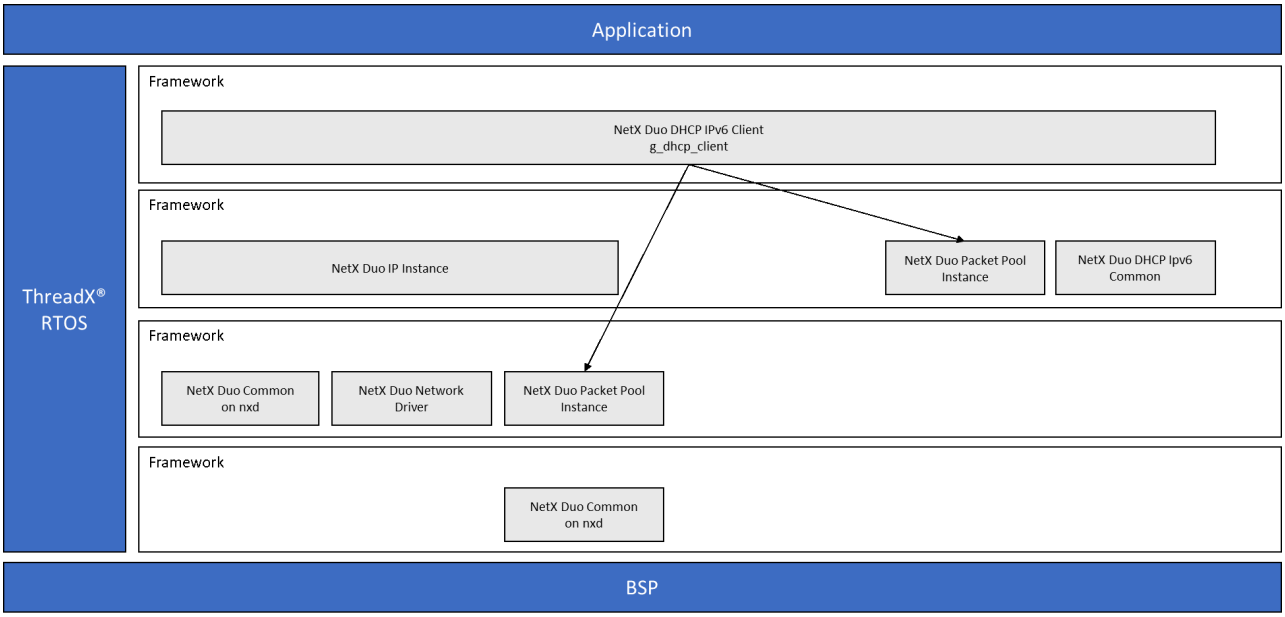


図 341:NetX Duo DHCP IPv6 クライアントモジュールのブロック図

注 : 上の図で、NetX Duo ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.15.2 NetX Duo DHCP IPv6 クライアントモジュールの API の概要

NetX Duo DHCPv6 クライアントフレームワークでは、クライアント情報の作成、削除、追加、取得のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX Duo DHCP IPv6 クライアントモジュールの API の要約

Function Name	Example API Call and Description
nx_dhcpv6_client_create	nx_dhcpv6_client_create(&dhcp_0, &ip_0, "DHCPv6 Client", &pool_0,NULL, NULL, pointer, 2048, dhcpv6_state_change_notify, dhcpv6_server_error_handler); Create a DHCPv6 Client instance.
nx_dhcpv6_client_delete	nx_dhcpv6_client_delete(&my_dhcp); Delete a DHCPv6 Client instance.
nx_dhcpv6_client_set_interface	nx_dhcpv6_client_set_interface(&dhcp_0, index); Set the Client network interface for communications with the DHCPv6 Server.
nx_dhcpv6_create_client_duid	nx_dhcpv6_create_client_duid(&dhcp_0, NX_DHCPV6_DUID_TYPE_LINK_TIME, NX_DHCPV6_HW_TYPE_IEEE_802, 0) Create a DHCPv6 Client DUID.
nx_dhcpv6_create_client_ia	nx_dhcpv6_create_client_ia(&dhcp_0, &ipv6_address, NX_DHCPV6_PREFERRED_LIFETIME, NX_DHCPV6_VALID_LIFETIME); Legacy Add a DHCPv6 Client Identity Address (IA).

Function Name	Example API Call and Description
nx_dhcpv6_create_client_iana	nx_dhcpv6_create_client_iana(&dhcp_0, DHCPV6_IA_ID, DHCPV6_T1, DHCPV6_T2); Create a DHCPv6 Client Identity Association for Non-Temporary Addresses (IANA).
nx_dhcpv6_add_client_ia	nx_dhcpv6_add_client_ia(&dhcp_0, &ipv6_address, NX_DHCPV6_PREFERRED_LIFETIME, NX_DHCPV6_VALID_LIFETIME); Add a DHCPv6 Client Identity Address (IA).
nx_dhcpv6_get_client_duid_time_id	nx_dhcpv6_get_client_duid_time_id(&dhcp_0, &time_ID); Get the time ID from DHCPv6 Client DUID.

nx_dhcpv6_get_ip_address	nx_dhcpv6_get_IP_address(&dhcp_0, &ipv6_address);
	nxd_ipv6_address_set(&ip_0, 0, &ipv6_address, 64, &address_index);
	
	Get the global IPv6 address assigned to the DHCPv6 client*.*

nx_dhcpv6_get_lease_time_data	nx_dhcpv6_get_lease_time_data(&dhcp_0, &T1, &T2, &preferred_lifetime, &valid_lifetime); Get T1 and T2 in the Identity Association (IANA) leased to the DHCPv6 Client.
nx_dhcpv6_get_iana_lease_time	nx_dhcpv6_get_iana_lease_time(&dhcp_0, &T1, &T2); Get T1, T2, valid and preferred lifetimes for the DHCPv6 Client IPv6 address by address index.

Function Name	Example API Call and Description
nx_dhcpv6_get_valid_ip_address_count	nx_dhcpv6_get_valid_ip_address_count(&dhcp_0, &address_count); This service retrieves the count of the Client's valid IPv6 addresses. A valid IPv6 address is bound (assigned) to the Client and registered with the IP instance. Also useful for determining if the DHPCv6 Client has reached the bound state.
nx_dhcpv6_get_valid_ip_address_lease_time	nx_dhcpv6_get_valid_ip_address_lease_time(&dhcp_0, &ip_address, &preferred_lifetime, &valid_lifetime); Get T1, T2, valid and preferred lifetimes for the DHCPv6 Client IPv6 address by address index.
nx_dhcpv6_get_DNS_server_address	nx_dhcpv6_get_DNS_server_address(&dhcp_0, index, &server_address); Get DNS Server address at the specified index into the DHCPv6 Client DNS server list.
nx_dhcpv6_get_other_option_data	nx_dhcpv6_get_other_option_data(&dhcp_0, option_code, buffer); Get the specified option data, such as domain name or time zone server.
nx_dhcpv6_get_time_accrued	nx_dhcpv6_get_time_accrued(&dhcp_0, &time_accrued); Get the time accrued the global IPv6 address lease has been bound to the DHCPv6 Client.
nx_dhcpv6_get_time_server_address	nx_dhcpv6_get_time_server_address(&dhcp_0, index, &server_address); Get Time Server address at the specified index into the DHCPv6 Client Time server list.

Function Name	Example API Call and Description
nx_dhcpv6_reinitialize	nx_dhcpv6_reinitialize(&dhcp_0); Reinitialize the DHCPv6 for restarting the DHCPv6 Client state machine and rerunning the DHCPv6 protocol.
nx_dhcpv6_request_confirm	nx_dhcpv6_request_confirm(&dhcp_0); Send a CONFIRM request to the Server.
nx_dhcpv6_request_inform_request	nx_dhcpv6_request_inform_request(&dhcp_0); Send an INFORM REQUEST message to the Server.
nx_dhcpv6_request_option_DNS_server	nx_dhcpv6_request_option_DNS_server(&dhcp_0, NX_TRUE); Add the DNS server option to the Client option request data in request messages to the Server.
nx_dhcpv6_request_option_FQDN	nx_dhcpv6_request_option_FQDN(&dhcp_0, "DHCPv6_Client", NX_DHCPV6_CLIENT_DESIRES_NO_SERVER_DNS_UPDATE); Add the FQDN option to the Client option request data in request messages to the Server.
nx_dhcpv6_request_option_domain_name	nx_dhcpv6_request_option_domain_name(&dhcp_0, NX_TRUE); Add the domain name option to the Client option request data in request messages to the Server.

Function Name	Example API Call and Description
nx_dhcpv6_request_option_time_server	nx_dhcpv6_request_option_time_server(&dhcp_0, NX_TRUE); Add the time server option to the Client option request data in request messages to the Server.
nx_dhcpv6_request_option_timezone	nx_dhcpv6_request_option_timezone(&dhcp_0, NX_TRUE); Add the time zone option to the Client option request data in request messages to the Server.
nx_dhcpv6_request_release	nx_dhcpv6_request_release(&dhcp_0); Send a RELEASE request to the Server.
nx_dhcpv6_request_solicit	nx_dhcpv6_request_solicit(&dhcp_0); Send a DHCPv6 SOLICIT request to any Server on the Client network (broadcast).
nx_dhcpv6_request_solicit_rapid	nx_dhcpv6_request_solicit_rapid(&dhcp_0); Send a DHCPv6 SOLICIT request to any Server on the Client network (broadcast) with the Rapid Commit option set.
nx_dhcpv6_resume	nx_dhcpv6_resume(&dhcp_0); Resume DHCPv6 Client processing.
nx_dhcpv6_set_time_accrued	nx_dhcpv6_set_time_accrued(&dhcp_0, time_accrued); Set the time accrued on the global Client IPv6 address lease in the Client record.

Function Name	Example API Call and Description
nx_dhcpv6_start	nx_dhcpv6_start(&dhcp_0); Start the DHCPv6 Client thread task. Note this is not equivalent to starting the DHCPv6 state machine and does not send a SOLICIT request.
nx_dhcpv6_stop	nx_dhcpv6_stop(&dhcp_0); Stop the DHCPv6 Client thread task.
nx_dhcpv6_suspend	nx_dhcpv6_suspend(&dhcp_0); Suspend the DHCPv6 Client thread task.
The following services are available if NX_DHCPV6_CLIENT_RESTORE_STATE is defined for the project:	
nx_dhcpv6_client_get_record	nx_dhcpv6_client_get_record(dhcpv6_ptr, client_record_ptr); Obtain a record of the client state (to save to non-volatile memory)
nx_dhcpv6_client_restore_record	nx_dhcpv6_client_restore_record(dhcpv6_ptr, client_record_ptr, time_elapsed); Apply saved client record to the current Client instance. Note the DHCPv6 Client thread must be not be running when this task is called.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

ステータス戻り値

Name	Description
NX_SUCCESS	Successful API call.

Name	Description
NX_PTR_ERROR*	Invalid pointer input.
NX_CALLER_ERROR*	Must be called from thread.
NX_DHCPV6_PARAM_ERROR	Invalid non pointer input.
NX_INVALID_INTERFACE	Invalid interface index input.
NX_DHCPV6_UNSUPPORTED_DUID_TYPE	DUID type unknown or not supported.
NX_DHCPV6_UNSUPPORTED_DUID_HW_TYPE	DUID hardware type unknown or not supported.
NX_DHCPV6_IA_ADDRESS_ALREADY_EXIST	Duplicate IA address.
NX_DHCPV6_REACHED_MAX_IA_ADDRESS	IA exceeds the max IAs Client can store.
NX_DHCPV6_INVALID_IA_ADDRESS	Invalid (e.g. null) IA address in IA.
NX_DHCPV6_IA_ADDRESS_NOT_VALID	IPv6 address successfully assigned.
NX_DHCPV6_UNKNOWN_OPTION	Unknown/unsupported option code.
NX_DHCPV6_ALREADY_STARTED	DHCPv6 Client is already running.
NX_DHCPV6_NOT_STARTED	DHCPv6 Client task not started.
NX_DHCPV6_MISSING_REQUIRED_OPTIONS	Client missing required options.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

注 : これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX と NetX Duo でのエラーチェックサービスの詳細については、それぞれ『NetX User Guide for the Renesas Synergy ™ Platform』または『NetX Duo User's Guide for the Renesas Synergy ™ Platform』を参照してください。*

4.3.15.3 NetX Duo DHCP IPv6 クライアントモジュールの動作の概要

DHCPv6 プロトコルでは、IPv6 アドレスを動的に取得するために UDP も使用されます。DHCPv6 クライアントが作成される前に、NetX Duo IP インスタンスが自動的に作成され、その IP インスタンスで UDP、IPv6、ICMPv6 が有効になります。DHCPv6 クライアントが作成されるときに、UDP ソケットが作成されてポート 546 にバインドされます。IP インスタンスの [*IPv6 Global Address*] プロパティはゼロの IPv6 アドレスに固定されます。そうでない場合、IPv4 アドレスは任意のネットワーク IP アドレスにすることができます。IP インスタンスの [*IPv6 Link Local Address*] プロパティがゼロに設定されている場合、DHCPv6 クライアントは MAC アドレスに基づいてリンクローカルアドレスを作成します。MAC アドレスは NetX ポート ETHER インスタンスで設定されています。[*Channel 1 MAC Address*

High Bits] プロパティと [*Channel 1 MAC Address Low Bits*] プロパティを参照してください。これがサーバーへの DHCPv6 メッセージのソースアドレスになります。

グローバル IPv6 アドレスの割り当てを要求するプロセスを開始するには、クライアントがまず nx_dhcpv6_request_solicit サービスを使用して SOLICIT メッセージをブロードキャストします。IPv6 では、ブロードキャストアドレスは All_DHCP_Relay_Agents_and_Servers アドレス（FF02::1:2.）です。DHCPv6 サーバーはクライアントのグローバル IPv6 アドレス（リンクローカルアドレスではなく）、IPv6 アドレスのリース時間、およびクライアントから要求された追加情報があればそれも含む ADVERTISE メッセージで応答します。DHCPv6 プロトコルでは、クライアントはネットワーク上のすべての DHCPv6 サーバーから ADVERTISE メッセージを受け取るために一定時間待つ必要があります。クライアントでは、各 ADVERTISE メッセージを有効なメッセージにするための前処理が行われ、さまざまな DHCPv6 パラメータのオプションデータがスキャンされます。また、プリファレンスオプションのプリファレンス値がサーバーから提供されている場合、その値が確認されます。複数の ADVERTISE メッセージが受信された場合、NetX DHCPv6 クライアントは待機期間が終了するまでに受信した ADVERTISE メッセージの中でプリファレンス値が最も高いものを選択します。プリファレンス値が 255 の ADVERTISE メッセージを受信した場合は、ただちにそのメッセージを受け入れ、その後の ADVERTISE メッセージはすべて破棄します。

クライアントでは、ADVERTISE メッセージからデータが抽出され、クライアントが選択したサーバーを指定した REQUEST メッセージが（すべての DHCPv6 サーバーに伝達されるように）ブロードキャストされます。そのサーバーでは、割り当てられたアドレスの情報とリース時間が確認されて REPLY メッセージが送信され、プロトコルが完了します。

DHCPv6 クライアントはバインドされた状態に昇格し、割り当てられた IPv6 アドレスが自動的に IP インスタンスに登録されます。

アドレス割り当て成功の通知と検証

DHCPv6 クライアントが IPv6 アドレスにバインドされたことをアプリケーションが知る方法は 2 つあります。1 つは、nx_dhcpv6_get_IP_address サービスを使用して DHCPv6 クライアントに直接、有効な IPv6 アドレスをクエリするもので、この方法は割り当てられた 1 つのグローバル IPv6 アドレスのみを使用するアプリケーション用クライアント向けです（一般的なケース）。もう 1 つは nx_dhcpv6_get_valid_ip_address_count サービスを使用するもので、これは複数の IPv6 アドレスが割り当てられたクライアント向けです。2 番目の方法では、DHCPv6 クライアントのスタック要素の [*Name of state change notification function*] プロパティ * で、DHCPv6 クライアントに状態変更コールバックを構成する必要があります。

DHCPv6 クライアントがサーバーからレスポンスを受け取ったにもかかわらずサーバーがアドレスを割り当てることができない場合、サーバーはエラーステータスを返します。DHCPv6 クライアントのスタック要素の [*Name of server error handler*] プロパティで DHCPv6 クライアントのサーバーエラーコールバックを構成しておくと、このエラーステータスを受信したことがアプリケーションに通知されます。

これらのコールバックはアプリケーションで定義する必要があります。これらのコールバック関数は DHCPv6 クライアントのスレッドタスクからコールされるためです。クライアントアプリケーションは、DHCPv6 クライアントのミューテックス制御（nx_dhcpv6_start、nx_dhcpv6_stop), など）を必要とする NetX Duo DHCPv6 クライアントのサービスや、コールバックからメッセージを直接送信する API（nx_dhcpv6_request_release). など）をコールすることはできません。

DHCP IPv6 プロトコルの状態は、以下のとおりです。

- NX_DHCPV6_STATE_INIT
- NX_DHCPV6_STATE_SENDING_SOLICIT
- NX_DHCPV6_STATE_SENDING_REQUEST
- NX_DHCPV6_STATE_SENDING_RENEW
- NX_DHCPV6_STATE_SENDING_REBIND
- NX_DHCPV6_STATE_SENDING_DECLINE
- NX_DHCPV6_STATE_SENDING_INFORM_REQUEST
- NX_DHCPV6_STATE_BOUND_TO_ADDRESS

IPv6 アドレスの重複アドレス検出

NetX Duo の [*Duplicate Address Detection*] プロパティによってデフォルトで有効になっている重複アドレス検出（DAD）プロトコルが構成されている場合、NetX Duo は自動的に「近隣要請」メッセージを送信して、割り当てられたアドレスがネットワークで一意かどうかを検証します。IPv6 アドレスが一意であれば、割り当てられたアドレスが内部で NX_IPV6_ADDR_STATE_TENTATIVE から NX_IPV6_ADDR_STATE_VALID に昇格したとき、NetX Duo は DHCPv6 クライアントに通知します。アプリケーションでは、DAD が処理を完了するまでの時間として約 4 ～ 5 秒を見込んでおく必要があります。DAD が有効になっていない場合、IP インスタンスはただちにアドレスを有効とマークします。

IPv6 アドレスが有効になると、デバイスはその IPv6 アドレスを使用して IPv6 メッセージを送信できます。

DAD プロトコルが失敗すると、NetX Duo は DHCPv6 クライアントに対して、DECLINE メッセージをサーバーに送信し DHCPv6 クライアントを INIT 状態で再起動するように通知します。

DHCPv6 クライアント要請の再送

DHCPv6 クライアントでは、サーバーの返信を待ってタイムアウトすると別の DHCPv6 メッセージが送信されます。デフォルトでは、RFC 3315 の推奨に従って最初の再送までのインターバルは 1 秒です。SOLICIT メッセージに対する有効なサーバーレスポンスを DHCPv6 クライアントが受信できない場合、それ以降は再送信ごとにインターバルが 2 倍になり、デフォルトでは最大で 120 秒まで延長されます。

DHCPv6 リースのタイムアウト

サーバーから割り当てられた IPv6 リースには、DHCPv6 クライアントの非一時アドレス用アイデンティティアソシエーション（IANA）に 2 つのタイムアウトパラメータ（T1 および T2）が含まれています。IANA は、DHCPv6 で IPv6 アドレスデータを格納するためのデータ型として RFC 3315 で規定されています。

割り当てられた IPv6 アドレスの経過時間が T1 に達すると、DHCPv6 クライアントは自動的に RENEW メッセージを送信します。更新されないまま経過時間が T2 に達すると、DHCPv6 クライアントは自動的に REBIND メッセージを送信します。それでもレスポンスがない場合、DHCP クライアントは IP インスタンスへの IPv6 アドレスの登録を解除し、DHCPv6 プロトコルを INIT 状態で再起動します。DHCPv6 プロセスでは、これ以外の 2 つの IPv6 リースパラメータとして、推奨期間および有効期間が IANA 内のアイデンティティアソシエーション（IA）に自動的に割り当てられます。推奨期間および有効期間が期限切れになると、割り当てられている IPv6 アドレスは非推奨となるか無効と見なされます。そのため、T1 は推奨期間よりも短く、T2 は有効期間よりも短くする必要があります。

NetX Duo DHCP IPv6 クライアントモジュールの動作に関する重要な注意事項と制限事項

- NetX Duo ソース要素には、DHCPv6 をサポートするうえで重要なプロパティとして [*NetX Duo IPv6 Support*] プロパティ、[*Checksum computation support on received ICMPV6 packets*]、[*Checksum computation support on transmitted ICMPv6 packets*] があります。2 番目と 3 番目は受信パケットと送信パケットの ICMPv6 ヘッダーに ICMPv6 チェックサムが確実に含まれるようにするためのもので、DHCPv6 クライアントモジュールでは自動的に有効になります。NetX Duo ソース要素をプロジェクトに追加している場合は、これらのプロパティが有効であることを確認してください。
- DHCPv6 クライアントを作成するには、事前にパケットプールが作成されていることが必要です。アプリケーションは、IP インスタンスが使用する IP デフォルトパケットプール（g_packet_pool0）を使用するか、独自のパケットプール（通常は g_packet_pool1.）を作成することができます。
- クライアントは、SOLICIT 要求を送信する前に、ネットワーク上のクライアントを一意に定義するための DHCP 固有識別子（DUID）を作成する必要があります。通常は MAC アドレスが使用されますが、別の一意の識別子も使用できます。このサービスの一般的な呼び出し方法は次のとおりです。

```
nx_dhcpv6_create_client_duid(&g_dhcpv6_client0,
 NX_DHCPV6_DUID_TYPE_LINK_TIME /* Use MAC address */,
              NX_DHCPV6_HW_TYPE_IEEE_802,
              0 /* Client DUID time, usually set to zero */);
```

- DHCPv6 クライアントは、クライアント用に IANA も作成する必要があります。この構造体には、IPv6 アドレスや T1 時間、T2 時間などの IPv6 リース情報が含まれます（クライアントは IANA を使用してリース時間を要求できます）。IANA を作成するには、次のように nx_dhcpv6_client_create_iana サービスを使用します。

```
status = nx_dhcpv6_create_client_iana(&g_dhcpv6_client0,
          DHCPV6_IANA_ID /* ULONG unique ID */,
          DHCPV6_T1 /* Request T1 time in seconds or
                       set to TX__WAIT_FOREVER */,
          DHCPV6_T2 /* Request T2 time;must be longer
                     than T1 */);
```

- クライアントはnx_dhcpv6_request_solicit.をコールする前にnx_dhcpv6_add_client_iaサービスをコールすることによって、SOLICIT 要求で特定の IPv6 アドレスの割り当てをサーバーに要求できます。このサービスでは、IPv6 アドレスに NXD_ADDRESS アドレスデータ型が使用されます。NetX Duo でのデータ型定義の詳細については、『*NetX Duo User Guide for the Renesas Synergy ™ Platform*』を参照してください。
- DNS サーバー、NTP サーバー、その他のオプションなどのネットワーク情報を要求するには、アプリケーションで以下の API をすべてコールしてから nx_dhcpv6_request_solicit: をコールします。
  - nx_dhcpv6_request_option_timezone(&g_dhcpv6_client, NX_TRUE);
  - nx_dhcpv6_request_option_DNS_server(&g_dhcpv6_client, NX_TRUE);
  - nx_dhcpv6_request_option_time_server(&g_dhcpv6_client, NX_TRUE);
  - nx_dhcpv6_request_option_domain_name(&g_dhcpv6_client, NX_TRUE);
- クライアントは、割り当てられた IPv6 アドレスを解放することが必要になると、nx_dhcpv6_request_release サービスをコールすることによって DHCPv6 サーバーに通知します。DHCPv6 クライアントはユニキャストの RELEASE メッセージをサーバーに送信し、サーバーの REPLY を待つ必要があります。
- DHCPv6 クライアントに関する情報を取得する DHCPv6 クライアントサービスでは、アドレスインデックスを指定する必要があることがあります。ほとんどのクライアントでは、割り当てられている IPv6 グローバルアドレスは 1 つなのでアドレスインデックスは 0 です。
- リース時間に関する具体的な情報を取得するには、nx_dhcpv6_get_valid_ip_address_lease_time サービスを使用します。その場合、アドレスインデックス（通常は 0）の入力が必要です。
- NetX Duo DHCPv6 クライアントでは、ユニキャスト DHCPv6 メッセージを DHCPv6 サーバーに送信する際、サーバーのユニキャストオプションが許可されていることがサーバーによって示されていても、このオプションはサポートされません。
- NetX Duo DHCPv6 クライアントでサポートされる DUID は、LINK（MAC アドレス）と LINK TIME（MAC アドレスおよび時刻入力）のみです。
- NetX Duo DHCPv6 クライアントでは、サーバーがネットワーク上のクライアントに対して IPv6 アドレスの変更を開始する再構成要求はサポートされません。
- NetX Duo DHCPv6 クライアントでは、DHCPv6 固有識別子制御ブロックのエンタープライズ形式はサポートされません。Link Layer 形式と Link Layer Plus Time 形式のみがサポートされます。
- NetX Duo DHCPv6 クライアントでは、一時アソシエーション（TA）アドレス要求はサポートされませんが、非一時（IANA）オプション要求はサポートされます。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.15.4 アプリケーションへの NetX Duo DHCP IPv6 クライアントモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX Duo DHCP IPv6 クライアントモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユー

ザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX Duo DHCP IPv6 クライアントモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX Duo DHCP IPv6 クライアントモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_dhcp_client0 NetX Duo DHCP IPv6 Client	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo DHPC IPv6 Client

次の図に示すように、NetX Duo DHCP IPv6 クライアントモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

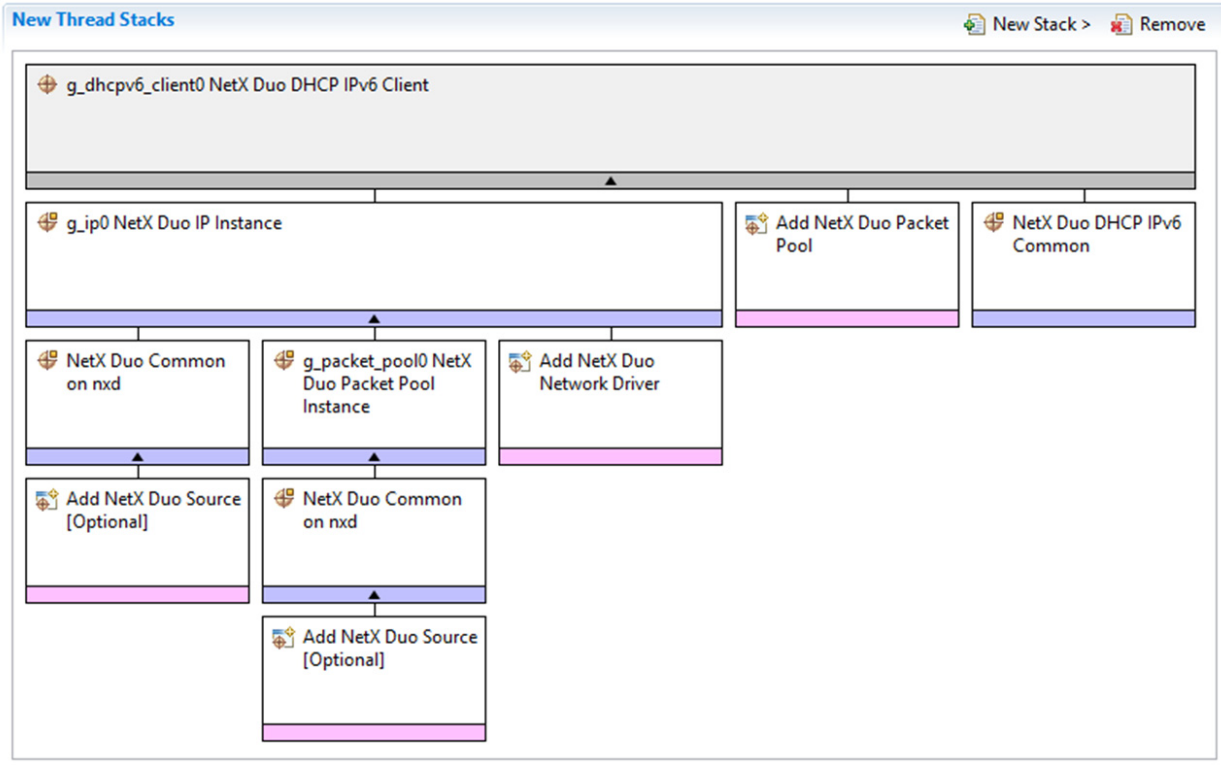


図 342:NetX Duo DHCP IPv6 クライアントモジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.15.5 NetX Duo DHCP IPv6 クライアントモジュールの構成

ユーザーは必要な動作に合わせて NetX Duo DHCP IPv6 クライアントモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX Duo DHCP IPv6 クライアントモジュールの構成設定

ISDE Property	Value	Description
Internal Thread Priority	3	Internal thread priority selection
Time out for obtaining DHCPv6 client mutex (ticks)	TX_WAIT_FOREVER	Time out for obtaining DHCPv6 client mutex selection
Time interval between current IP address lease time update (seconds)	1	Time interval between current IP address lease time update selection
Maximum IA addresses allowed in client record	1	Maximum IA addresses allowed in client record selection
Number of DNS servers the client will store	2	Number of DNS servers the client will stored selection
Number of time servers the client will store	1	Number of time servers the client will store selection
Domain name buffer size (bytes)	32	Domain name buffer size selection
Current time zone information buffer size (bytes)	16	Current time zone information buffer size selection
Maximum DHCPv6 server messages buffer size (bytes)	100	Maximum DHCPv6 server messages buffer size selection
Name	g_dhcpv6_client0	Module name
Internal thread stack size (bytes)	4096	Internal thread stack size selection
Name of state change notification function	dhcpv6_state_change_notify	Name of state change notification function selection

ISDE Property	Value	Description
Name of server error handler	dhcpv6_server_error_handler	Name of server error handler selection
Name of generated initialization function	nx_dhcpv6_client_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

スタックモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、異なるイーサネットインタフェースピンやリセットの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX Duo DHCP IPv6 クライアントのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	0,0,0,0	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection

ISDE Property	Value	Description
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo DHCP IPv6 共通インスタンスの構成設定

ISDE Property	Value	Description
Type of Service for UDP requests	Normal, Minimum delay, Maximum data, Maximum reliabililty, Minimum cost Default: Normal	Type of service UDP requests selection
Time to live	128	Time to live selection
Packet Queue depth	5	Packet queue depth selection
packet alocation timeout (seconds)	3	Packet allocation timeout selection
Interval for active session time update (seconds)	3	Interval for active session time update selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo DHCP IPv6 クライアントモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX Duo DHCP IPv6 クライアントモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I^2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.15.6 アプリケーションでの NetX Duo DHCP IPv6 クライアントモジュールの使用

次の例は、実際に使用できる有効な IP、ARP、UDP が確立済みで、リンクが実行されているシステムを前提にしています。

一般的なアプリケーションで NetX Duo DHCP IPv6 クライアントモジュールを使用する際の手順は次のとおりです。

1) nx_dhcpv6_create_client_duid. を使用して、DHCPv6 クライアントのクライアント DUID を作成します。
2) nx_dhcpv6_create_client_iana. を使用して、DHCPv6 クライアントの IANA を作成します。

3) nx_dhcpv6_request_option_DNS_server および nx_dhcpv6_request_option_time_server API を使用して、ネットワークオプション（DNS サーバーやタイムサーバーなど）を要求します（オプション）。
4) nx_dhcpv6_start API で DHCPv6 クライアントを起動します。これにより、DHCPv6 クライアントの状態が設定され、クライアントソケットがバインドされて、DHCPv6 クライアントがサーバーとメッセージを交換できるようになります。
5) nx_dhcpv6_request_solicit API を使用して、SOLICIT メッセージをサーバーに送信します。DHCPv6 プロトコルの他の部分は、DHCPv6 クライアントが内部で管理します。
6) nx_dhcpv6_get_valid_ip_address_count API を使用して、address_count > 0 が返されることを確認して IPv6 アドレス解決をチェックします。これで、DHCPv6 クライアントが有効な IPv6 アドレスになります。

次の図は、一般的な手順を示した通常の動作フロー図です。

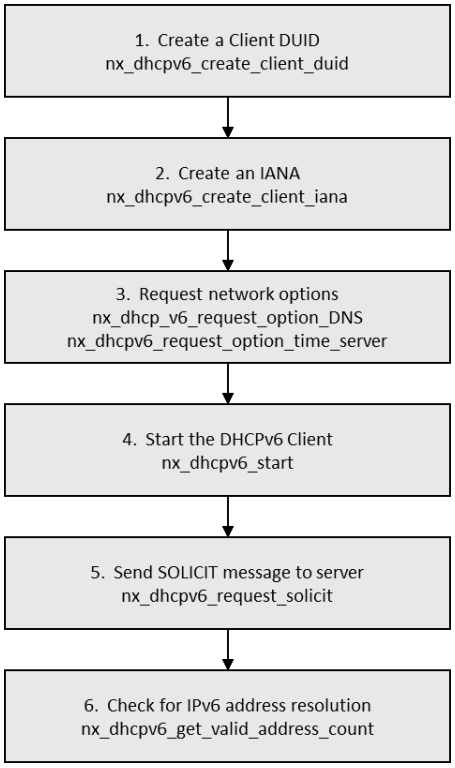


図 343:一般的な NetX Duo DHCP IPv6 クライアントモジュールアプリケーションのフロー図

4.3.16 NetX Duo DHCP IPv6 サーバー

動的ホスト構成プロトコル（DHCP）は、IP アドレスとネットワークパラメータを取得するために使用されます。DHCP は、(静的なアドレス構成に限定されている) BOOTP の基本的な機能を拡張し、クライアントに IP アドレスを一定時間「リースする」ことによって完全に動的な IP アドレス割り当てを実現するように設計されています。DHCP は、BOOTP のように静的な IP アドレス割り当てを行うように構成することもできます。アプリケーションの IP ア

ドレスは、NetX ™コンポーネントに提供されるパラメータの 1 つです。静的に、またはユーザー構成を通じて割り当てられた IP アドレスがアプリケーションにとって既知である場合、IP アドレスを提供することで問題が発生することはありません。アプリケーションが割り当てられる IP アドレスを認識しない場合や、認識する必要がない場合、IP アドレスをゼロとして NetX を初期化します。これにより、NetX に追加された DHCP クライアントコンポーネントは IP アドレスを動的に取得できます。

IPv6 ネットワークでは、DHCP（IPv4 に限定）の代わりに DHCPv6 を使用することで、DHCPv6 サーバーからグローバル IPv6 アドレスの動的な割り当てを行います。DHCPv6 の機能は大部分が同じですが、多くの機能強化が行われています。NetX Duo ™ DHCPv6 サーバーAPI を DHCPv6 クライアントの IPv6 アドレス要求に利用する方法は DHCPv6 で詳しく説明されています。

4.3.16.1 NetX Duo DHCP IPv6 サーバーモジュールの特長

- NetX Duo DHCPv6 サーバーは、RFC 3315、RFC 3646、および関連する RFC に準拠しています。
- また、以下を行うためのハイレベル API を提供します。
  - DHCPv6 サーバーインスタンスの作成と削除
  - DHCPv6 サーバースレッドタスクの開始と停止
  - リースする IPv6 アドレスのプールの作成
  - 要求を行うクライアントに割り当て可能な DHCPv6 リースのテーブルの維持

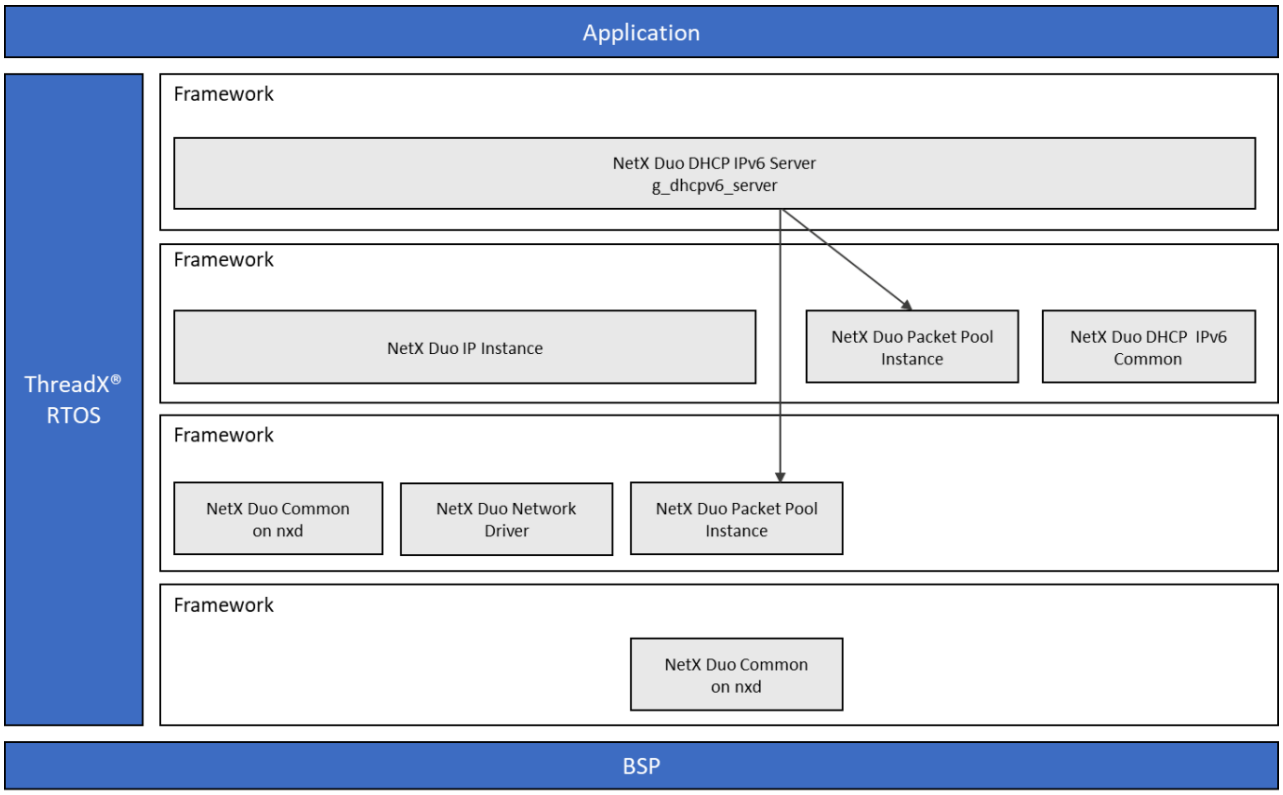


図 344:NetX Duo DHCP IPv6 サーバーモジュールのブロック図

注 : 上の図で、NetX Duo ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.16.2 NetX Duo DHCP IPv6 サーバーモジュールの API の概要

NetX Duo DHCPv6 サーバーモジュールでは、サーバー情報の作成、削除、追加、取得のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX Duo DHCP IPv6 サーバーモジュールの API の要約

Function Name	Example API Call and Description
nx_dhcpv6_server_create	nx_dhcpv6_server_create(&g_dhcpv6_server0, &server_ip_address, "DHCPv6 Server", &g_packet_pool0, stack_pointer, NX_DHCPV6_SERVER_THREAD_STACK_SIZE, address_declined_handler, option_request_handler); Create a DHCPv6 Server instance.
nx_dhcpv6_server_delete	nx_dhcpv6_client_delete(&g_dhcpv6_server0); Delete a DHCPv6 Server instance and release resources (unbind port, delete socket, timers and thread).
nx_dhcpv6_create_ip_address_lease	nx_dhcpv6_create_ip_address_range(&g_dhcpv6_server0, &start_ipv6_address, &end_ipv6_address, &addresses_added); Create the Server's IPv6 address lease pool.
nx_dhcpv6_reserve_ip_address_range	nx_dhcpv6_reserve_ip_address_range(&g_dhcpv6_server0, &start_ipv6_address, &end_ipv6_address, &addresses_reserved); Reserve the specified range of IPv6 addresses not to be leased out to a requesting Client.

Function Name	Example API Call and Description
nx_dhcpv6_add_ip_address_lease	nx_dhcpv6_add_ip_address_lease(&g_dhcpv6_server0, table_index, &lease_IP_address, T1, T2, valid_lifetime, preferred_lifetime); Copy an IPv6 lease record into the specified index into the Server table. Intended for use in non-volatile storage of IPv6 lease data.
nx_dhcpv6_add_client_record	nx_dhcpv6_add_client_record(&g_dhcpv6_server0, table_index, message_xid, &client_address, client_state, IP_lease_time_accrued, valid_lifetime, duid_type, duid_hardware, physical_address_msw, physical_address_lsw, duid_time, duid_vendor_number, duid_vendor_private, duid_private_length); Copy a Client record into the specified index into the Server table. Intended for use in non-volatile storage of Client IPv6 address data.
nx_dhcpv6_create_dns_address	nx_dhcpv6_create_dns_address(&g_dhcpv6_server0, &dns_ipv6_address); Set the DNS server address to include in network parameters sent to clients.
nx_dhcpv6_retrieve_client_record	nx_dhcpv6_retrieve_client_record(&g_dhcpv6_server0, table_index, message_xid, &client_address, client_state, IP_lease_time_accrued, valid_lifetime, duid_type, duid_hardware, physical_address_msw, physical_address_lsw, duid_time, duid_vendor_number, duid_vendor_private, duid_private_length); Retrieve items from the Client specified by the index into the Server table. Intended for use in non-volatile storage of Client IPv6 address data.

Function Name	Example API Call and Description
nx_dhcpv6_retrieve_ip_address_lease	nx_dhcpv6_retrieve_ip_address_lease(&g_dhcpv6_server0, table_index, &lease_IP_address, T1, T2, valid_lifetime, preferred_lifetime); Retrieve items from the IPv6 lease specified by the index into the Server table. Intended for use in non-volatile storage of IPv6 lease data.
nx_dhcpv6_server_interface_set	nx_dhcpv6_server_interface_set(&g_dhcpv6_server0, 0, 1); Set the interface the DHCPv6 Server will run on, and the global address the DHCPv6 Server will use in messages to Clients. By default, the DHCPv6 Server runs on the primary interface (index 0).
nx_dhcpv6_set_server_duid	nx_dhcpv6_set_server_duid(&g_dhcpv6_server0, NX_DHCPV6_SERVER_DUID_TYPE, NX_DHCPV6_SERVER_HW_TYPE, physical_address_msw, physical_address_lsw, duid_time); Create the Server DUID which is a required part of the DHCPv6 header and uniquely identifies the DHCPv6 Server.
nx_dhcpv6_server_start	nx_dhcpv6_server_start(&g_dhcpv6_server0); Start the DHCPv6 thread task.
nx_dhcpv6_server_suspend	nx_dhcpv6_server_suspend(&g_dhcpv6_server0); Suspend the DHCPv6 server task.
nx_dhcpv6_server_resume	nx_dhcpv6_server_resume(&g_dhcpv6_server0); Resume the DHCPv6 server task.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

ステータス戻り値

Name	Description
NX_SUCCESS	Successful API call.
NX_PTR_ERROR*	Invalid pointer input.
NX_CALLER_ERROR*	Must be called from thread.
NX_DHCPV6_PARAM_ERROR	Invalid non pointer input.
NX_INVALID_INTERFACE	Invalid interface index input.
NX_DHCPV6_UNSUPPORTED_DUID_TYPE	DUID type unknown or not supported.
NX_DHCPV6_UNSUPPORTED_DUID_HW_TYPE	DUID hardware type unknown or not supported.
NX_DHCPV6_IA_ADDRESS_ALREADY_EXIST	Duplicate IA address.
NX_DHCPV6_REACHED_MAX_IA_ADDRESS	IA exceeds the max IAs Client can store.
NX_DHCPV6_INVALID_IA_ADDRESS	Invalid (e.g. null) IA address in IA.
NX_DHCPV6_IA_ADDRESS_NOT_VALID	IPv6 address successfully assigned.
NX_DHCPV6_UNKNOWN_OPTION	Unknown/unsupported option code.
NX_DHCPV6_ALREADY_STARTED	DHCPv6 Client is already running.
NX_DHCPV6_NOT_STARTED	DHCPv6 Client task not started.
NX_DHCPV6_MISSING_REQUIRED_OPTIONS	Client missing required options.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

注 : これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。
NetX と NetX Duo でのエラーチェックサービスの詳細については、それぞれ『NetX User Guide for the Renesas Synergy ™ Platform』または『NetX Duo User's Guide for the Renesas Synergy ™ Platform』を参照してください。

4.3.16.3 NetX Duo DHCP IPv6 サーバーモジュールの動作の概要

NetX Duo DHCPv6 サーバーモジュールは、サーバー用の NetX Duo IP インスタンス、およびクライアント要求を受信待ちする既知の DHCPv6 サーバーのポート 547 にバインドされた UDP ソケットを作成します。DHCPv6 を開始する前に、サーバーにグローバル IPv6 アドレスが必要なので、IP NetX Duo の [Instance] プロパティボックスで [IPv6 Global Address] プロパティを設定します（IPv4 アドレスは 32 ビット長ですが、これは 128 ビット長のアドレスです）。

DHCPv6 サーバーは、nx_ip_status_check サービスを使用して IPv4 アドレスが検証されるのを待つ必要があります。これは、その IP アドレスをサーバーが DHCPv6 メッセージに使用しない場合でも同様です。ドライバーを IP レイヤーからの情報に基づいて初期化し、リンクを有効にする必要があります。これらはすべて IPv4 アドレスの登録時に行われます。

DHCPv6 サーバーを起動する前に、アプリケーションは nx_dhcpv6_create_ip_address_range サービスを使用して、割り当て可能な IPv6 アドレスのプールを作成する必要があります。また、nx_dhcp_set_server_duid サービスを使用してサーバーの DUID（DHCP 固有識別子。通常は MAC アドレスに基づき、すべての DHCPv6 サーバーメッセージで必要）も作成する必要があります。オプションで、nx_dhcpv6_create_dns_server サービスを使用してクライアントに対して DHCPv6 オプションデータにネットワーク DNS サーバーを含めるよう設定できます。これで、nx_dhcpv6_server_start サービスによってサーバーを起動できます。

注 : テキストで参照しているプロパティはすべて、特に説明がない限り NetX Duo DHCP IPv6 サーバーのプロパティボックスにあります。

サーバーでは、IPv6 アドレスのテーブルが維持され、アドレスの状態が更新され、どのアドレスがリース中であるかが示されます。リース中のアドレスは、リース先のクライアントがテーブルに示されます。このテーブルのサイズは [Maximum Size of the Server's IP lease table] プロパティで設定され、IPv6 アドレスプール内の IPv6 アドレスの数以上にする必要があります。サーバーでは、クライアントレコードのために別のテーブルも維持されます。このテーブルのサイズは DHCP サーバーのプロパティボックスの [Size of Server's Client record table] プロパティで設定され、IPv6 リーステーブルのサイズ以上にする必要があります。サーバーは Client Release または Decline を受信すると、それに応じて IPv6 リーステーブルおよびクライアントレコードテーブルを更新します。

DHCPv6 サーバーにより、2 つのタイマが作成されます。1 つはクライアントにリースされている IPv6 アドレスの残り時間を追跡するタイマです。サーバーがクライアントへのリースを確認するインターバルは [Client lease time expiration check interval] プロパティで設定され、デフォルトは 60 秒です。サーバーが有効期限を極端に短くしてリースを発行する場合、その値をタイムアウト値の約 10 ～ 20% にする必要があります。リースの有効期限が切れると、リースは自動的に IPv6 アドレスのプールに戻され、クライアントレコードが削除されます。もう 1 つのタイマはクライアントセッションの非アクティブ状態を監視するために使用されます。セッション非アクティブタイムアウトのデフォルトは 20 秒です。サーバーがクライアントレコードに対してセッション非アクティブタイムアウトの期限切れを確認するインターバルは、NetX Duo の [DHCP IPv6 Common] プロパティボックスの [Interval for active session time update] プロパティで設定され、デフォルトのインターバルは 3 秒です。

NetX Duo DHCP IPv6 サーバーモジュールの動作に関する重要な注意事項と制限事項

- NetX Duo では IPv6 と ICMPv6 を有効にする必要があります。NetX Duo の [Source] プロパティボックスで [NetX Duo IPv6 Support] プロパティが有効になっていることを確認します。この設定により、ICMPv6 が自動的に有効になります。

デバイスで ICMPv6 チェックサムの計算がサポートされている場合、以下の値は無効のままで構いません。

- デフォルトでは、ICMPv6 チェックサムは無効です。このチェックサムを有効にするには（ハードウェアで ICMPv6 チェックサムの計算がサポートされていない場合）、NetX Duo の [Source] プロパティボックスの [Checksum computation support on transmitted ICMPv6 packets] プロパティを有効にします。[Checksum computation support on received ICMPv6 packets] プロパティも有効にする必要があります。
- サーバーのグローバル IPv6 アドレスが一意であることを検証するために、重複アドレス検出（DAD）が推奨されます。このプロトコルは IPv4 アドレスの一意性を確認するために IPv4 で空の ARP プローブを送信することと似ていますが、IPv6 アドレスに対してのみ使用できます。DAD（デフォルトでは無効）を有効にするには、NetX Duo の [Source] プロパティボックスで [Duplicate Address Detection support] プロパティを有効に設定します。DAD のために送信される要請パケットの数は、[Neighbor Solicitation message count before interface address marked valid] プロパティで設定され、デフォルトは 3 です（約 1 秒間隔で送信されます）。この構成の場合、アプリケーションスレッドは DAD プロトコルが完了するまで約 4 秒待つ必要があります。

- DHCPv6 サーバーからクライアントへのレスポンスでは以下の DHCPv6 パラメータが提供されます。
  - T1 時間：クライアントが IPv6 アドレスリースの更新を開始するまでの時間。[Server interval for first client IP address renewal attempt] プロパティで設定します。
  - 推奨時間：クライアントの IPv6 アドレスを非推奨にするまでの時間。[Time interval after which the client IP is deprecated] プロパティで設定します。これは、RFC 3315 の推奨に基づいて T1 時間の 2 倍にする必要があります。
  - T2 時間：IPv6 アドレスの更新が失敗した場合にクライアントが再バインドを開始するまでの時間。[Server interval for the second client IP address renewal attempt] プロパティで設定します。
  - 有効期間：クライアントの IPv6 アドレスを期限切れにして、クライアントが使用できないようにするまでの時間。これは、[Time interval after which leased IP is invalid] プロパティで設定します。RFC 3315 の推奨に基づいて推奨時間の 2 倍にする必要があります。
  - IPv6 リース時間（有効期間）に上限はありませんが、これら 4 つの時間パラメータ相互のインターバルは、DHCPv6 プロトコルの更新（および必要な場合は再バインド状態）に関する論理的な順番が守られるものでなければなりません。
- Client Decline メッセージを処理するためのアドレス拒否ハンドラコールバックは、現在の NetX Duo DHCPv6 サーバーには実装されていません。このコールバックは、アドレス拒否イベントをサーバーがアプリケーションに通知する際に使用するように、RFC 3315 の DHCPv6 仕様で推奨されています。
- 高速コミットオプション：DHCPv6 アドレス要求プロセスを Solicit および Reply のメッセージ交換だけに最適化します。
- 再構成オプション：サーバーがクライアントの IP アドレスのステータスを変更することを許可します。
- ユニキャストオプション：すべてのクライアントメッセージは、DHCPv6 サーバーに直接送信するのではなく、All_DHCP_Relay_Agents_and_Servers マルチキャストアドレスに送信する必要があります。
- 一時アドレス用アイデンティティアソシエーション（IA_TA）オプション：クライアントに付与される一時的な IP アドレス。
- 複数 IA（IPv6 アドレス）オプション：クライアント要求ごと。
- DHCPv6 クライアントとサーバーの間でのホストのリレー：クライアントとサーバーは同じネットワーク上に存在する必要があります。
- NetX Duo DHCPv6 サーバー：DNS サーバーオプション要求のみを直接サポートします。
- プレフィックス委譲オプション：サポートされません。
- オプション要求コールバック：アプリケーションで使用することを目的とし、サポートされる DHCPv6 オプション、およびクライアントへのレスポンスで DHCPv6 サーバーに提供する情報を決定します。ただし、この情報を DHCPv6 サーバーレスポンスに反映する処理は実装されていません。このコールバックは、DHCPv6 サーバーメッセージには影響しません。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.16.4 アプリケーションへの NetX Duo DHCP IPv6 サーバーモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX Duo DHCP IPv6 サーバーモジュールのいずれか、または両方を組み込む方法について説明します。

注：このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX Duo DHCP IPv6 サーバーモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX Duo DHCP IPv6 サーバーモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_dhcp_server0 NetX Duo DHCP IPv6 Server	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo DHPC IPv6 Server

次の図に示すように、NetX Duo DHCP IPv6 サーバーモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

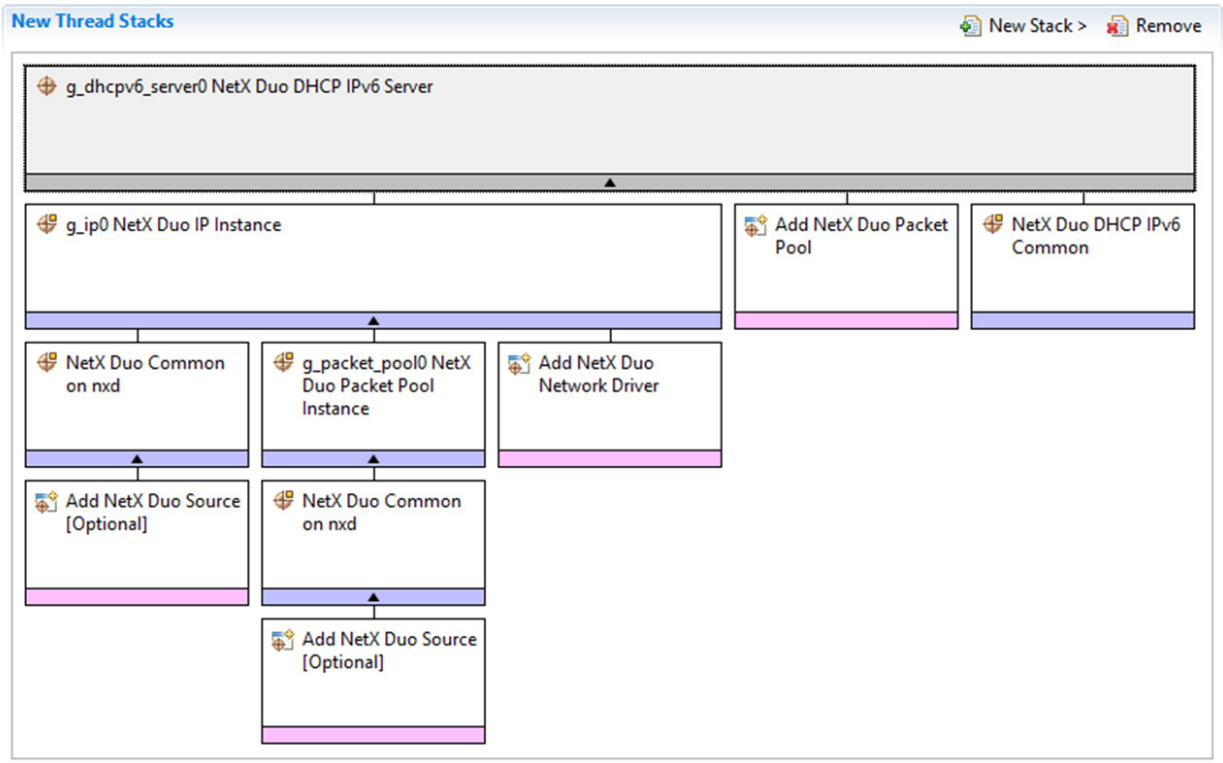


図 345:NetX Duo DHCP IPv6 サーバーモジュールのスタック

上記のスタックで、NetX ネットワークドライバー (NetX Duo スタックの場合は NetX Duo ネットワークドライバー) はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.16.5 NetX Duo DHCP IPv6 サーバーモジュールの構成

ユーザーは必要な動作に合わせて NetX Duo DHCP IPv6 サーバーモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX Duo DHCP IPv6 サーバーモジュールの構成設定

ISDE Property	Value	Description
Internal thread priority	1	Internal thread priority selection
Client lease time expiration check interval (seconds)	60	Client lease time expiration check interval selection
DHCPv6 packet receive timeout (seconds)	1	DHCPv6 packet receive timeout selection
Server preference ranking for clients	0	Server preference rankng for clients selection
Maximum options to extract from a client message	6	Maximum options to extract from a client message selection
Server interval for first client IP address renewal attempt (seconds)	2000	Server interval for first client IP address renewal attempt selection
Server interval for second client IP address renewal attempt (seconds)	3000	Server interval for second client IP address renewal attempt selection

ISDE Property	Value	Description
Time interval after which client IP is deprecated (seconds)	2*NX_DHCPV6_DEFAULT_T1_TIME	Time interval after which client IP is deprecated selection
Time interval after which leased IP in invalid (seconds)	2*NX_DHCPV6_DEFAULT_PREFERRED_TIME	Time interval after which leased IP in invalid selection
Maximum server status option message size (bytes)	100	Maximum server status option message size selection
Maximum Size of the Server's IP lease table (count)	100	Maximum Size of the Server's IP lease table selection
Size of the Server's Client record table (count)	120	Size of the Server's Client record table selection
Server socket fragmentation option	Don't fragment, fragment okay Default: Don't fragment	Server socket fragmentation option selection
Vendor assigned unique ID	abcdefghijklmnopqrstuvwxyz	Vendor assigned unique ID selection
Private vendor ID	0x12345678	Private vendor ID selection
Size of Vendor ID buffer (bytes)	48	Size of vendor ID buffer selection
Client request success message: granted	IA OPTION GRANTED	Client request successmessage: granted selection
Client request failure message: Failure unspecified	IA OPTION NOT GRANTED - FAILURE UNSPECIFIED	Client request failure message: Failure unspecified selection
Client request failure message: No addresses available	IA OPTION NOT GRANTED - NO ADDRESSES AVAILABLE	Client request failure message: No addresses available selection
Client request failure message: Invalid client request	IA OPTION NOT GRANTED - INVALID CLIENT REQUEST	Client request failure message: Invalid client request selection

ISDE Property	Value	Description
Client request failure message: Client not on link	IA OPTION NOT GRANTED - CLIENT NOT ON LINK	Client request failure message: Client not on link selection
Client request failure message: Client must use multicast	IA OPTION NOT GRANTED - CLIENT MUST USE MULTICAST	Client request failure message: Client must use multicast selection
Session inactivity timeout (seconds)	20	Session inactivity timeout selection
Name	g_dhcpv6_server0	Module name
Internal thread stack size (bytes)	4096	Internal thread stack size selection
Name of address declined handler function	dhcpv6_address_declined_ handler	Name of address declined handler function selection
Name of option request handler	dhcpv6_option_request_han dler	Name of option request handler selection
Name of generated initialization function	dhcpv6_server_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

スタックモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、異なるイーサネットインタフェースピンやリセットの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX Duo DHCP IPv6 サーバーのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	0,0,0,0	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection

ISDE Property	Value	Description
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo DHCP IPv6 共通インスタンスの構成設定

ISDE Property	Value	Description
Type of Service for UDP requests	Normal, Minimum delay, Maximum data, Maximum reliabililty, Minimum cost Default: Normal	Type of service UDP requests selection
Time to live	128	Time to live selection
Packet Queue depth	5	Packet queue depth selection
packet alocation timeout (seconds)	3	Packet allocation timeout selection
Interval for active session time update (seconds)	3	Interval for active session time update selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo DHCP IPv6 サーバーモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX Duo DHCP IPv6 サーバーモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I^2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.16.6 アプリケーションでの NetX Duo DHCP IPv6 サーバーモジュールの使用

次の例は、実際に使用できる有効な IP、ARP、UDP が確立済みで、リンクが実行されているシステムを前提にしています。

一般的なアプリケーションで NetX Duo DHCP IPv6 サーバーモジュールを使用する際の手順は次のとおりです。

1) nx_dhcpv6_set_server_duid API を使用して、DHCPv6 サーバーの DUID を作成します。
2) nx_dhcpv6_create_ip_address_range. を使用して、DHCPv6 サーバー用の割り当て可能な IPv6 アドレスのプールを作成します。
3) nx_dhcpv6_create_dns_address API を使用して、IPv6 の DNS サーバーオプションを設定します（オプション）。
4) nx_dhcpv6_server_start API で DHCPv6 サーバーを起動します。
5) 必要に応じて、nx_dhcpv6_server_suspend API を使用して DHCPv6 サーバーをサスペンドできます。
6) nx_dhcpv6_server_resume API を使用して DHCPv6 サーバーを再開できます（オプション）。
7) nx_dhcpv6_server_delete API を使用して、DHCPv6 サーバーを削除します。

次の図は、一般的な手順を示した通常の動作フロー図です。

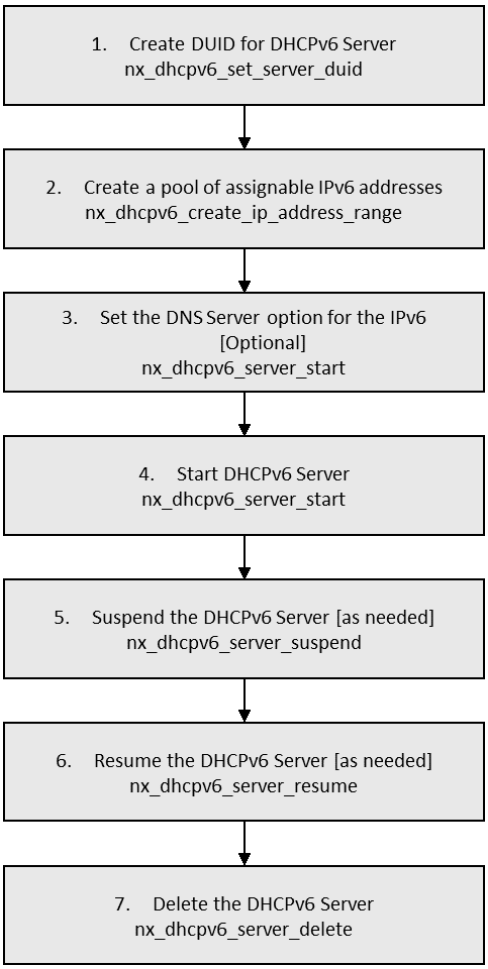


図 346:一般的な NetX Duo DHCP IPv6 サーバーモジュールアプリケーションのフロー図

4.3.17 NetX/NetX Duo DNS クライアント

ドメイン名システム（DNS）は、ドメイン名と物理 IP アドレスの間のマッピングを格納した分散データベースです。このデータベースが「分散」と呼ばれるのは、すべてのマッピングを含む単一のエンティティがインターネット上に存在するわけではないからです。マッピングの一部を保持する 1 つのエンティティを DNS サーバーと呼びます。インターネットは大量の DNS サーバーで構成されており、それぞれにデータベースのサブセットが格納されています。また DNS サーバーは、ドメイン名のマッピング情報を求める DNS クライアント要求に対し、要求されたマッピングを保持している場合は応答します。NetX および NetX Duo の DNS クライアントプロトコルでは、1 つまたは複数の DNS サーバーにマッピング情報を要求するサービスがアプリケーションに提供されます。

注 : 特に説明がない限り、NetX Duo DNS クライアントモジュールは DNS クエリの応用、設定、実行に関して NetX DNS クライアントモジュールと同じです。NetX Duo で IPv6 用に IP インスタンスを設定する方法については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。

4.3.17.1 NetX/NetX Duo DNS クライアントモジュールの特長

- DNS を運用するための独立したパケットプールの作成（オプション）
- タイプ A、AAAA、NS の DNS クエリのサポート
- CNAME、SRV、TXT、SOA、MX の DNS レコードタイプのサポート
- キャッシュした DNS データの格納、取得に使用する DNS キャッシュのサポート
- 以下を行うためのハイレベル API
  - 名前と IP アドレスのルックアップ
  - DNS サーバーの追加と削除
  - DNS インスタンスの作成と削除
- NetX DNS は以下の RFC に準拠しています
  - RFC 1034：ドメイン名 – 概念と機能
  - RFC 1035：ドメイン名 – 実装と仕様
  - RFC 1480：US ドメイン
  - RFC 2782 サービスの場所を指定するための DNS RR (DNS SRV)
  - RFC 3596：IP バージョン 6 をサポートするための DNS 機能拡張

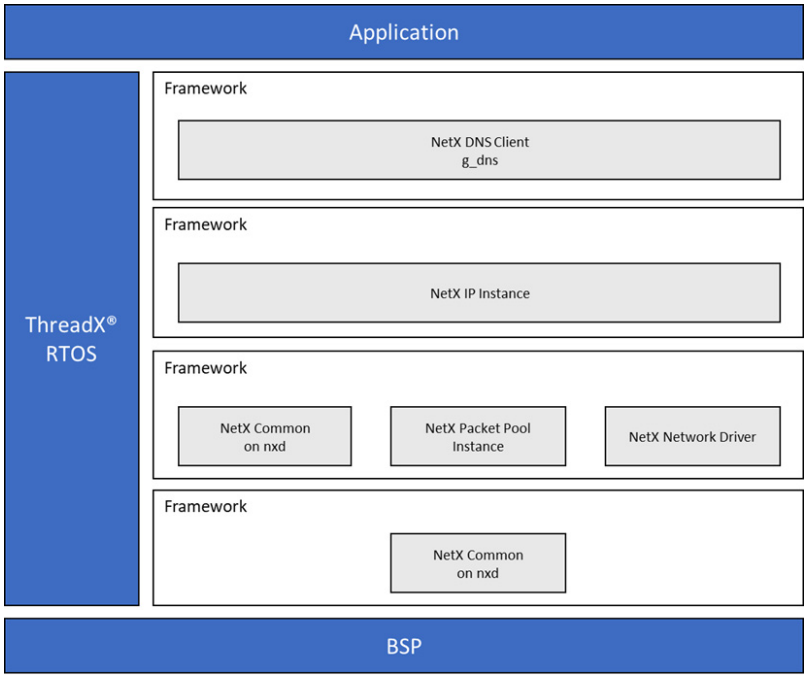


図 347:NetX/NetX Duo DNS クライアントモジュールのブロック図

注：上の図で、NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.17.2 NetX/NetX Duo DNS クライアントモジュールの API の概要

NetX/NetX Duo DNS クライアントモジュールでは、接続、バインド、受信待ち、送信、受信のための API が定義されています。次の表には、使用可能なすべての API 関数のリスト、API 関数コールの例、各 API 関数の簡単な説明が記載されています。

NetX/NetX Duo DNS クライアントモジュールの API の要約

Function Name	Example API Call and Description
nx_dns_create	nx_dns_create(&my_dns, &my_ip, "My DNS"); Create a DNS Client instance.
nx_dns_delete	nx_dns_delete(&my_dns); Delete a DNS Client instance.
nx_dns_packet_pool_set	nx_dns_packet_pool_set(&my_dns, &packet_pool); Set the DNS Client packet pool.
nx_dns_get_serverlist_size	nx_dns_get_serverlist_size (&my_dns, 5); Return the size of the DNS Client's Server list.
nx_dns_info_by_name_get	nx_dns_info_by_name_get(&my_dns, "www.abc1234.com", &ip_address, &port, 200); Return IPv4 address, port querying on input host name.
nx_dns_ipv4_address_by_name_get	nx_dns_ipv4_address_by_name_get(&client_dns,(UCHAR *)"www.my_example.com", record_buffer,sizeof(record_buffer),&record_count,500); Look up the IPv4 address for the specified host name.

Function Name	Example API Call and Description
nx_dns_host_by_address_get	nx_dns_host_by_address_get(&my_dns, IP_ADDRESS(192.2.2.10), &resolved_name[0], BUFFER_SIZE, 450); Look up a host name from a specified IP address (supports only IPv4 addresses).
nx_dns_host_by_name_get	nx_dns_host_by_name_get(&my_dns, "www.my_example.com", &ip_address, 4000); Look up an IP address from the host name (supports only IPv4 addresses).
nx_dns_server_add	nxd_dns_server_add(&my_dns, server_address); Add a DNS Server of the specified IP address to the Client's Server list (supports only IPv4).
nx_dns_server_get	nx_dns_server_get(&my_dns, 5, &my_server_address); Return the DNS Server in the Client list at the specified index (supports only IPv4 addresses).
nx_dns_server_remove	nx_dns_server_remove(&my_dns, IP_ADDRESS(202,2,2,13)); Remove a DNS Server from the Client list (supports only IPv4 addresses).
nx_dns_server_remove_all	nx_dns_server_remove_all(&my_dns); Remove all DNS Servers from the Client list.
nx_dns_cache_initialize	nx_dns_cache_initialize(&my_dns, &dns_cache, 2048); Initialize a DNS Cache.

Function Name	Example API Call and Description
nx_dns_cache_notify_clear	nx_dns_cache_notify_clear(&my_dns); Clear the DNS cache full notify function.
nx_dns_cache_notify_set	nx_dns_cache_notify_set(&my_dns, cache_full_notify_cb); Set the DNS cache full notify function.
nx_dns_cname_get	nx_dns_cname_get(&client_dns, (UCHAR *)"www.my_example.com ", record_buffer, sizeof(record_buffer), 500); Look up the canonical domain name for the input domain name alias.
nx_dns_authority_zone_start_get	nx_dns_authority_zone_start_get(&client_dns, (UCHAR *)"www.my_example.com", record _buffer, sizeof(record_buffer), &record_count, 500); Look up the start of a zone of authority associated with the specified host name.
nx_dns_domain_name_server_get	nx_dns_domain_name_server_get(&client_dns, (UCHAR *)" www.my_example.com ", record_buffer, sizeof(record_buffer), &record_count, 500); Look up the authoritative name servers for the input domain zone
nx_dns_domain_mail_exchange_get	nx_dns_domain_mail_exchange_get(&client_dns, (UCHAR *)" www.my_example.com",record_buffer, sizeof(record_buffer), &record_count, 500); Look up the mail exchange associated with the specified host name.

Function Name	Example API Call and Description
nx_dns_domain_service_get	nx_dns_domain_service_get(&client_dns, (UCHAR *)"www.my_example.com ", record_buffer, sizeof(record_buffer), &record_count, 500); Look up the service(s) associated with the specified host name.
nxd_dns_ipv6_address_by_name_get**	nxd_dns_ipv6_address_by_name_get(&client_dns, (UCHAR *)"www.my_example.com", record__buffer, sizeof(record_buffer), &record_count, 500); Look up the IPv6 address from the specified host name.
nxd_dns_host_by_address_get**	nxd_dns_host_by_address_get(&my_dns, &host_address, resolved_name, sizeof(resolved_name), 4000); Look up a host name from the input IP address (supports both IPv4 and IPv6 addresses).
nxd_dns_host_by_name_get**	nxd_dns_host_by_name_get(&my_dns, “www.my_example.com”, &ip_address, 4000, NX_IP_VERSION_V4); Look up an IP address from the input host name (supports both IPv4 and IPv6 addresses).
nxd_dns_server_add**	nxd_dns_server_add(&my_dns, &server_address); Add the input DNS Server to the Client server list (supports both IPv4 and IPv6 addresses).

Function Name	Example API Call and Description
nxd_dns_server_get**	nxd_dns_server_get(&my_dns, 5, &my_server_address); Return the DNS Server in the Client list at the specified index (supports both IPv4 and IPv6 addresses).
nxd_dns_server_remove**	nxd_dns_server_remove(&my_dns,& server_ADDRESS); Remove a DNS Server of the specified IP address from the Client list (supports both IPv4 and IPv6 addresses).

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :** この API は、NetX Duo DNS クライアントでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

ステータス戻り値

Name	Description
NX_SUCCESS	API Call Successful
NX_DNS_NO_SERVER	Client server list is empty
NX_DNS_QUERY_FAILED	No valid DNS response received
NX_DNS_NEED_MORE_RECORD_BUFFER	Input buffer size insufficient to hold the minimum data
NX_PTR_ERROR*	Invalid IP or DNS pointer
NX_CALLER_ERROR*	Invalid caller of this service
NX_DNS_ERROR	Internal error in DNS Client processing
NX_DNS_PARAM_ERROR	Invalid non-pointer input
NX_DNS_CACHE_ERROR	Invalid Cache pointer
NX_DNS_TIMEOUT	Timed out on obtaining DNS mutex

Name	Description
NX_DNS_BAD_ADDRESS_ERROR	Null input address
NX_DNS_INVALID_ADDRESS_TYPE	Index points to invalid address type (e.g. IPv6)
NX_DNS_IPV6_NOT_SUPPORTED	Cannot process record with IPv6 disabled
NX_NOT_ENABLED	Client not configured for this option
NX_DNS_DUPLICATE_ENTRY	DNS Server is already in the Client list
NX_NO_MORE_ENTRIES	No more DNS Servers allowed (list is full)
NX_DNS_SERVER_NOT_FOUND	Server not in client list

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

注 : これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX と NetX Duo でのエラーチェックサービスの詳細については、それぞれ『NetX User Guide for the Renesas Synergy ™ Platform』または『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

4.3.17.3 NetX/NetX Duo DNS クライアントモジュールの動作の概要

DNS メッセージ

NetX DNS クライアントモジュールは有効な IP アドレスを登録するほか、IP インスタンスを作成し、NetX で UDP サービスを有効化し、ネットワークドライバーを初期化します。また、ポート 53 で DNS クエリの受信待ちをしている DNS サーバーとの間で DNS メッセージを送受信するための UDP ソケットを作成します。

マッピングを取得するには、解決する名前または IP アドレスを含む DNS クエリメッセージを DNS クライアントが準備します。メッセージがサーバーリストの先頭の DNS サーバーに送信されます。そのサーバーにマッピングが存在する場合、サーバーは要求されたマッピングを含む DNS レスポンスメッセージを使用して DNS クライアントに返信します。サーバーが応答しない場合、DNS クライアントはリスト内の次のサーバーにクエリを送信し、すべての DNS サーバーにクエリを送信するまで繰り返します。どの DNS サーバーからもレスポンスが得られない場合、DNS クライアントはサーバーリストの先頭に戻って DNS メッセージを再送します。DNS クエリは再送タイムアウトが期限切れになるまで送信されます。レスポンスが受信されない限り、再送タイムアウトの時間は、送信タイムアウトに達するまでサーバーリストの処理を繰り返すたびに 2 倍になります。このタイムアウト値は [*Maximum duration to retransmit a DNS query (seconds)*] プロパティで設定されます。デフォルト値は 64 秒です。DNS クライアントがサーバーリストの処理を繰り返す回数の上限は [*Maximum retries for a server*] プロパティで設定され、デフォルト値は 3 です。

一般的な NetX DNS クライアントクエリは次のとおりです。

- nx_dns_host_by_name_get サービスを使用した、IPv4 アドレスのルックアップ（タイプ A）。
- nx_dns_host_by_address_get サービスを使用した、Web ホスト名を取得するための IP アドレスの逆引き（タイプ PTR のクエリ）。
- nxd_dns_host_by_name_get サービスによる、入力された IP アドレスのデータ型に基づく IPv6 アドレスのルックアップ（タイプ AAAA）または IPv4 アドレスのルックアップ（タイプ A）（NetX Duo DNS クライアントでのみ使用可能）。

- nxd_dns_host_by_address_get サービスを使用した、Web ホスト名を取得するための IP アドレスの逆引き（タイプ PTR のクエリ）（NetX Duo DNS クライアントでのみ使用可能）。

NetX Duo DNS クライアントモジュールでは、nx_dns_host_by_name_get サービスおよび nx_dns_host_by_address_get サービスも引き続きサポートされます。これらは同等のサービスですが、IPv4 ネットワーク通信に限定されるため、開発時には代わりに nxd_dns_host_by_name_get サービスと nxd_dns_host_by_address_get サービスを使用することが推奨されます。

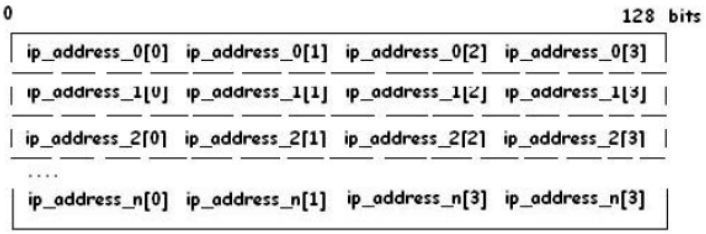
DNS リソースレコードタイプの拡張

[*Extended RR types support*] プロパティを有効にすると、NetX DNS クライアントモジュールでは以下のレコードタイプのクエリもサポートされます。

- CNAME：エイリアスの正式名が含まれます
- TXT：テキスト文字列が含まれます
- NS：権威 DNS サーバーが含まれます
- SOA：権威を持つゾーンの開始が含まれます
- MX：メールの送受信に使用されます
- SRV：ドメインで提供されるサービスに関する情報が含まれます

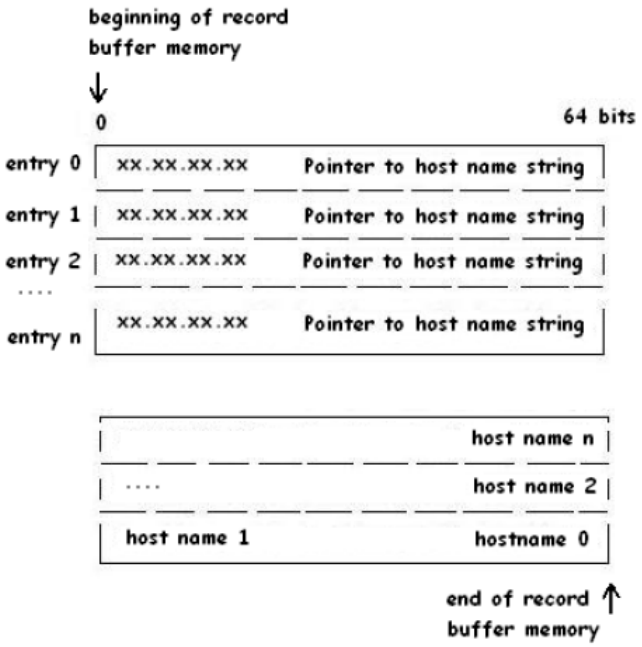
CNAME と TXT レコードタイプ以外では、アプリケーションは DNS データレコードを受信するために、4 バイトでアラインされたバッファを用意する必要があります。

NetX DNS クライアントモジュールでは、バッファ領域を効率的に使用するようにレコードデータが格納されます。次の例では、固定長のレコードバッファを示します（タイプ AAAA レコード）。



データ長が可変のレコードタイプを持つクエリの場合（たとえば NS レコードはホスト名の長さが可変です）、NetX DNS クライアントモジュールは次の方法でデータを保存します。

- DNS クライアントクエリで提供されたバッファを、固定長のデータと構造化されていないメモリ領域に編成します。
- メモリバッファの先頭を 4 バイトでアラインされたレコードエントリに編成します。
- 各レコードエントリに対して、IP アドレスと、その IP アドレスの可変長のデータを指すポインタを設定します。
- 各 IP アドレスの可変長のデータを、メモリバッファの終了位置から始まる構造化されていないメモリ領域に保存します。
- 後続の各レコードエントリについては、メモリの次の領域に前のレコードエントリの可変データに隣接して、可変長データを保存します。
- この可変データは、次のレコードエントリと可変データを格納するだけのメモリがなくなるまで、レコードエントリを格納する構造化されたメモリ領域に向かって「増えて」いきます。



NetX DNS クライアントクエリでは、レコードバッファに保存されたレコードの数がレコード記憶フォーマットを使用して返されます。アプリケーションはこの情報により、NS レコードをレコードバッファから抽出できます。アプリケーションは、記憶領域に対する receive_buffer ポインタを指定して nx_dns_domain_service_get をコールしてから、SRV エントリのサイズが既知であることを利用して SRV データを順番に抽出します。

```
NX_DNS_SRV_ENTRY *nx_dns_srv_entry_ptr[RECORD_COUNT];

status = nx_dns_domain_service_get(&client_dns,
                    (UCHAR *)"www.my_example.com",
                    &record_buffer[0], BUFFER_SIZE, &record_count,
                    NX_IP_PERIODIC_RATE);

for(i =0; i< record_count; i++)
{
    nx_dns_srv_entry_ptr[i] =
            (NX_DNS_SRV_ENTRY *)(record_buffer + i * sizeof(NX_DNS_SRV_ENTRY));
}
```

DNS キャッシュ

[*Cache support*] プロパティを有効にすると、NetX DNS クライアントモジュールで DNS キャッシュ機能がサポートされるようになります。アプリケーションでは、*nx_dns_cache_initialize* サービスを使用して、DNS キャッシュを設定する必要があります。キャッシュが有効な場合、DNS クライアントは DNS クエリを送信する前に DNS キャッシュのリソースレコードを確認します。キャッシュ内に回答が見つかった場合は、それを直接アプリケーションに返します。見つからない場合は、クエリメッセージを DNS サーバーに送信し、返信を待ちます。DNS クライアントはレスポンスメッセージを受信すると、利用できるキャッシュエントリがあればリソースレコードをキャッシュに追加します。

各キャッシュエントリは、リソースレコードを保持するために使用されるデータ構造体です。リソースレコード内の文字列エントリ（リソースレコードの名前とデータ）は長さが可変です。そのため、リソースレコード内に直接保存されることはありません。代わりに、キャッシュ内で文字列が格納されている実際のメモリ位置へのポインタがリソースレコードに設定されます。文字列とリソースレコードは同じキャッシュ内に置かれます。レコードはキャッシュの先頭から保存され、キャッシュの末尾に向かって順番に配置されます。文字列エントリはキャッシュの末尾から保存され、キャッシュの先頭に向かって順番に配置されます。各文字列エントリには、長さフィールドとカウンタフィールドがあります。文字列エントリがキャッシュに追加されたとき、同じ文字列が既にテーブルに存在していると、カウンタ値がインクリメントされます。その文字列に対してメモリが割り当てられることはありません。リソースレコードまたは新しい文字列エントリをキャッシュに追加できなくなると、キャッシュはいっぱいになったと見なされます。

NetX/NetX Duo DNS クライアントモジュールの動作に関する重要な注意事項と制限事項

NetX DNS クライアントでは、DNS メッセージを送信するためにパケットプールが必要です。デフォルトでは、アプリケーションは DNS クライアントサービスを使用する前にパケットプールを設定する必要があります。これは、IP インスタンスが使用するパケットプール（g_packet_pool0),）か、次の手順でプロジェクトに追加した独自のパケットプールです。[X-ware] -> [NetX Duo] -> [NetX Duo Packet Pool Instance]（NetX DNS クライアントでは [X-ware] -> [NetX] -> [NetX Packet Pool Instance])。

[*Use application packet pool*] が無効な場合、DNS クライアントモジュールは DNS クライアントの作成時にパケットプールを作成します。[*Maximum DNS queries size*] プロパティはパケットペイロードのサイズを決定します。デフォルトは 512 バイトです。[*Packets in DNS packet pool*] プロパティは、DNS クライアントパケットプール内のパケットの数を設定します（デフォルトは 6)。

注 : ユーザー作成のパケットプールの場合、パケットのサイズは [DNS maximum message size] プロパティの値になります。デフォルトのメッセージサイズは、512 バイトに UDP ヘッダー（8 バイト)、IPv4 ヘッダー（20 バイト）または IPv6 ヘッダー（40 バイト)、ネットワークフレームヘッダーのための領域を加えた大きさです。イーサネットフレームヘッダーは、4 バイトでアラインするために 16 バイトに切り上げられます。このユーザー作成のパケットプールは、DNS パケットの送信にのみ使用されます。IP パケットプールでは、パケットペイロードが 1568 バイトに設定されています。パケットプールのこの設定は、Synergy ネットワークドライバーとデバイス MTU（通常は 1518 バイト）にとって最適で、ドライバーレイヤーでパケットをチェーンすることも不要になります。

アプリケーションは DNS クエリを送信する前に、DNS クライアントが保持するサーバーリストに少なくとも 1 台の DNS サーバーを追加する必要があります。サーバーリストの最大サイズは [*Maximum number of DNS Servers in the Client server list*] プロパティで設定されます。

アプリケーションで DNS サーバーを追加するには、nx_dns_server_add サービス（IPv4 パケットのみ）または nxd_dns_server_add サービス（IPv4 パケットと IPv6 パケットをサポート）を使用します。

注 : 最新リリースの SSP では、NetX Duo DNS クライアントで [Client has DNS and Gateway Server] プロパティを有効にしても意味はありません。DNS クライアントが作成されるとき、IP インスタンスにルーター / ゲートウェイアドレスは設定されません。

NetX DNS クライアントでは [*Client has Gateway Server*] を無効にする必要があります。コンフィギュレータのビルドプロセスによって DNS クライアントが作成、初期化されるまで、IP インスタンスがゲートウェイアドレスを持つことはできないので、[*DNS Client IP instance gateway*] は使用できません。DNS サーバーを設定するには、*nx_dns_server_add* を使用することをお勧めします。

- アプリケーションで DNS キャッシュを設定するには、キャッシュメモリバッファとして定義済みのバッファをポイントするように指定して nx_dns_cache_initialize サービスを使用します。キャッシュがいっぱいになったときに通知を受け取るには、nx_dns_cache_notify_set サービスを使用します。このコールバックを「無効化」するには、nx_dns_cache_notify_clear サービスを使用してコールバックをクリアします。
- 便利な機能として、[*Clear previous DNS queries from queue*] プロパティがあります。このプロパティにより、DNS クライアントは現在のクエリに一致するレスポンスを検索する際、DNS サーバーからの以前のレスポンスを DNS クライアントの受信キューから削除できます。そのため、以前の DNS クエリに対して受信された古いパケットが削除され、DNS クライアントソケットがオーバーフローして有効なパケットがドロップされるのを防ぐことができます。
- NetX DNS サービスと NetX Duo DNS での同等のサービスについては、次のリストを参照してください。

NetX DNS API Service (IPv4 only)	NetX Duo DNS API Service (IPv4 and IPv6 supported)
nx_dns_host_by_name_get	nxd_dns_host_by_name_get
nx_dns_host_by_address_get	nxd_dns_host_by_address_get
nx_dns_server_get	nxd_dns_server_get
nx_dns_server_add	nxd_dns_server_add
nx_dns_server_remove	nxd_dns_server_remove

- DNS クライアントがサポートする DNS 要求は一度に 1 つです。別の DNS 要求を試みるスレッドは、その前の DNS 要求が完了するまで一時的にブロックされます。
- NetX DNS クライアントモジュールは、権威のある回答に含まれるデータを使用して別の DNS クエリを他の DNS サーバーに転送することはありません。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.17.4 アプリケーションへの NetX/NetX Duo DNS クライアントモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo DNS クライアントモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo DNS クライアントモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo DNS クライアントモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_dns0 NetX DNS Client	Threads	New Stack> X-Ware> NetX> Protocols> NetX DNS Client
g_dns0 NetX Duo DNS Client	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo DNS Client

次の図に示すように、NetX/NetX Duo DNS クライアントモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

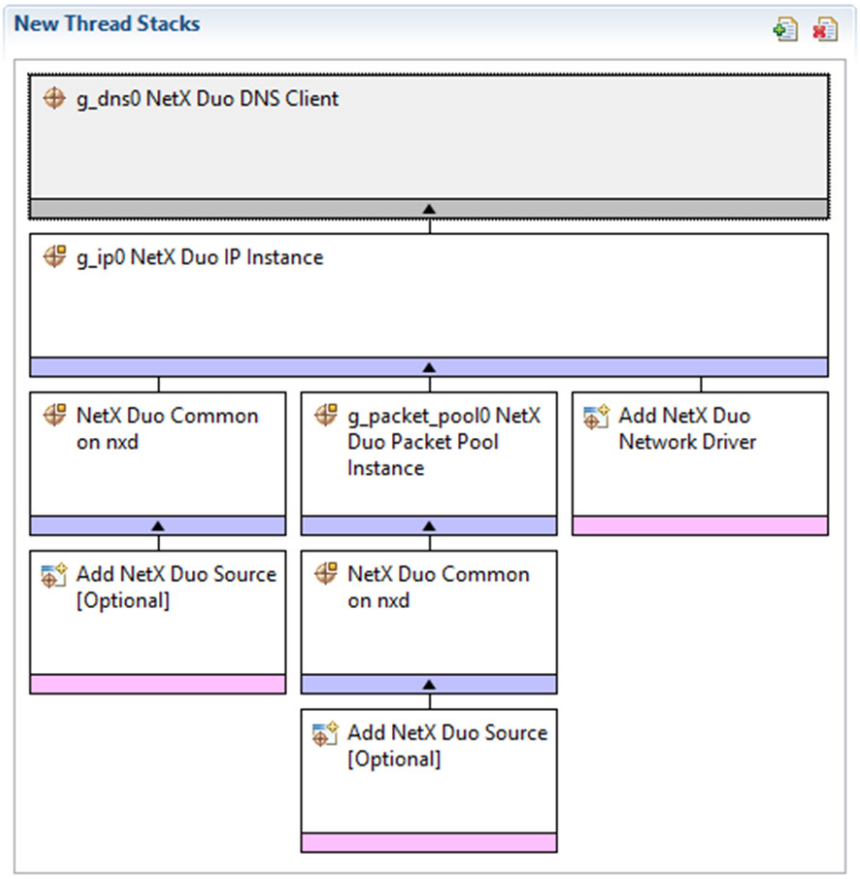


図 348:NetX/NetX Duo DNS クライアントモジュールのスタック

上記のスタックで、NetX ネットワークドライバー (NetX Duo スタックの場合は NetX Duo ネットワークドライバー) はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.17.5 NetX/NetX Duo DNS クライアントモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo DNS クライアントモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo DNS クライアントモジュールの構成設定

ISDE Property	Value	Description
DNS Control Type of Service	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost Default: Normal	DNS control type of service selection
Socket fragmentation option	Don't fragment, Fragment okay Default: Don't fragment	Socket fragmentation option selection
Time to live	128	Time to live selection
**Client DNS IP version	IPv4, IPv6 Default: IPv4	Client DNS IP version selection
Maximum number of DNS Servers in Client server list	5	Maximum number of DNS Servers in Client server list selection
Maximum DNS queries size (bytes)	512	Maximum DNS queries size selection
Packets in DNS packet pool (units)	16	Packets in DNS packet pool selection

ISDE Property	Value	Description
Maximum retries for a server	3	Maximum retries for a server selection
Maximum duration to retransmit a DNS query (seconds)	64	Maximum duration to retransmit a DNS query selection
Packet allocate timeout (seconds)	1	Packet allocate timeout selection
Client has DNS and Gateway server	Enable, Disable Default: Disable	DNS control type of service selection
DNS Client IP instance gateway (use commas for separation) (Not included in NetX Duo)	192,16,0,1	DNS client IP instance gateway selection
Use application packet pool	Enable, Disable Default: Enable	Use application packet pool selection
Clear previous DNS queries from queue	Enable, Disable Default: Disable	Clear previous DNS queries from queue selection
Extended RR types support	Enable, Disable Default: Disable	Extended RR types support selection
Cache support	Enable, Disable Default: Disable	Cache support selection
Name	g_dns0	Module name
Name of generated initialization function	dns_client_init0	Name of generated initialization function selection

ISDE Property	Value	Description
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

ローレベルモジュールのデフォルト以外の設定が望ましいこともあります。たとえば、異なる MAC アドレスまたは IP アドレスの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo DNS クライアントのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
Default Gateway Address (use commas for separation)	0,0,0,0	Default gateway address selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection

ISDE Property	Value	Description
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function

ISDE Property	Value	Description
Link status change callback	NULL	Link status change callback selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo DNS クライアントモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo DNS クライアントモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin Selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin

Property	Value	Description
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.17.6 アプリケーションでの NetX/NetX Duo DNS クライアントモジュールの使用

一般的なアプリケーションで NetX/NetX Duo DNS クライアントモジュールを使用する際の手順は次のとおりです。

1) nx_ip_status_check API を使用して、有効な IP アドレスを待機します。
2) nx_dns_cache_initialize をコールして、DNS キャッシュを有効にします（オプション）。
3) nx_dns_server_add API を使用して、1 台以上のサーバーをクライアントのリストに追加します。
4) nxd_dns_host_by_name_get API を使用して、DNS 名クエリを送信して IP アドレスを取得します。
5) DNS クライアントによってパッケージ化されたレコードのサイズと配置に関する情報を利用して、DNS サーバーのリソースレコードを抽出します。

次の図は、一般的な手順を示した通常の動作フロー図です。

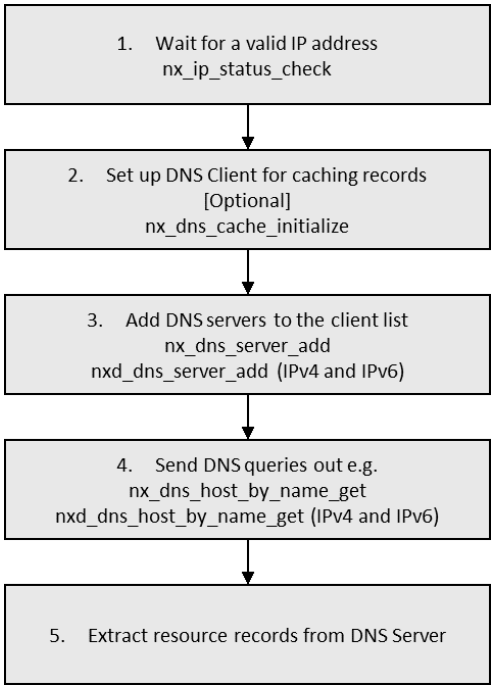


図 349:一般的な NetX/NetX Duo DNS クライアントモジュールアプリケーションのフロー図

4.3.18 NetX/NetX Duo FTP クライアント

ファイル転送プロトコル（FTP）は、ファイル転送のために設計されたプロトコルです。FTP は、信頼性の高い伝送制御プロトコル（TCP）サービスを利用してファイル転送機能を実行します。FTP による実際のファイル転送は、専用の FTP 接続上で実行されます。

注 :NetX ™ FTP クライアントモジュールは、IPv4 のみをサポートします。NetX Duo ™ FTP クライアントモジュールは、IPv4 と IPv6 の両方のネットワークに対応しています。IPv6 によって FTP プロトコルが直接的な変更を受けることはありませんが、IPv6 に対応するためには、元の NetX FTP API にいくつかの変更が必要です。それらについてはこのドキュメントで説明します。

4.3.18.1 NetX/NetX Duo FTP クライアントモジュールの特長

- マルチスレッドサポート
  - ハイレベル API
  - 接続と切断
  - ディレクトリ操作
  - ファイル操作

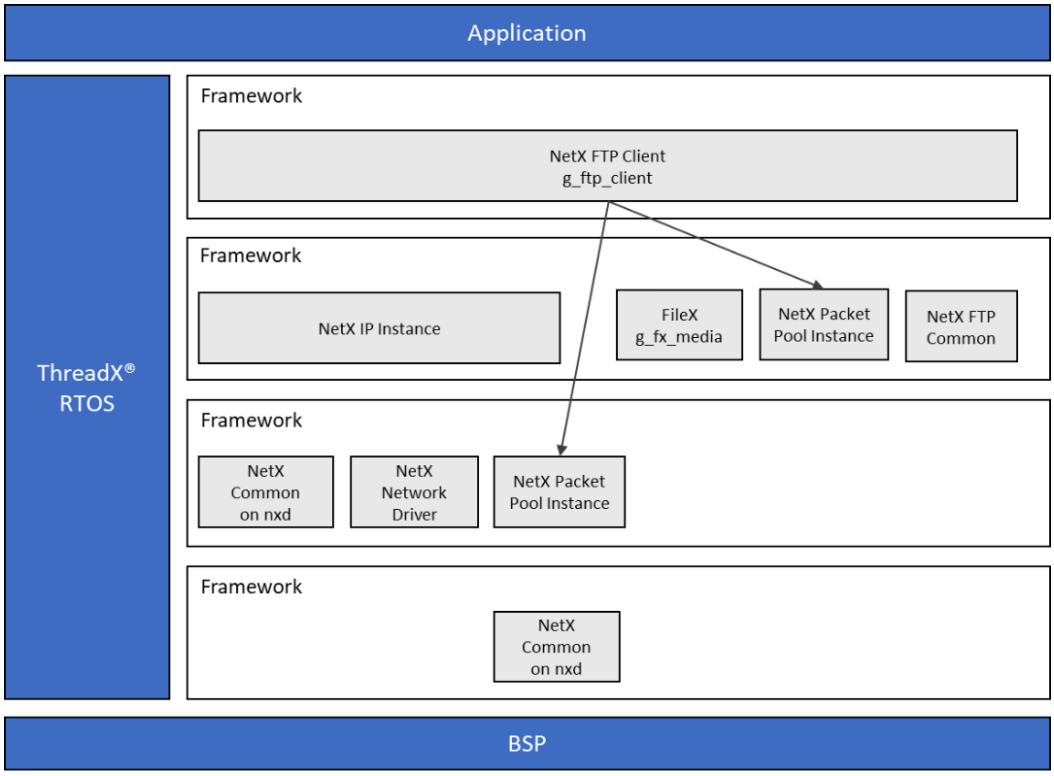


図 350:NetX/NetX Duo FTP クライアントモジュールのブロック図

注 : 上の図で、FileX および NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.18.2 NetX/NetX Duo FTP クライアントモジュールの API の概要

NetX/NetX Duo FTP クライアントでは、クライアントの作成と削除、サーバーへの接続と切断、ディレクトリ操作、ファイル操作のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX/NetX Duo FTP クライアントモジュールの API の要約

Function Name	Example API Call and Description
nx_ftp_client_create	nx_ftp_client_create(&my_client, "My Client", &client_ip, 2000, &client_pool); Delete an FTP Client instance.

Function Name	Example API Call and Description
nx_ftp_client_delete	nx_ftp_client_delete(&my_client); Delete an FTP Client instance.
nx_ftp_client_connect	nxd_ftp_client_connect(&my_client, server_ip_addr, NULL, NULL, 100); Connect to an FTP Server with IPv6 and IPv4 support.
**nxd_ftp_client_connect	nxd_ftp_client_connect(&my_client, server_ip_addr, NULL, NULL, 100); Connect to an FTP Server with IPv6 and IPv4 support.
nx_ftp_client_disconnect	nx_ftp_client_disconnect(&my_client, 200); Disconnect from the FTP Server.
nx_ftp_client_create	nx_ftp_client_directory_create(&my_client, “my_dir”, 200); Create a directory on the server.
nx_ftp_client_directory_set	nx_ftp_client_directory_listing_set(&my_client, “my_dir”, &my_packet, 200); Get a directory listing from the server.
nx_ftp_client_directory_delete	nx_ftp_client_directory_delete(&my_client, “my_dir”, 200); Delete a directory on the server.
nx_ftp_client_directory_listing_get	nx_ftp_client_directory_listing_get(&my_client, “my_dir”, &my_packet, 200); Get a directory listing from the server.

Function Name	Example API Call and Description
nx_ftp_client_directory_listing_continue	nx_ftp_client_directory_listing_continue(&my_client, &my_packet, 200); Continue a directory listing from the server.
nx_ftp_client_file_open	nx_ftp_client_file_open(&my_client, "my_file.txt", NX_FTP_OPEN_FOR_READ, 200); Open a file on the server.
nx_ftp_client_file_close	nx_ftp_client_file_close(&my_client, 200); Close the currently open file on the server.
nx_ftp_client_file_read	nx_ftp_client_file_read(&my_client, &my_packet, 200); Read from a file already open on the server.
nx_ftp_client_file_write	nx_ftp_client_file_write(&my_client, my_packet, 200); Write to an already open file on the server.
nx_ftp_client_file_rename	nx_ftp_client_file_rename(&my_client, "my_file.txt", "new_file.txt", 200); Rename a file on the server.
nx_ftp_client_file_delete	nx_ftp_client_file_delete(&my_client, "my_file.txt", 200); Delete a file on the server.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :** この API は、NetX Duo FTP クライアントでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

ステータス戻り値

Name	Description
NX_SUCCESS	Successful FTP function.
NX_FTP_NOT_DISCONNECTED	FTP client is already connected.
NX_FTP_200_CODE_NOT_RECEIVED	Server rejects FTP connection.
NX_FTP_300_CODE_NOT_RECEIVED	Server rejects user/password.
NX_PTR_ERROR	Invalid FTP, username, or password pointer.
NX_CALLER_ERROR	Invalid caller of this service.
NX_IP_ADDRESS_ERROR	Invalid IP address.
NX_FTP_NO_2XX_RESPONSE_XXD	FTP server error response.
NX_FTP_NO_2XX_RESPONSE_PORT	FTP server response to PORT.
NX_FTP_NO_1XX_RESPONSE	FTP server response to NLST.
NX_FTP_END_OF_LISTING	No more entries in directory.
NX_FTP_NO_2XX_RESPONSE_DISCONNECT	FTP server response to disconnect.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.3.18.3 NetX/NetX Duo FTP クライアントモジュールの動作の概要

FTP クライアントの要件

NetX FTP クライアントパッケージが適切に動作するためには、NetX が必要です。同様に、NetX Duo FTP クライアントパッケージには NetX Duo が必要です。ホストアプリケーションは、NetX サービスおよび定期タスクを実行するための IP インスタンスを作成する必要があります。FTP ホストアプリケーションを IPv6 ネットワークで実行する場合は、IP タスクで IPv6 および ICMPv6 を有効にする必要があります。また、IPv6 と IPv4 のどちらかのネットワークで TCP を有効にする必要もあります。IPv6 ホストアプリケーションでは、IPv6 API と DHCPv6 のどちらかまたは両方を使用して、リンクローカルおよびグローバル IPv6 アドレスを設定する必要があります。

また、FTP サーバーおよび FTP クライアントは、FileX® 組み込みファイルシステムと連携するように設計されています。FileX を使用できない場合、ホスト開発者は filex_stub.h で推奨されているガイドラインに従って、このファイルに示されている各サービスを定義することにより独自のファイルシステムを実装するか置き換えることができます。

NetX FTP パッケージの FTP クライアントの部分には、追加要件はありません。

FTP ファイル名

FTP ファイル名は、ターゲットファイルシステム（通常は FileX）のフォーマットに従う必要があります。これは末尾が NULL の ASCII 文字列である必要があり、必要に応じてフルパス情報を含みます。NetX FTP の実装では、FTP ファイル名のサイズに制限は指定されていません。パケットプールペイロードのサイズは、最も長いパスとファイル名を格納できる必要があります。

FTP クライアントのコマンド

FTP には、接続を開き、ファイル操作とディレクトリ操作を実行するためのシンプルなメカニズムが用意されています。通常は、TCP の周知されたポート 21 で接続が正常に確立された後でクライアントによって発行される標準的な FTP コマンドのセットがあります。以下に FTP の基本的なコマンドの一部を示します。IPv6 で FTP を実行する場合の唯一の違いは、PORT コマンドが EPRT コマンドに置き換えられることです。

FTP クライアントのコマンド

FTP Command	Meaning
CWD path	Change working directory
DELE filename	Delete specified file name
EPRT ip_address, port	Provide IPv6 address and Client data port
LIST directory	Get directory listing
MKD directory	Make new directory
NLST directory	Get directory listing
NOOP	No operation, returns success
PASS password	Provide password for login
PORT ip_address,port	Provide IP address and Client data port
PWD path	Pickup current directory path
QUIT	Terminate Client connection
RETR filename	Read specified file
RMD directory	Delete specified directory
RNFR oldfilename	Specify file to rename
RNTO newfilename	Rename file to supplied file name
STOR filename	Write specified file
TYPE I	Select binary file image

FTP Command	Meaning
USER username	Provide username for login

注 : これらの ASCII コマンドは、NetX FTP クライアントソフトウェアが FTP サーバーとの間で FTP 操作を実行するために、内部で使用します。

FTP サーバーのレスポンス

FTP サーバーは、クライアント要求を処理すると 3 桁のコードによる ASCII 形式のレスポンス、およびオプションでその後に続けて ASCII テキストを返します。FTP クライアントソフトウェアはこの数値レスポンスを使用して、操作が成功したか失敗したかを判断します。次のリストに、クライアント要求に対する FTP サーバーのさまざまなレスポンスを示します。

1 つ目の数値フィールド

First Numeric Field	Meaning
1xx	Positive preliminary status – another reply coming.
2xx	Positive completion status.
3xx	Positive preliminary status – another command must be sent.
4xx	Temporary error condition.
5xx	Error condition.

2 つ目の数値フィールド

Second Numeric Field	Meaning
0xx	Syntax error in command.
1xx	Positive preliminary status – another reply coming.
2xx	Positive completion status.
3xx	Positive preliminary status – another command must be sent.
4xx	Temporary error condition.
5xx	Error condition.

FTP ライト要求

1) クライアントがサーバーポート 21 への TCP 接続を発行します。
2) 成功を示すためにサーバーが 220 レスポンスを送信します。
3) クライアントがユーザー名を含む USER メッセージを送信します。
4) 成功を示すためにサーバーが 331 レスポンスを送信します。
5) クライアントがパスワードを含む PASS メッセージを送信します。
6) 成功を示すためにサーバーが 230 レスポンスを送信します。
7) バイナリ転送のためにクライアントが TYPE I メッセージを送信します。
8) 成功を示すためにサーバーが 200 レスポンスを送信します。
9) IPv6 アプリケーション：クライアントが IP アドレスとポートを含む EPRT メッセージを送信します。
   IPv4 アプリケーション：クライアントが IP アドレスとポートを含む PORT メッセージを送信します。
10) 成功を示すためにサーバーが 200 レスポンスを送信します。
11) クライアントがライト先のファイル名を含む STOR メッセージを送信します。
12) サーバーがデータソケットを作成し、先行する EPRT または PORT コマンドで指定されたクライアントデータポートで接続します。
13) ファイルライトが始まったことを示すためにサーバーが 125 レスポンスを送信します。
14) クライアントがデータ接続を通じてファイルの内容を送信します。ファイル全体が転送されるまで、このプロセスが続きます。
15) 完了すると、クライアントがデータ接続を切断します。
16) ファイルライトが成功したことを示すためにサーバーが 250 レスポンスを送信します。
17) FTP 接続を終了するためにクライアントが QUIT を送信します。
18) 切断が成功したことを示すためにサーバーが 221 レスポンスを送信します。
19) サーバーが FTP 接続を切断します。

FTP 認証

FTP 接続を確立するたびに、クライアントはサーバーにユーザー名とパスワードを提供する必要があります。一部の FTP サイトでは、具体的なユーザー名とパスワードなしで FTP アクセスができる Anonymous FTP が許可されています。このタイプの接続では、ユーザー名は anonymous、パスワードは完全なメールアドレスを指定する必要があります。

NetX FTP クライアントと NetX Duo FTP クライアントに対して、ログインおよびログアウトの認証ルーチンを提供する必要があります。これらは nxd_ftp_server_create および nx_ftp_server_create サービスの中で提供し、パスワード処理からコールされます。両者の違いは次のとおりです。nxd_ftp_server_create でのログインおよびログアウトの認証関数に対する入力関数ポインタでは、NetX Duo のアドレスタイプが NXD_ADDRESS. であると想定されます。このデータ型は IPv4 アドレスと IPv6 のアドレスのどちらも格納できるため、この関数は IPv4 と IPv6 の両方のネットワークをサポートする Duo サービスとなっています。nx_ftp_server_create でのログインおよびログアウトの認証関数に対する入力関数ポインタでは、IP アドレスタイプが ULONG であると想定されます。この関数は IPv4 ネットワークのみに制限されます。できるだけ Duo サービスを使用することをお勧めします。

ログイン関数から NX_SUCCESS, が返された場合は、接続が認証され、FTP 操作が実行できるようになります。ログイン関数から NX_SUCCESS, 以外の値が返された場合、接続の試行は拒否されました。

NetX/NetX Duo FTP クライアントモジュールの動作に関する重要な注意事項と制限事項

FTP のマルチスレッドサポート

- NetX FTP クライアントサービスは、複数のスレッドから同時にコールすることができます。ただし、特定の FTP クライアントインスタンスに対するリードまたはライト要求は、同じスレッドから順番に行う必要があります。

RFC

- NetX Duo FTP は、RFC 959、RFC 2428、および関連する RFC に準拠しています。

FTP の制約

FTP の規格には、ファイルデータの表示に関してさまざまなオプションが用意されています。NetX FTP には ls – al などのスイッチオプションは実装されていません。NetX FTP サーバーと NetX Duo FTP サーバーでは、要求とその引数を連続した複数のパケットでなく 1 つのパケットで受け取ることが想定されています。

UNIX での実装と同様に、NetX FTP ではファイル形式について以下の制約が前提とされます。

- ファイルタイプ：バイナリ
- ファイル形式：Nonprint のみ
- ファイル構造：File Structure のみ

4.3.18.4 アプリケーションへの NetX/NetX Duo FTP クライアントモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo FTP クライアントモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo FTP クライアントモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo FTP クライアントモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_dns0 NetX FTP Client	Threads	New Stack> X-Ware> NetX> Protocols> NetX FTP Client
g_dns0 NetX Duo FTP Client	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo FTP Client

次の図に示すように、NetX/NetX Duo FTP クライアントモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

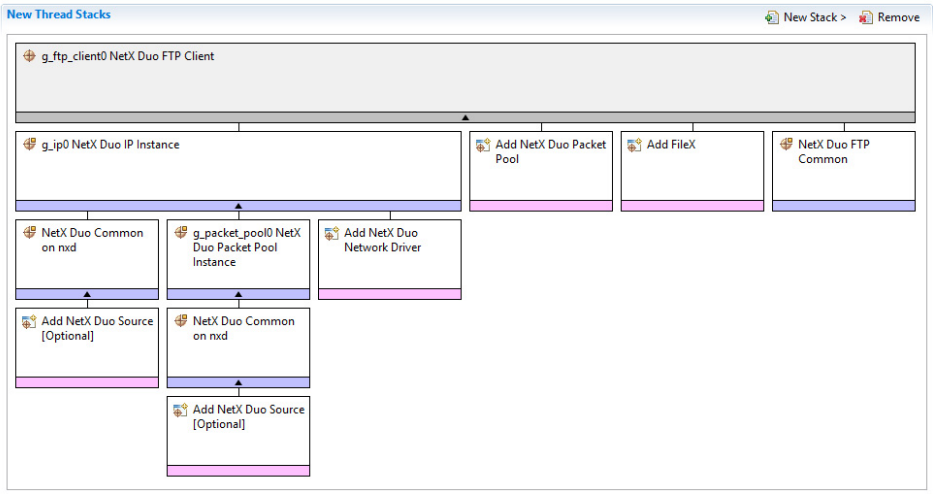


図 351:NetX/NetX Duo FTP クライアントモジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

また、上記のスタックで FileX スタックはまだ設定されていません。FileX モジュールには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- FileX スタブ
- ブロックメディア上の FileX（sf_block_media_ram) 上のブロックメディアフレームワークに実装）
- USB 大容量記憶装置上の FileX（USBX ホストクラス大容量記憶装置上に実装）

4.3.18.5 NetX/NetX Duo FTP クライアントモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo FTP クライアントモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo FTP クライアントモジュールの構成設定

ISDE Property	Value	Description
**TCP socket to use	NX_ANY_PORT	TCP socket to use selection
Name	g_ftp_client0	Module name.
TCP socket window size (bytes)	2048	TCP socket window size selection
Name of generated initialization function	ftp_client_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

ローレベルモジュールのデフォルト以外の設定が望ましいこともあります。たとえば、イーサネットペリフェラル用に異なるピンの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo FTP クライアントのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name

ISDE Property	Value	Description
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
Default Gateway Address (use commas for separation)	0,0,0,0	Default gateway address selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection

ISDE Property	Value	Description
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function
Link status change callback	NULL	Link status change callback selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo FTP 共通インスタンスの構成設定

ISDE Property	Value	Description
FileX support	Enabled, Disabled Default: Enabled	FileX support selection
Control Type of Service	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost Default: Normal	Control type of service selection

ISDE Property	Value	Description
Data Type of Service	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost Default: Normal	Data type of service selection
Fragmentation option	Don't fragment, Fragment okay Default: Don't fragment	Fragment option selection
Time to live	128	Time to live selection
Duration between client inactivity check (seconds)	60	Duration between client inactivity check selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name

ISDE Property	Value	Description
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo FTP クライアントモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo FTP クライアントモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin Selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.18.6 アプリケーションでの NetX/NetX Duo FTP クライアントモジュールの使用

一般的なアプリケーションで NetX/NetX Duo FTP クライアントモジュールを使用する際の手順は次のとおりです。

1) nx_ftp_client_create API を使用して FTP クライアントを作成します。
2) nx_ftp_client_connect API を使用して、FTP サーバーに接続します。

以下の手順を使用してファイルに書き込むことができます。

1) nx_ftp_client_open API を使用してファイルを開きます。
2) 必要に応じて、nx_ftp_client_write API を使用してファイルに書き込みます。

3) nx_ftp_client_close API を使用してファイルを閉じます。

以下の手順を使用してファイルから読み取ることができます。

1) nx_ftp_client_open API を使用してファイルを開きます。
2) 必要に応じて、nx_ftp_client_read API を使用してファイルから読み取ります。
3) nx_ftp_client_close API を使用してファイルを閉じます。

次の図は、一般的な手順を示した通常の動作フロー図です。

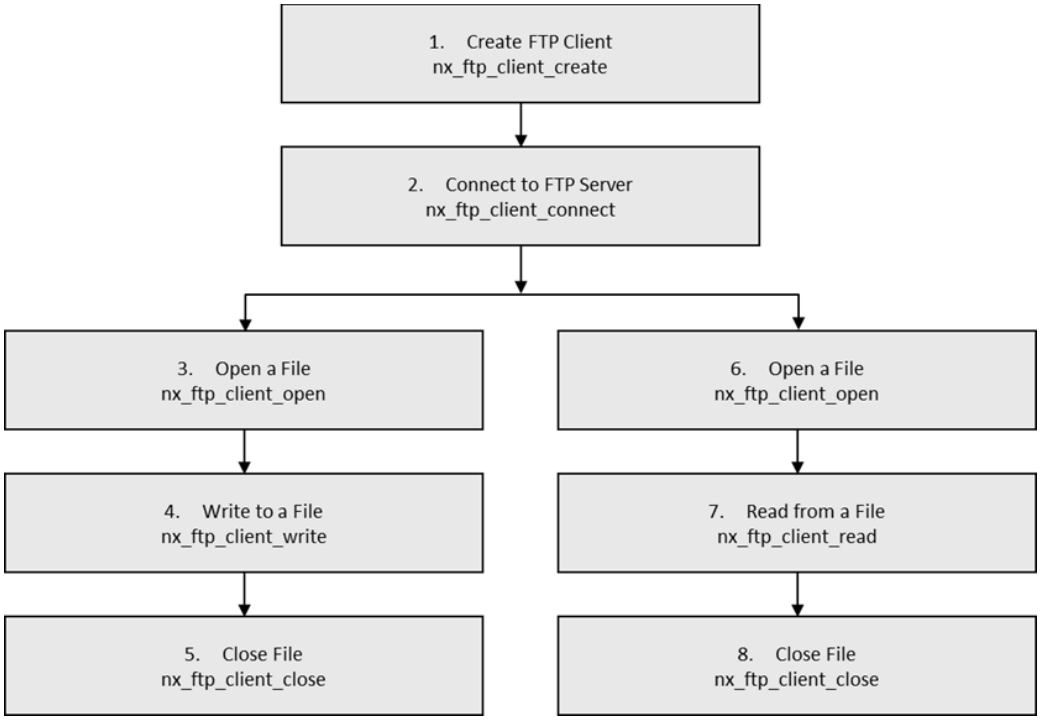


図 352:一般的な NetX/NetX Duo FTP クライアントモジュールアプリケーションのフロー図

4.3.19 NetX/NetX Duo FTP サーバー

ファイル転送プロトコル（FTP）は、ファイル転送のために設計されたプロトコルです。FTP は、信頼性の高い伝送制御プロトコル（TCP）サービスを利用してファイル転送機能を実行します。FTP による実際のファイル転送は、専用の FTP 接続上で実行されます。

注 :NetX ™ FTP サーバーモジュールは IPv4 専用です。NetX Duo ™ FTP サーバーモジュールは、IPv4 と IPv6 の両方のネットワークに対応しています。IPv6 によって FTP プロトコルが直接的な変更を受けることはありませんが、IPv6 に対応するためには、元の NetX FTP API にいくつかの変更が必要です。それらについてはこのドキュメントで説明します。

4.3.19.1 NetX/NetX Duo FTP サーバーモジュールの特長

- NetX は IPv4 専用です
- NetX Duo は IPv4 と IPv6 の両方のネットワークをサポートしています
- FileX® ファイルシステムと連携します
- ファイル名のサイズは無制限で、末尾が NULL の ASCII 文字列を使用します
- クライアント要求に応えるために TCP ポート 21 をサポートします
- サービスの作成、開始、停止、削除のためのハイレベル API を提供します
- NetX FTP および NetX Duo FTP は、RFC 959、RFC 2428、および関連する RFC に準拠しています。

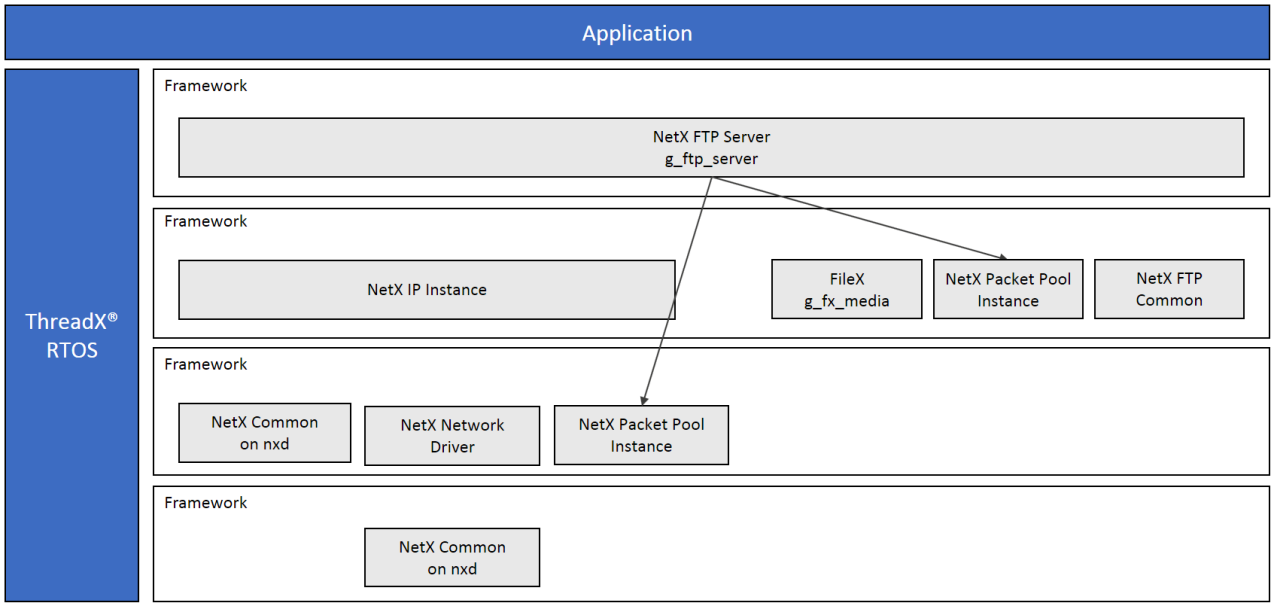


図 353:NetX/NetX Duo FTP サーバーモジュールのブロック図

注 : 上の図で、FileX および NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.19.2 NetX/NetX Duo FTP サーバーモジュールの API の概要

NetX FTP サーバーでは、サービスの作成、削除、開始、停止のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX/NetX Duo FTP サーバーモジュールの API の要約

Function Name	Example API Call and Description
nx_ftp_server_create	nx_ftp_server_create(&my_server, "My Server Name", &ip_0, &ram_disk, stack_ptr, stack_size, &pool_0, my_login, my_logout); Create FTP Server with IPv4 support only.
**nxd_ftp_server_create	nxd_ftp_server_create(&my_server, "My Server Name", &ip_0, &ram_disk, stack_ptr, stack_size, &pool_0, my_duo_login, my_duo_logout); Create FTP Server with IPv4 and IPv6 support.
nx_ftp_server_delete	nx_ftp_server_delete(&my_server); Delete FTP Server.
nx_ftp_server_start	nx_ftp_server_start(&my_server); Start FTP Server.
nx_ftp_server_stop	nx_ftp_server_stop(&my_server); Stop FTP Server.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :** この API は、NetX Duo FTP クライアントでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

ステータス戻り値

Name	Description
NX_SUCCESS	Successful FTP server function.

Name	Description
NX_PTR_ERROR	Invalid FTP pointer.
NX_CALLER_ERROR	Invalid caller of this service.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.3.19.3 NetX/NetX Duo FTP サーバーモジュールの動作の概要

FTP クライアントとサーバーの動作は密接にリンクしているため、以下の説明ではクライアントとサーバーの両方の動作の主要な要素を取り上げています。

FTP の要件

NetX FTP パッケージが適切に動作するためには、NetX または NetX Duo が必要です。ホストアプリケーションは、NetX サービスおよび定期タスクを実行するための IP インスタンスを作成する必要があります。FTP ホストアプリケーションを IPv6 ネットワークで実行する場合は、IP タスクで IPv6 および ICMPv6 を有効にする必要があります。また、IPv6 と IPv4 のどちらかのネットワークで TCP を有効にする必要もあります。IPv6 ホストアプリケーションでは、IPv6 API と DHCPv6 のどちらかまたは両方を使用して、リンクローカルおよびグローバル IPv6 アドレスを設定する必要があります。

また、FTP サーバーおよび FTP クライアントは、FileX 組み込みファイルシステムと連携するように設計されています。FileX を使用できない場合、ホスト開発者は filex_stub.h で推奨されているガイドラインに従って、このファイルに示されている各サービスを定義することにより独自のファイルシステムを実装するか置き換えることができます。これについては、このガイドの後のセクションで説明します。

NetX FTP パッケージの FTP クライアントの部分には、追加要件はありません。

NetX FTP パッケージの FTP サーバーの部分には、いくつかの追加要件があります。すべてのクライアント FTP コマンド要求を処理するために、TCP の周知されたポート 21 への完全なアクセスが要求されるほか、すべてのクライアント FTP データ転送を処理するために、周知されたポート 20 への完全なアクセスも必要です。

FTP ファイル名

FTP ファイル名は、ターゲットファイルシステム（通常は FileX）のフォーマットに従う必要があります。これは末尾が NULL の ASCII 文字列である必要があり、必要に応じてフルパス情報を含みます。NetX FTP の実装では、FTP ファイル名のサイズに制限は指定されていません。パケットプールペイロードのサイズは、最も長いパスやファイル名を格納できる必要があります。

FTP クライアントのコマンド

FTP には、接続を開き、ファイル操作とディレクトリ操作を実行するためのシンプルなメカニズムが用意されています。TCP の周知されたポート 21 で接続が正常に確立された後でクライアントによって発行される標準的な FTP コマンドのセットがあります。次の表に FTP の基本的なコマンドの一部を示します。IPv6 で FTP を実行する場合の唯一の違いは、PORT コマンドが EPRT コマンドに置き換えられることです。

FTP サーバーのコマンド

FTP Command	Meaning
CWD path	Change working directory
DELE filename	Delete specified file name

FTP Command	Meaning
EPRT ip_address, port	Provide IPv6 address and Client data port
LIST directory	Get directory listing
MKD directory	Make new directory
NLST directory	Get directory listing
NOOP	No operation, returns success
PASS password	Provide password for login
PORT ip_address,port	Provide IP address and Client data port
PWD path	Pickup current directory path
QUIT	Terminate Client connection
RETR filename	Read specified file
RMD directory	Delete specified directory
RNFR oldfilename	Specify file to rename
RNTO newfilename	Rename file to supplied file name
STOR filename	Write specified file
TYPE I	Select binary file image
USER username	Provide username for login

これらの ASCII コマンドは、NetX FTP クライアントソフトウェアが FTP サーバーとの間で FTP 操作を実行するために、内部で使用します。

FTP サーバーのレスポンス

FTP サーバーは、クライアント要求を処理すると 3 桁のコードによる ASCII 形式のレスポンス、およびオプションでその後に続けて ASCII テキストを返します。FTP クライアントソフトウェアはこの数値レスポンスを使用して、操作が成功したか失敗したかを判断します。次のリストに、クライアント要求に対する FTP サーバーのさまざまなレスポンスを示します。

1 つ目の数値フィールド

First Numeric Field	Meaning
1xx	Positive preliminary status – another reply coming.
2xx	Positive completion status.
3xx	Positive preliminary status – another command must be sent.
4xx	Temporary error condition.
5xx	Error condition.

2 つ目の数値フィールド

Second Numeric Field	Meaning
x0x	Syntax error in command.
x1x	Informational message.
x2x	Connection related.
x3x	Authentication related.
x4x	Unspecified.
x5x	File system related.

たとえば、QUIT コマンドによって FTP 接続を切断するクライアント要求に対しては、通常、切断が成功するとサーバーからコード 221 のレスポンスがあります。

FTP 通信

FTP サーバーでは、クライアント要求に応えるために、周知された TCP ポート 21 が使用されます。FTP クライアントは、利用可能な任意の TCP ポートを使用できます。FTP イベントの一般的なシーケンスは次のとおりです。既に説明したとおり、IPv6 で FTP を実行する場合の唯一の違いは、PORT コマンドが EPRT コマンドに置き換えられることです。

FTP リードファイル要求

1) クライアントがサーバーポート 21 への TCP 接続を発行します。
2) 成功を示すためにサーバーが 220 レスポンスを送信します。

3) クライアントが「ユーザー名」を含む USER メッセージを送信します。
4) 成功を示すためにサーバーが 331 レスポンスを送信します。
5) クライアントが「パスワード」を含む PASS メッセージを送信します。
6) 成功を示すためにサーバーが 230 レスポンスを送信します。
7) バイナリ転送のためにクライアントが TYPE I メッセージを送信します。
8) 成功を示すためにサーバーが 200 レスポンスを送信します。
9) クライアントが IP アドレスとポートを含む PORT メッセージを送信します。
10) 成功を示すためにサーバーが 200 レスポンスを送信します。
11) クライアントがリード元のファイル名を含む RETR メッセージを送信します。
12) サーバーがデータソケットを作成し、EPRT コマンドで指定されたクライアントデータポートで接続します。
13) ファイルリードが始まったことを示すためにサーバーが 125 レスポンスを送信します。
14) サーバーがデータ接続を通じてファイルの内容を送信します。ファイル全体が転送されるまで、このプロセスが続きます。
15) 完了すると、サーバーがデータ接続を切断します。
16) ファイルリードが成功したことを示すためにサーバーが 250 レスポンスを送信します。
17) FTP 接続を終了するためにクライアントが QUIT を送信します。
18) 切断が成功したことを示すためにサーバーが 221 レスポンスを送信します。
19) サーバーが FTP 接続を切断します。

FTP ライト要求：

1) クライアントがサーバーポート 21 への TCP 接続を発行します。
2) 成功を示すためにサーバーが 220 レスポンスを送信します。
3) クライアントが「ユーザー名」を含む USER メッセージを送信します。
4) 成功を示すためにサーバーが 331 レスポンスを送信します。
5) クライアントが「パスワード」を含む PASS メッセージを送信します。
6) 成功を示すためにサーバーが 230 レスポンスを送信します。
7) バイナリ転送のためにクライアントが TYPE I メッセージを送信します。
8) 成功を示すためにサーバーが 200 レスポンスを送信します。
9) IPv6 アプリケーション：クライアントが IP アドレスとポートを含む EPRT メッセージを送信します。
IPv4 アプリケーション：クライアントが IP アドレスとポートを含む PORT メッセージを送信します。
10) 成功を示すためにサーバーが 200 レスポンスを送信します。
11) クライアントがライト先のファイル名を含む STOR メッセージを送信します。
12) サーバーがデータソケットを作成し、先行する EPRT または PORT コマンドで指定されたクライアントデータポートで接続します。
13) ファイルライトが始まったことを示すためにサーバーが 125 レスポンスを送信します。
14) クライアントがデータ接続を通じてファイルの内容を送信します。ファイル全体が転送されるまで、このプロセスが続きます。

15) 完了すると、クライアントがデータ接続を切断します。
16) ファイルライトが成功したことを示すためにサーバーが 250 レスポンスを送信します。
17) FTP 接続を終了するためにクライアントが QUIT を送信します。
18) 切断が成功したことを示すためにサーバーが 221 レスポンスを送信します。
19) サーバーが FTP 接続を切断します。

FTP 認証

FTP 接続を確立するたびに、クライアントはサーバーにユーザー名とパスワードを提供する必要があります。一部の FTP サイトでは、具体的なユーザー名とパスワードなしで FTP アクセスができる Anonymous FTP が許可されています。このタイプの接続では、ユーザー名は **anonymous**、パスワードは完全なメールアドレスを指定する必要があります。

NetX FTP に対して、ログインおよびログアウトの認証ルーチンを提供する必要があります。これらは nxd_ftp_server_create および nx_ftp_server_create サービスの中で提供し、パスワード処理からコールされます。両者の違いは次のとおりです。nxd_ftp_server_create でのログインおよびログアウトの認証関数に対する入力関数ポインタでは、NetX Duo のアドレスタイプが NXD_ADDRESS. であると想定されます。このデータ型は IPv4 アドレスと IPv6 のアドレスのどちらも格納できるため、この関数は IPv4 と IPv6 の両方のネットワークをサポートする **duo** サービスとなっています。nx_ftp_server_create でのログインおよびログアウトの認証関数に対する入力関数ポインタでは、IP アドレスタイプが ULONG であると想定されます。この関数は IPv4 ネットワークのみに制限されます。できるだけ **duo** サービスを使用することをお勧めします。

ログイン関数から NX_SUCCESS, が返された場合は、接続が認証され、FTP 操作が実行できるようになります。ログイン関数から NX_SUCCESS, 以外の値が返された場合、接続の試行は拒否されました。

NetX/NetX Duo FTP サーバーモジュールの動作に関する重要な注意事項と制限事項

FTP のマルチスレッドサポート

- NetX FTP クライアントサービスは、複数のスレッドから同時にコールすることができます。ただし、特定の FTP クライアントインスタンスに対するリードまたはライト要求は、同じスレッドから順番に行う必要があります。

RFC

- NetX Duo FTP は、RFC 959、RFC 2428、および関連する RFC に準拠しています。

FTP の制約

FTP の規格には、ファイルデータの表示に関してさまざまなオプションが用意されています。NetX FTP には ls – al などのスイッチオプションは実装されていません。NetX FTP サーバーと NetX Duo FTP サーバーでは、要求とその引数を連続した複数のパケットでなく 1 つのパケットで受け取ることが想定されています。

UNIX での実装と同様に、NetX FTP ではファイル形式について以下の制約が前提とされます。

- ファイルタイプ：バイナリ
- ファイル形式：Nonprint のみ
- ファイル構造：File Structure のみ

4.3.19.4 アプリケーションへの NetX/NetX Duo FTP サーバーモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo FTP サーバーモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユー

ザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo FTP サーバーモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo FTP サーバーモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_dns0 NetX FTP Server	Threads	New Stack> X-Ware> NetX> Protocols> NetX FTP Server
g_dns0 NetX Duo FTP Server	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo FTP Server

次の図に示すように、NetX/NetX Duo FTP サーバーモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

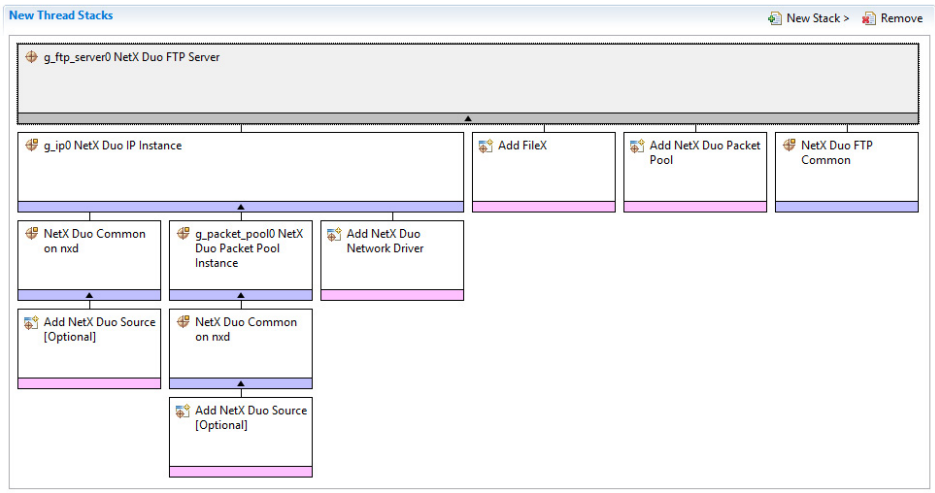


図 354:NetX/NetX Duo FTP サーバーモジュールのスタック

上記のスタックで、NetX ネットワークドライバー (NetX Duo スタックの場合は NetX Duo ネットワークドライバー) はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上

に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

また、上記のスタックで FileX スタックはまだ設定されていません。FileX モジュールには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- FileX スタブ
- ブロックメディア上の FileX（sf_block_media_ram) 上のブロックメディアフレームワークに実装）
- USB 大容量記憶装置上の FileX（USBX ホストクラス大容量記憶装置上に実装）

4.3.19.5 NetX/NetX Duo FTP サーバーモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo FTP サーバーモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo FTP サーバーモジュールの構成設定

ISDE Property	Value	Description
Internal Thread Priority	16	Internal thread priority selection
Internal thread time slicing interval (ticks)	2	Internal thread time slicing interval selection
Maximum clients to serve simultaneously	4	Maximum number of clients allowed
Control window size (bytes)	400	Control window size selection

ISDE Property	Value	Description
Data window size (bytes)	2048	Data window size selection
Duration internal services will suspend for (seconds)	1	Duration internal services will suspend for selection
Maxixum username length (bytes)	20	Maximum username length selection
Maximum password length (bytes)	20	Maximum password length selection
Duration allowed with no activity (seconds)	240	Duration allowed with no activity witection
Socket retransmit timeout (seconds)	2	Duration for initial timeout initialtection
Maximum queued transmit packets	20	Maximum queued transmit selection
Number of socket retransmissions	10	Maximum retries per selection
Binary left shift as multiplier for retry duration	1	Binary left shift as multiplier for retry duration selection
Name	g_ftp_server0	Module name
Internal thread stack size (bytes)	4096	Internal thread stack size stacks selection
Name of Login Function	ftp_login	Name of login function selection
Name of Logout Function	ftp_logout	Name of logout function selection
Name of generated initialization function	ftp_server_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、異なる MAC アドレスの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo FTP サーバーのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
Default Gateway Address (use commas for separation)	0,0,0,0	Default gateway address selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection

ISDE Property	Value	Description
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function
Link status change callback	NULL	Link status change callback selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo FTP 共通インスタンスの構成設定

ISDE Property	Value	Description
FileX support	Enabled, Disabled Default: Enabled	FileX support selection
Control Type of Service	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost Default: Normal	Control type of service selection
Data Type of Service	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost Default: Normal	Data type of service selection
Fragmentation option	Don't fragment, Fragment okay Default: Don't fragment	Fragment option selection
Time to live	128	Time to live selection
Duration between client inactivity check (seconds)	60	Duration between client inactivity check selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection

ISDE Property	Value	Description
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo FTP サーバーモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo FTP サーバーモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin Selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.19.6 アプリケーションでの NetX/NetX Duo FTP サーバーモジュールの使用

一般的なアプリケーションで NetX/NetX Duo FTP サーバーモジュールを使用する際の手順は次のとおりです。

1) nx_ftp_server_create（NetX Duo システムでは nxd_ftp_server_create）API を使用して FTP サーバーを作成します。
2) nx_ftp_server_start API を使用して FTP サーバーを起動します。
3) nx_ftp_client_create API を使用して FTP クライアントを作成します。
4) nx_ftp_client_connect API を使用して、FTP サーバーに接続します。
5) nx_ftp_client_open API を使用してファイルを開きます。
6) nx_ftp_client_read API を使用してファイルを読み取ります。
7) nx_ftp_client_close API を使用してファイルを閉じます。

次の図は、一般的な手順を示した通常の動作フロー図です。

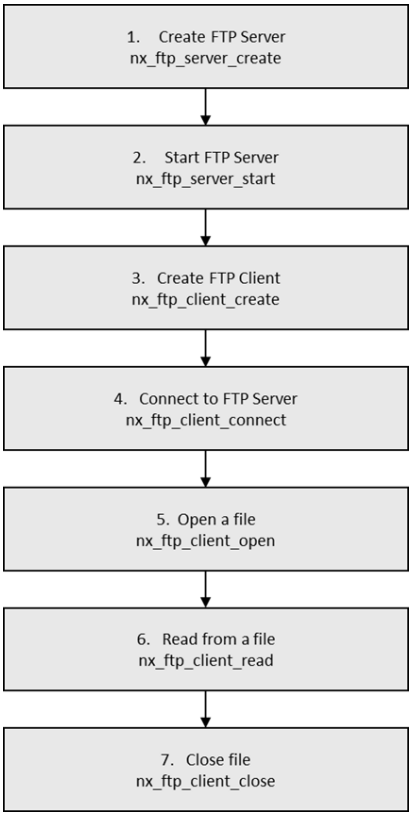


図 355:一般的な NetX/NetX Duo FTP サーバーモジュールアプリケーションのフロー図

4.3.20 NetX/NetX Duo HTTP クライアント

ハイパーテキストトランスファープロトコル（HTTP）は、Web 上のコンテンツを転送するために設計されたプロトコルです。HTTP は、信頼性の高い伝送制御プロトコル（TCP）サービスを利用してコンテンツ転送機能を実行するシンプルなプロトコルです。Web 上のすべての動作には HTTP プロトコルが利用されます。

注 :NetX Duo ™ HTTP クライアントは IPv4 と IPv6 の両方のネットワークに対応していますが、NetX ™ HTTP クライアントは IPv4 通信のみをサポートします。IPv6 によって HTTP プロトコルが直接的な影響を受けることはありませんが、IPv6 に対応するためには、NetX HTTP クライアントにいくつかの違いが必要です。それらについてはこのドキュメントで説明します。

4.3.20.1 NetX/NetX Duo HTTP クライアントモジュールの特長

- 以下を行うためのハイレベル API を提供します。
  - HTTP クライアントインスタンスの作成および削除
  - HTTP サーバーへの GET および PUT 要求の送信
- NetX HTTP は RFC 1945「ハイパーテキストトランスファープロトコル /1.0」、RFC 2581「TCP 輻輳制御」、RFC 1122「インターネットホストに関する要件」、および関連する RFC に準拠しています。

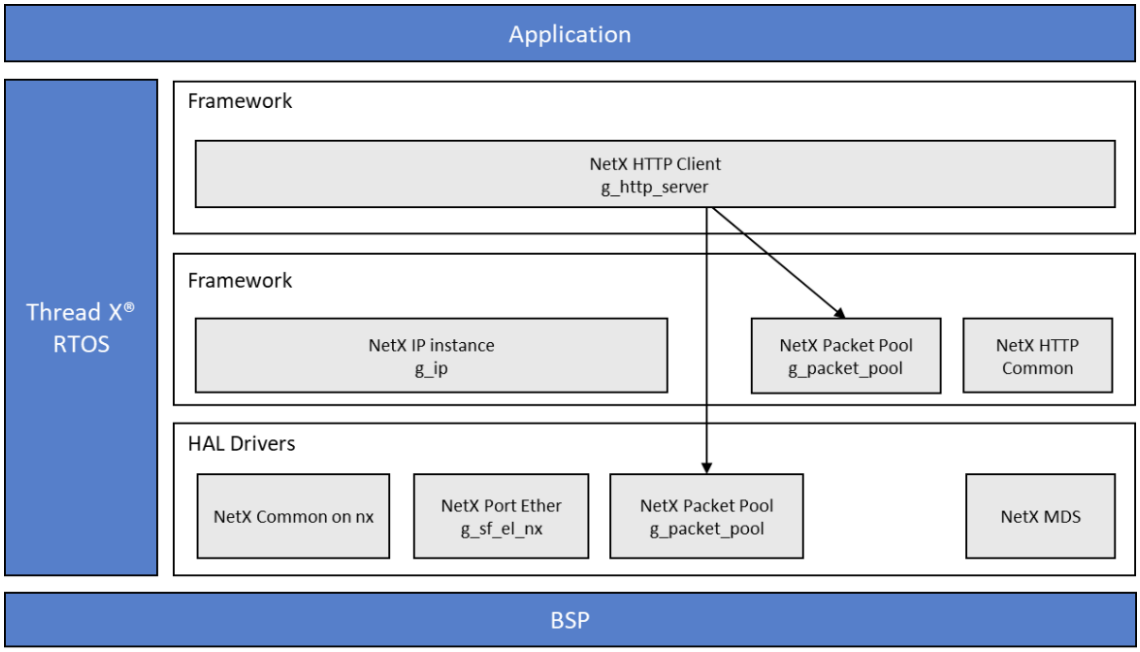


図 356:NetX/NetX Duo HTTP クライアントモジュールのブロック図

注 : 上の図で、NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.20.2 NetX/NetX Duo HTTP クライアントモジュールの API の概要

NetX HTTP クライアントモジュールでは、作成、削除、GET および PUT のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX/NetX Duo HTTP クライアントモジュールの API の要約

Function Name	Example API Call and Description
nx_http_client_create	nx_http_client_create(&my_client, "my client", &ip_0, &pool_0, 100); Create an HTTP Client Instance.
nx_http_client_delete	nx_http_client_delete(&my_client); Delete an HTTP Client instance.
nx_http_client_get_start	nx_http_client_get_start(&my_client, IP_ADDRESS(1,2,3,5), "/TEST.HTM", NX_NULL, 0, "myname", "mypassword", 1000); Start an HTTP GET request (IPv4 only).
nxd_http_client_get_start**	nxd_http_client_get_start(&my_client, &server_ip_address, "/TEST.HTM", NX_NULL, 0, "myname", "mypassword", 1000); Start an HTTP GET request (IPv4 or IPv6)
nx_http_client_get_packet	nx_http_client_get_packet(&my_client, &next_packet, 1000); Get next resource data packet.

Function Name	Example API Call and Description
nx_http_client_put_start	nx_http_client_put_start(&my_client, IP_ADDRESS(1, 2, 3, 5), “/TEST.HTM”, “myname”, “mypassword”, 20, NX_WAIT_FOREVER); Start an HTTP PUT request (IPv4 only).
nxd_http_client_put_start**	nxd_http_client_put_start(&my_client, &server_ip_address, "/client_test.htm", "name", "password", 103, 50); Start an HTTP PUT request (IPv4 or IPv6)
nx_http_client_put_packet	nx_http_client_put_packet(&my_client, packet_ptr, NX_WAIT_FOREVER); Send next resource data packet.
nx_http_client_set_connect_port**	nx_http_client_set_connect_port(&g_http_client0, 81); Connect to the HTTP server port on the specified port. Intended for situations where the Client must use another port beside 80.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :** この API は、NetX Duo HTTP クライアントでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

ステータス戻り値

Name	Description
NX_SUCCESS	Successful HTTP function
NX_CALLER_ERROR**	Invalid caller of the service
NX_PTR_ERROR**	Invalid HTTP, ip_ptr, or packet pool pointer
NX_INVALID_PORT**	Invalid port input

Name	Description
NX_HTTP_POOL_ERROR	Invalid payload size in packet pool
NX_HTTP_NOT_READY	HTTP Client not in ready state
NX_HTTP_PASSWORD_TOO_LONG	Password exceeded expected length
NX_HTTP_AUTHENTICATION_ERROR	Invalid name and/or password
NX_HTTP_FAILED	HTTP client error communicating with the HTTP server
NX_HTTP_GET_DONE	HTTP client get packet operation is complete
NX_HTTP_BAD_PACKET_LENGTH	Invalid packet received - length incorrect
NX_HTTP_INCOMPLETE_PUT_ERROR	Server responds before PUT is complete
NX_HTTP_REQUEST_UNSUCCESSFUL_CODE	Received an error code instead of 2xx from server
NX_HTTP_PASSWORD_TOO_LONG	Password exceeded expected length
NX_HTTP_USERNAME_TOO_LONG	Username exceeded expected length
NX_SIZE_ERROR	Invalid total size of resource in PUT request
NX_INVALID_PACKET	Invalid TCP packet; not enough room for packet header

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

*注 :** これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX と NetX Duo でのエラーチェックサービスの詳細については、それぞれ『NetX User's Guide for the Renesas Synergy Platform』または『NetX Duo User's Guide for the Renesas Synergy Platform』を参照してください。*

4.3.20.3 NetX/NetX Duo HTTP クライアントモジュールの動作の概要

NetX HTTP クライアントモジュールは、このインスタンスが NetX の動作を実行し、NetX ライブラリ内の TCP サービスを有効にする IP インスタンスを作成します。これが、ポート 80 で受信待ちをしているサーバーとの間で HTTP メッセージを送受信するために、HTTP クライアントインスタンスと TCP ソケットを作成します。HTTP クライアントにはパケットプールが必要であり、モジュールが IP デフォルトパケットプール（g_packet_pool0),）を共有することによって提供するか、新規に作成します。パケットペイロードの下限は、HTTP クライアントインスタンスの [Minimum packet size] プロパティによって設定されます。このパケットプールはパケットの送信のためにのみ HTTP クライアントに使用されるので、パケットプールのサイズとペイロードは HTTP クライアントから送信されるパケットの想定されるサイズと数に基づいて最適化できます。

NetX Duo HTTP クライアントでは IPv4 と IPv6 の両方の接続がサポートされます。HTTP クライアントがサーバーへの接続に IPv6 を使用する必要がある場合は、NetX Duo ソース要素で [NetX Duo IPv6 Support] プロパティが有効になっていることを確認します。基礎となる ICMPv6 プロトコルのために、ICMPv6 チェックサムの計算を有効にする

ことも必要になる場合があります。その場合は、NetX Duo ソース要素の [Checksum computation support on transmitted ICMPv6 packets] プロパティと [Checksum computation support on received ICMPv6 packets] プロパティを有効に設定します（ホストハードウェアで自動的に ICMPv6 チェックサムが計算される場合、これらは無効のままで構いません）。IP インスタンス要素でクライアントホストの [IPv6 Global Address] が設定されていることを確認します。IPv6 の基礎となるプロトコルが必要とする IPv6 および ICMPv6 サービスを有効にするために必要な処理が NetX Duo によって実行されます。

HTTP クライアントは、有効な IP アドレスが設定されると、PUT および GET 要求を実行できるようになります。パケットをアップロードするには、nx_http_client_put_start** サービスを使用します。このサービスはサーバーの IP アドレスを入力として受け取るので、HTTP クライアントはサーバーに接続できるようになります。アップロードするデータのパケットが複数になる場合、アプリケーションはすべてのデータがアップロードされるまで nx_http_client_put_packet サービスを使用します。サーバーからデータをダウンロードするには、nx_http_client_get_start サービスを使用します。これには、HTTP クライアントがサーバーに接続できるようサーバーの IP アドレスが必要です。ダウンロードするデータのパケットが複数になる場合、アプリケーションはすべてのデータがダウンロードされ、NX_HTTP_GET_DONE ステータスが返されることによってそのことが知らされるまで、nx_http_client_get_packet サービスを使用します。

NetX Duo HTTP クライアントモジュールでは、アプリケーションは IPv4 接続に nxd_http_client_put_start サービスを、IPv4 または IPv6 接続に nxd_http_client_get_start サービスを使用できます。Net Duo HTTP クライアントでは、nx_http_client_put_start および nx_http_client_get_start サービスも使用できます。nx_http_client_put_packet サービスと nx_http_client_get_packet サービスは HTTP サーバーの IP アドレスを必要としないため、これらのサービスに Duo 版の API はありません。

NetX Duo HTTP クライアントでは、HTTP クライアントがデフォルトのポート 80 以外のポートで HTTP サーバーに接続する必要がある場合、nx_http_client_set_connect_port サービスを使用できます。

HTTP サーバーのレスポンス

HTTP サーバーはクライアントコマンドを処理すると、次の表に示す 3 桁の数値によるステータスコードを含む ASCII 文字列のレスポンスを返します。HTTP クライアントソフトウェアはこの数値レスポンスを使用して、操作が成功したか失敗したかを判断します。

クライアントコマンドに対する HTTP サーバーのさまざまなレスポンス

Numeric Field	Meaning
200	Request was successful
400	Request was not formed properly
401	Unauthorized request, client needs to send authentication
404	Specified resource in request was not found
500	Internal HTTP server error
501	Request not implemented by HTTP server
502	Service is not available

たとえば、ファイル test.htm を PUT するクライアント要求が成功すると、メッセージ HTTP/1.0 200 OK のレスポンスが返されます。

NetX/NetX Duo HTTP クライアントモジュールの動作に関する重要な注意事項と制限事項

- HTTP クライアントのパケットプールには、HTTP ヘッダー全体を保持するのに十分な大きさが必要です。
- クライアントの TCP ソケットを削除してサーバーから切断するまでの待機時間を決定するオプションは、[Operation Timeout] プロパティで設定します。他のすべての HTTP クライアントサービスに関する待機オプションは、API で設定します。
- GET と PUT の開始サービスはどちらもリソースを必要とし、入力としてユーザー名とパスワードを必要とします。それぞれの最大サイズは、NetX HTTP 共通要素の [Maximum resource name length]、[Maximum username length]、[Maximum password length] の各プロパティで設定します。GET および PUT 操作が完了すると、HTTP クライアントはサーバーとの接続を切断します。
- HTTP クライアントの TCP ソケット受信ウィンドウは、[TCP socket window size] プロパティで設定します。これは TCP プロトコルで、受信済みデータのパケットの確認が保留されているときに、ソケットの片側が相手側に対して追加のデータを送信しないように通知するために使用されます。
- NetX および NetX Duo の HTTP プロトコルでは HTTP 1.0 標準が実装されています。1.1 はサポートされません。制約は次のとおりです。
  - 継続した接続はサポートされません
  - 要求のパイプライン化はサポートされません
  - コンテンツ圧縮はサポートされません
  - TRACE、OPTIONS、および CONNECT 要求はサポートされません
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.20.4 アプリケーションへの NetX/NetX Duo HTTP クライアントモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo HTTP クライアントモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo HTTP クライアントモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo HTTP クライアントモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_dns0 NetX HTTP Client	Threads	New Stack> X-Ware> NetX> Protocols> NetX HTTP Client

Resource	ISDE Tab	Stacks Selection Sequence
g_dns0 NetX Duo HTTP Client	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo HTTP Client

次の図に示すように、NetX/NetX Duo HTTP クライアントモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

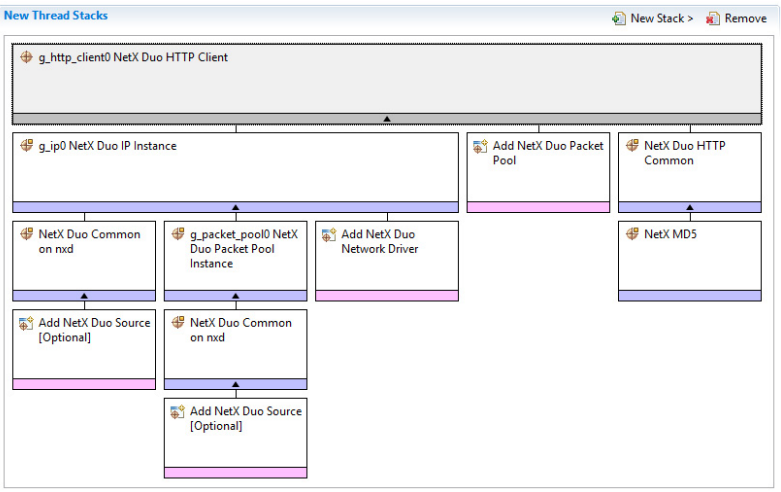


図 357:NetX/NetX Duo HTTP クライアントモジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.20.5 NetX/NetX Duo HTTP クライアントモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo HTTP クライアントモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo HTTP クライアントモジュールの構成設定

ISDE Property	Value	Description
Minimum packet size (bytes)	300	Minimum packet size selection
Operation timeout (seconds)	10	Operation timeout selection
*Maximum password length (bytes)	20	Maximum password length selection
*Maximum username length (bytes)	20	Maximum username length selection
Name	g_http_client0	Module name
TCP socket window size (bytes)	1024	TCP socket window size selection
Name of generated initialization function	http_client_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注 : NetX Duo では使用できないプロパティを表します。*

ローレベルモジュールのデフォルト以外の設定が望ましいこともあります。たとえば、イーサネットペリフェラル用に異なるピンの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo HTTP クライアントのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
Default Gateway Address (use commas for separation)	0,0,0,0	Default gateway address selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection

ISDE Property	Value	Description
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function
Link status change callback	NULL	Link status change callback selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo HTTP 共通インスタンスの構成設定

ISDE Property	Value	Description
Type of Service	Normal, Minimum delay, Maximum data, Maximum reliabililty, Minimum cost Default: Normal	Type of service UDP requests selection
Fragmentation option	Don't fragment, Fragment okay Default: Don't fragment	Fragment option selection
Time to live	128	Time to live selection
MD5 Support	Enable, Disable Default: Disable	MD5 support selection
Maximum resource name length (bytes)	40	Packet queue depth selection
**Maximum password length (bytes)	20	Maximum password length selection
**Maximum username length (bytes)	20	Minimum username length selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo 共通インスタンスの構成設定表

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection

ISDE Property	Value	Description
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX MD5 インスタンスの構成設定

ISDE Property	Value	Description
No configurable properties()	-	-

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo HTTP クライアントモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo HTTP クライアントモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin Selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin

Property	Value	Description
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.20.6 アプリケーションでの NetX/NetX Duo HTTP クライアントモジュールの使用

一般的なアプリケーションで NetX/NetX Duo HTTP クライアントモジュールを使用する際の手順は次のとおりです。

1) nx_ip_status_check API を使用して、有効な IP アドレスとネットワークドライバーの初期化を待ちます。
2) nx_http_client_put_start API を使用して、HTTP サーバーにデータをアップロードします。IPv6 接続の場合は、NetX Duo HTTP クライアントの nxd_http_client_put_start API（IPv4 接続にも使用可能）を使用します。
3) nx_http_client_get_start API を使用して、HTTP サーバーからデータをダウンロードします。IPv6 接続の場合は、NetX Duo HTTP クライアントの nxd_http_client_get_start API（IPv4 接続にも使用可能）を使用します。
4) nx_http_client_delete API で HTTP クライアントを削除します（パケットプールは、アプリケーション内の他の場所で使用されない限り削除して構いません）。

次の図は、一般的な手順を示した通常の動作フロー図です。

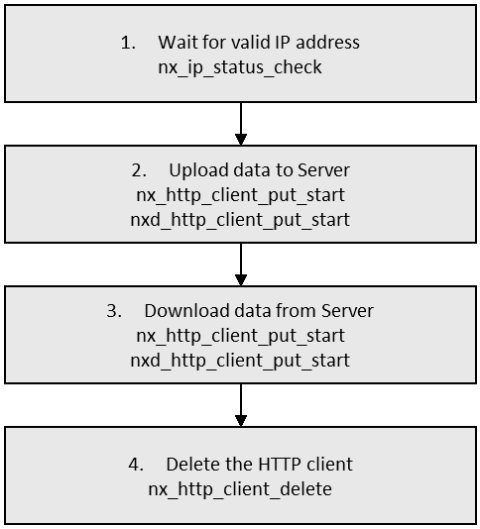


図 358:一般的な NetX/NetX Duo HTTP クライアントモジュールアプリケーションのフロー図

4.3.21 NetX/NetX Duo HTTP サーバー

ハイパーテキストトランスファープロトコル（HTTP）は、信頼できる伝送制御プロトコル（TCP）サービスを使用してコンテンツ転送機能を実行します。Web 上のすべての動作には HTTP プロトコルが利用されます。NetX™ Duo HTTP サーバーは IPv4 と IPv6 の両方のネットワークに対応していますが、NetX™ HTTP サーバーは IPv4 通信のみをサポートします。IPv6 によって HTTP プロトコルが直接的な影響を受けることはありませんが、IPv6 に対応するためには、NetX HTTP サーバーにいくつかの違いが必要です。それらについてはこのドキュメントで説明します。

注 :NetX Duo HTTP サーバーモジュールは、内部処理を除けば HTTP セッションの応用、設定、実行が NetX HTTP サーバーモジュールとほぼ同じです。相違点は、このドキュメントで説明しています。

4.3.21.1 NetX/NetX Duo HTTP サーバーモジュールの特長

- RFC（Request for Comments）1945「ハイパーテキストトランスファープロトコル /1.0」、RFC 2581「TCP 輻輳制御」、RFC 1122「インターネットホストに関する要件」、および関連する RFC に準拠
- マルチパートサポート
- 基本認証とダイジェスト認証のサポート
- いくつかの重要な関数のコールバックのサポート：
  - HTTP 認証コールバック
  - HTTP 要求通知コールバック
  - HTTP 無効なユーザー名 / パスワードコールバック
  - HTTP GMT 日付ヘッダー挿入コールバック
  - HTTP キャッシュ情報取得コールバック

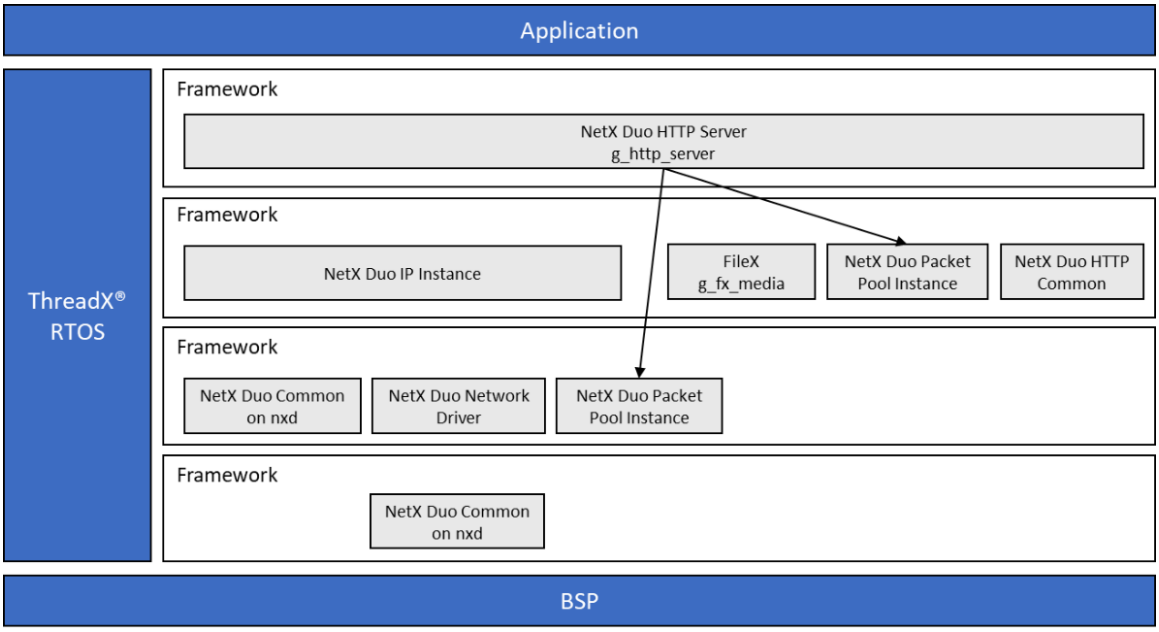


図 359:NetX/NetX Duo HTTP サーバーモジュールのブロック図

注 : 上の図で、FileX および NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.21.2 NetX/NetX Duo HTTP サーバーモジュールの API の概要

NetX HTTP サーバーモジュールでは、レスポンスパケットの作成、削除、生成、レスポンス送信、受信パケットからの情報の取得のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX/NetX Duo HTTP サーバーモジュールの API の要約

Function Name	Example API Call and Description
nx_http_server_cache_info_callback_set	nx_http_server_cache_info_callback_set(& my_server, cache_info_get); Set callback to retrieve age and last modified date of specified URL.

Function Name	Example API Call and Description

nx_http_server_callback_data_send	nx_http_server_callback_data_send(server_ptr, "HTTP/1.0 200 \r\nContent-Length: 103\r\nContent-Type: text/html\r\n\r\n",63);
	
	nx_http_server_callback_data_send(server_ptr, "<HTML>\r\n<HEAD><TITLE>NetX HTTP Test </TITLE></HEAD>\r\n <BODY>\r\n<H1>NetX Test Page </H1>\r\n</BODY>\r\n</HTML>\r\n", 103);
	
	Send HTTP data from callback function.

nx_http_server_callback_generate_response_header	nx_http_server_callback_generate_response_header (server_ptr, &packet_ptr, status_code, content_length, content_type, additional_header); Create response header in callback functions.
nx_http_server_callback_packet_send	nx_http_server_callback_packet_send(server_ptr, packet_ptr); Send an HTTP packet from an HTTP callback.
nx_http_server_callback_response_send	nx_http_server_callback_response_send(server_ptr,"HTTP/1.0 404 ", "NetX HTTP Server unable to find file: ", resource); Send response from callback function.
nx_http_server_content_get	nx_http_server_content_get(server_ptr, packet_ptr, 0, my_buffer, 100, &actual_size); Get content from the request.
nx_http_server_content_get_extended	nx_http_server_content_get_extended(server_ptr, packet_ptr, 0, my_buffer, 100, &actual_size); Get content from the request; supports empty (zero Content Length) requests.

Function Name	Example API Call and Description
nx_http_server_content_length_get	nx_http_server_content_length_get(packet_ptr); Get length of content in the request. Length is the status return value. A length of zero indicates an error.
nx_http_server_content_length_get_extended	nx_http_server_content_length_get_extended(packet_ptr, &content_length); Get length of content in the request; supports empty (zero Content Length) requests.
nx_http_server_create	nx_http_server_create(&my_server, "my server", &ip_0, &ram_disk, stack_ptr, stack_size, &pool_0, my_authentication_check, my_request_notify); Create an HTTP Server instance.
nx_http_server_delete	nx_http_server_delete(&my_server); Delete an HTTP Server instance.
nx_http_server_get_entity_content	nx_http_server_get_entity_content(server_ptr, &packet_ptr, &offset, &length); Return size and location of entity content in URL.
nx_http_server_get_entity_header	nx_http_server_get_entity_header(server_ptr, &packet_ptr, entity_header_buffer, buffer_size); Extract URL entity header into specified buffer.
nx_http_server_gmt_callback_set	nx_http_server_gmt_callback_set(&my_server, gmt_get); Set callback to retrieve GMT date and time.

Function Name	Example API Call and Description
**nx_http_server_invalid_userpassword_notify_set	nx_http_server_invalid_userpassword_notify_set(&my_server, invalid_username_password_callback); Set callback for when invalid username and password is received in a Client request.
nx_http_server_mime_maps_additional_set	nx_http_server_mime_maps_additional_set(&my_server, &my_mime_maps[0], 2); Define additional mime maps for HTML.
nx_http_server_packet_content_find	nx_http_server_packet_content_find(server_ptr, packet_ptr, &content_length); Extract content length in HTTP header and set pointer to start of content data.
nx_http_server_packet_get	nx_http_server_packet_get(server_ptr, &packet_ptr); Receive client packet directly.
nx_http_server_param_get	nx_http_server_param_get(packet_ptr, 0, param_destination, 30); Get parameter from the request.
nx_http_server_query_get	nx_http_server_query_get(packet_ptr, 0, query_destination, 30); Get query from the request.
nx_http_server_start	nx_http_server_start(&my_server); Start the HTTP Server.
nx_http_server_stop	nx_http_server_stop(&my_server); Stop the HTTP Server.

Function Name	Example API Call and Description
nx_http_server_type_get	nx_http_server_type_get(server_ptr, resource_name, type_string); Extract HTTP type, for example, text/plain from header. Type is returned in the status return. A value of zero indicates an error.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :** この API は、NetX Duo HTTP サーバーでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

ステータス戻り値

Name	Description
NX_SUCCESS	Successfully performed function
NX_PTR_ERROR**	Invalid pointer input
NX_CALLER_ERROR **	Invalid caller of service
NX_HTTP_DATA_END	End of request content
NX_HTTP_TIMEOUT	HTTP Server timeout in getting next packet of content
NX_CALLER_ERROR	Invalid caller of this service
NX_HTTP_INCOMPLETE_PUT_ERROR	Improper HTTP header format
NX_HTTP_POOL_ERROR	Packet payload of pool is not large enough to contain complete HTTP request
NX_HTTP_BOUNDARY_ALREADY_FOUND	Content for the HTTP server internal multipart markers is already found
NX_HTTP_NOT_FOUND	Entity header field or client request parameter or multipart component not found
NX_HTTP_IMPROPERLY_TERMINATED_PARAM	Client request parameter not properly terminated
NX_HTTP_NO_QUERY_PARSED	Server unable to find query in client request

Name	Description
NX_HTTP_TIMEOUT	No packet received in the specified wait interval
NX_HTTP_ERROR	Internal HTTP Server error
NX_HTTP_SERVER_DEFAULT_MIME	No matching extension type found. Return the default MIME type. Not an error status.
NX_SIZE_ERROR	Invalid total size of resource in PUT request
NX_INVALID_PACKET	Invalid TCP packet; not enough room for packet header

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

*注 :** これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX と NetX Duo でのエラーチェックサービスの詳細については、それぞれ『NetX User's Guide for the Renesas Synergy Platform』または『NetX Duo User's Guide for the Renesas Synergy Platform』を参照してください。*

4.3.21.3 NetX/NetX Duo HTTP サーバーモジュールの動作の概要

NetX HTTP サーバーモジュールは、NetX 動作を実行して、NetX ライブラリ内の TCP サービスに対してインスタンスを有効にする IP インスタンスを作成します。次に、クライアント要求をポート 80 で受信待ちするために、HTTP サーバーインスタンスと TCP ソケットを作成します。HTTP サーバーにはパケットプールが必要であり、モジュールが IP デフォルトパケットプール（g_packet_pool0）を共有することによって提供するか、新規に作成します。パケットペイロードの下限は、HTTP サーバーモジュールの [Minimum size of packets in pool] プロパティによって設定されます。このパケットプールはパケットの送信のためにのみ HTTP サーバーに使用されるので、パケットプールのサイズとペイロードは HTTP サーバーから送信されるパケットの想定されるサイズと数に基づいて最適化できます。

NetX Duo HTTP サーバーは、IPv4 と IPv6 の両方の接続に対応しています。HTTP サーバーに対して IPv6 での接続を求めるクライアントがある場合は、NetX Duo ソース要素で [NetX Duo IPv6 Support] プロパティが有効になっていることを確認します。基礎となる ICMPv6 プロトコルのために、ICMPv6 チェックサムの計算を有効にすることが必要になる場合があります。その場合は、NetX Duo ソース要素の [Checksum computation support on transmitted ICMPv6 packets] プロパティと [Checksum computation support on received ICMPv6 packets] プロパティを [Enabled] に設定します（ホストハードウェアで自動的に ICMPv6 チェックサムが計算される場合、これらは無効のままで構いません）。IP インスタンス要素でクライアントホストの [IPv6 Global Address] が設定されていることを確認します。その後、IPv6 の基礎となるプロトコルが必要とする IPv6 および ICMPv6 サービスを有効にするために必要な処理が NetX Duo によって実行されます。

また、HTTP サーバーは、FileX® 組み込みファイルシステムと連携するように設計されています。

HTTP サーバーのレスポンス

HTTP サーバーはクライアントコマンドを処理すると、3 桁の数値によるステータスコードを含む ASCII 文字列のレスポンスを返します。HTTP クライアントソフトウェアはこの数値レスポンスを使用して、操作が成功したか失敗したかを判断します。次のリストに、クライアントコマンドに対する HTTP サーバーのさまざまなレスポンスを示します。

クライアントコマンドに対する HTTP サーバーのレスポンス

Numeric Field	Meaning
200	Request was successful
400	Request was not formed properly
401	Unauthorized request, client needs to send authentication
404	Specified resource in request was not found
500	Internal HTTP Server error
501	Request not implemented by HTTP Server
502	Service is not available

たとえば、ファイル test.htm を PUT するクライアント要求が成功すると、メッセージ HTTP/1.0 200 OK のレスポンスが返されます。

HTTP 認証

HTTP 認証はオプションであり、すべての Web 要求で必要とされるわけではありません。認証には基本認証とダイジェスト認証の 2 種類があります。基本認証は、多くのプロトコルで使用される名前とパスワードを使った認証に相当します。HTTP 基本認証では、名前とパスワードが連結され、base64 フォーマットでエンコードされます。基本認証の主な欠点は、要求で名前とパスワードが隠されることなく送信されるため、名前とパスワードが盗まれやすいことです。ダイジェスト認証は、要求で名前とパスワードを送信しないことによってこの問題に対処しています。代わりにアルゴリズムを使用して、名前、パスワードなどの情報から 128 ビットのキー（ダイジェスト）を生成します。NetX HTTP サーバーは、標準的な MD5 ダイジェストアルゴリズムをサポートしています。

HTTP サーバー認証コールバックにより、要求されたリソースに認証が必要かどうかを確認できます。認証が必要であり、クライアント要求に適切な認証情報が含まれていない場合は、必要な認証の種類とともに HTTP/1.0 401 Unauthorized レスポンスがクライアントに送信されます。その場合、クライアントは適切な認証情報を含む新しい要求を作成する必要があります。

HTTP 認証コールバック

HTTP サーバー認証コールバックルーチンは、HTTP サーバースレッドの [Name of Authentication Checking Function] プロパティで指定されます。この関数は、各 HTTP クライアント要求の開始時にコールされます。

コールバックルーチンは、リソースに関連付けられたユーザー名、パスワード、realm 文字列を NetX HTTP サーバーに提供し、必要な認証タイプを返します。認証が不要なリソースの場合、認証コールバックは値 NX_HTTP_DONT_AUTHENTICATE. を返します。指定されたリソースで基本認証が必要な場合、このルーチンは NX_HTTP_BASIC_AUTHENTICATE. を返します。MD5 ダイジェスト認証が必要な場合、コールバックルーチンは NX_HTTP_DIGEST_AUTHENTICATE. を返します。

認証コールバックルーチンのフォーマットは、次のように定義されます。

```
UINT nx_http_server_authentication_check(NX_HTTP_SERVER *server_ptr, UINT reques
t_type, CHAR *resource, CHAR **name, CHAR **password, CHAR **realm);
```

入力パラメータは次のように定義されます。

入力パラメータの定義

Parameter	Meaning
request_type	Specifies the HTTP Client request, valid requests are defined as:
	NX_HTTP_SERVER_GET_REQUEST
	NX_HTTP_SERVER_POST_REQUEST
	NX_HTTP_SERVER_HEAD_REQUEST
	NX_HTTP_SERVER_PUT_REQUEST
	NX_HTTP_SERVER_DELETE_REQUEST
resource	Specific resource requested.
name	Destination for the pointer to the required username.
password	Destination for the pointer to the required password.
realm	Destination for the pointer to the realm for this authentication.

認証コールバックルーチンから NX_HTTP_DONT_AUTHENTICATE が返された場合、名前、パスワード、realm ポインタは使用されません。HTTP サーバーの開発者は、ユーザー名とパスワードの最大サイズ（NetX HTTP 共通の [Maximum username length] および [Maximum password length] プロパティで定義）が、認証コールバックで指定されるユーザー名とパスワードにとって十分な大きさとなるようにする必要があります。どちらもデフォルトのサイズは 20 文字です。

HTTP サーバー要求通知コールバック

要求コールバックが指定されている場合（NetX HTTP サーバーモジュールの [Name of Request Notify Callback Function] プロパティ）、クライアント要求の認証と検証がエラーなしで完了した後、NetX HTTP サーバーは指定された関数に要求を転送します。コールバックは、クライアント要求をそれ以上処理する必要がないか（戻りステータス NX_HTTP_CALLBACK_COMPLETED),）、コールバック処理でエラーが発生したか（ステータスがゼロ以外）、または処理が正常に完了して HTTP サーバーはクライアント要求の処理を続行する必要があるかを（戻りステータスによって）示す必要があります。このコールバックのフォーマットは次のとおりです。

```
UINT        request_notify(NX_HTTP_SERVER *server_ptr, UINT request_type, CHAR *
resource,
 NX_PACKET *packet_ptr)
```

要求通知コールバックを無効にするには、[Name of Request Notify Callback Function] プロパティを NULL に設定します。

HTTP 無効なユーザー名 / パスワードコールバック

NetX HTTP サーバーモジュールにオプションとして含まれる無効なユーザー名 / パスワードコールバックは、HTTP サーバーが受け取ったクライアント要求に含まれるユーザー名とパスワードの組み合わせが無効な場合に呼び出されます。無効なユーザー名/パスワードコールバック関数を設定するには、nx_http_server_invalid_user_password_set サービスを使用します。このサービスは、NetX Duo HTTP サーバーモジュールでは NXD_ADDRESS *client_ip_address,、NetX HTTP サーバーモジュールでは ULONG client_ip_address. を受け取ります。

HTTP GMT 日付ヘッダー挿入コールバック

NetX HTTP サーバーは、レスポンスメッセージに日付ヘッダーを挿入するオプションのコールバックをサポートしています。このコールバックは、サーバーがクライアントの PUT 要求または GET 要求に応答するときに呼び出されます。GMT コールバックを設定するには、NetX HTTP サーバースレッドを開始する前に nx_http_server_gmt_callback サービスを使用します。

HTTP キャッシュ情報取得コールバック

NetX HTTP サーバーには、特定のリソースの最長有効期間と日付を HTTP アプリケーションから要求するオプションのコールバックがあります。この情報は、HTTP サーバーがクライアントの GET 要求へのレスポンスとしてページ全体を送信するかどうかを決定するために使用されます。クライアント要求の if modified since が見つからない場合、またはキャッシュ取得コールバックから返された最終変更日と一致しない場合は、ページ全体が送信されます。キャッシュコールバック関数を設定するには、nx_http_server_cache_info_callback_set サービスを使用します。

HTTP マルチパートサポート

多目的インターネットメール拡張（MIME）は、当初は SMTP プロトコル用でしたが、HTTP にも利用が拡張されました。MIME を使用すると、1 つのメッセージに複数のメッセージタイプ（たとえば、image/jpg と text/plain）を混在させることができます。NetX HTTP サーバーには、クライアントからの MIME を含む HTTP メッセージ内のコンテンツタイプを特定するサービスが追加されています。マルチパートサポートを有効にするには、NetX HTTP サーバーモジュールの [Multipart HTTP requests support] プロパティを有効に設定します。

NetX/NetX Duo HTTP サーバーモジュールの動作に関する重要な注意事項と制限事項

- NetX HTTP サーバーモジュールでは、FileX メディア（ブロックメディアまたは USB 大容量記憶装置）が必要です。HTTP サーバースタック要素がプロジェクトに追加されると、[Add FileX] ボックスがアタッチされます。サーバーが起動する前に、コンフィギュレータがサーバーに合わせて FileX メディアを自動的にセットアップして初期化します。FileX の構成の詳細については、『*FileX ™ User's Guide for the Renesas Synergy ™ Platform*』を参照してください。
- NetX HTTP サーバーには、パケットを送信するためにパケットプールも必要です。IP のデフォルトパケットプールを共有することも、個別のパケットプールを作成することもできます。HTTP サーバーパケットプールの設定の詳細については、「*アプリケーションへの NetX HTTP サーバーモジュールの組み込み*」セクションを参照してください。
- 継続した接続はサポートされません。
- 要求のパイプライン化はサポートされません。
- HTTP サーバーでは、基本認証と MD5 ダイジェスト認証の両方がサポートされますが、MD5-sess はサポートされません。
- コンテンツ圧縮はサポートされません。
- TRACE、OPTIONS、および CONNECT 要求はサポートされません。
- HTTP サーバーに関連付けられたパケットプールには、HTTP ヘッダー全体を保持するのに十分な大きさが必要です。

4.3.21.4 アプリケーションへの NetX/NetX Duo HTTP サーバーモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo HTTP サーバーモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユー

ザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo HTTP サーバーモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo HTTP サーバーモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_http_server0 NetX HTTP Server	Threads	New Stack> X-Ware> NetX> Protocols> NetX HTTP Server
g_http_server0 NetX Duo HTTP Server	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo HTTP Server

次の図に示すように、NetX/NetX Duo HTTP サーバーモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

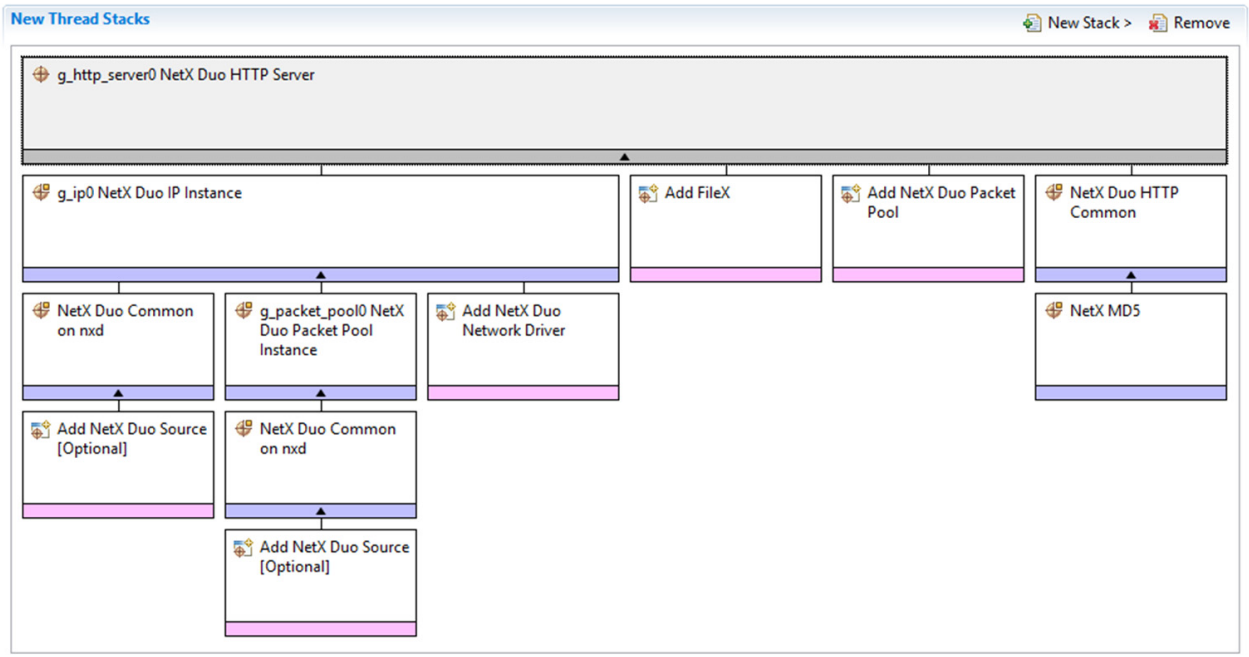


図 360:NetX/NetX Duo HTTP サーバーモジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

また、上記のスタックで FileX スタックはまだ設定されていません。FileX モジュールには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- FileX スタブ
- ブロックメディア上の FileX（sf_block_media_ram) 上のブロックメディアフレームワークに実装）
- USB 大容量記憶装置上の FileX（USBX ホストクラス大容量記憶装置上に実装）

4.3.21.5 NetX/NetX Duo HTTP サーバーモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo HTTP サーバーモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更

しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo HTTP サーバーモジュールの構成設定

ISDE Property	Value	Description
FileX Support	Enable, Disable Default: Enable	FileX support selection
Multipart HTTP requests support	Enable, Disable Default: Disable	Multipart HTTP requests support selection
Internal thread priority	16	Internal thread priority selection
Internal thread time slicing interval (bytes)	NetX Default: 1 NetX Duo Default: 2	Internal thread time slicing interval selection
Server socket window size (bytes)	2048	Server socket window size selection
Server time out (seconds)	10	Server time out selection
Server time out for accept (seconds)	10	Server time out for accept selection
Server time out for disconnect (seconds)	10	Server time out for disconnect selection
Server time out for receive (seconds)	10	Server time out for receive selection
Server time out for send (seconds)	10	Server time out for send selection

ISDE Property	Value	Description
Maximum size of header field (bytes)	256	Maximum size of header field selection
Maximum connections in queue	5	Maximum connections in queue selection
Maximum client user user name length (bytes)	20	Maximum client user password length selection
Maximum client user password length (bytes)	20	Minimum size of packets in pool selection
Minimum size of packets in pool (bytes)	600	Minimum size of packets in pool selection
Name	g_http_server0	Module name
Internal thread stack size (bytes)	4096	Internal thread stack size selection
Name of Authentication Checking Function	authentication_check	Name of Authentication Checking Function selection
Name of Request Notify Callback Function	request_notify	Name of Authentication Checking Function selection
Name of generated initialization function	http_server_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、異なる IP マスクとアドレスの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo HTTP サーバーのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
Default Gateway Address (use commas for separation)	0,0,0,0	Default gateway address selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection

ISDE Property	Value	Description
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function
Link status change callback	NULL	Link status change callback selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo HTTP 共通インスタンスの構成設定

ISDE Property	Value	Description
Type of Service	Normal, Minimum delay, Maximum data, Maximum reliabililty, Minimum cost Default: Normal	Type of service UDP requests selection

ISDE Property	Value	Description
Fragmentation option	Don't fragment, Fragment okay Default: Don't fragment	Fragment option selection
Time to live	128	Time to live selection
MD5 Support	Enable, Disable Default: Disable	MD5 support selection
Maximum resource name length (bytes)	40	Packet queue depth selection
**Maximum password length (bytes)	20	Maximum password length selection
**Maximum username length (bytes)	20	Minimum username length selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX MD5 インスタンスの構成設定

ISDE Property	Value	Description
No configurable properties()	-	-

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo HTTP サーバーモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo HTTP サーバーモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin Selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.21.6 アプリケーションでの NetX/NetX Duo HTTP サーバーモジュールの使用

一般的なアプリケーションで NetX/NetX Duo HTTP サーバーモジュールを使用する際の手順は次のとおりです。

アプリケーションで NetX および NetX Duo HTTP サーバーを初期化するための自動生成コード（common_data.c）

- nx_packet_pool_create API を使用して HTTP パケットプールを作成します。
- nx_ip_create API を使用して IP インスタンスを作成します。
- nx_arp_enable API を使用して ARP を有効にします。
- nx_tcp_enable API を使用して TCP を有効にします。
- nx_http_server_create API を使用して HTTP サーバーを作成します。

ユーザーアプリケーションコード（<thread>_entry.c）

1) nx_ip_status_check API を使用して、有効な IP アドレスを待機します。
2) nx_http_server_start API を使用して HTTP サーバーを起動します。
3) HTTP サーバーに登録されている場合、オプションのコールバックを処理します（認証チェック、要求通知、GMT セット、キャッシュ取得、無効なユーザー名）。
4) nx_http_server_stop API を使用して HTTP サーバーを停止します。
5) nx_http_server_delete API を使用して HTTP サーバーを削除します。

注：サーバーパケットプールを使用しているのがサーバーのみの場合には、そのサーバーパケットプールも nx_packet_pool_delete API を使用して削除できます。

ユーザーは自動生成コードについて心配する必要はありません。ユーザーがスタックの構成後にプロジェクトを生成すると、自動生成コードが含められます。ユーザーが行う必要があるのは、関連するファイル（通常は <thread>_entry.c）にユーザーアプリケーションコードを記述することだけです。

次の図は、一般的な手順を示した通常の動作フロー図です。

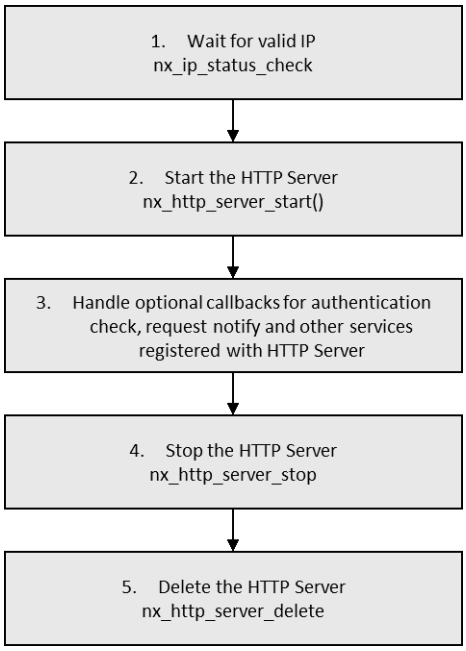


図 361:一般的な NetX/NetX Duo HTTP サーバーモジュールアプリケーションのフロー図

4.3.22 NetX Duo Web HTTP/HTTPs クライアント

NetX Web HTTP/HTTPs クライアントモジュールは、Web 上のコンテンツを転送するためのハイパーテキストトランスファープロトコル（HTTP）にハイレベルな API を提供します。HTTP プロトコルは、伝送制御プロトコル（TCP）サービスを使用してコンテンツ転送機能を実行します。HTTPs は、セキュアな HTTP プロトコルであり、基礎となる TCP 接続を保護するトランスポート層セキュリティ（TLS）プロトコルの上に HTTP を使用したものです。

HTTP/HTTPS クライアントは、NetX Duo IP および NetX Duo パケットプール上に実装されます。NetX Duo IP は、イーサネット /Wi-Fi/ セルラーなどの適切なリンク層ドライバーに接続します。HTTP クライアントは、オプションとしてセキュアな接続を介して HTTP サーバーに接続できます。その場合、NetX Duo TLS 共通で提供されるサービスを使用します。

4.3.22.1 NetX Duo Web HTTP/HTTPs クライアントモジュールの特長

- NetX Web HTTP/HTTPs クライアントは、以下の RFC に準拠しています。
  - RFC1945「ハイパーテキストトランスファープロトコル /1.0」
  - RFC 2616「ハイパーテキストトランスファープロトコル – HTTP/1.1」
  - RFC 2581「TCP 輻輳制御」
  - RFC 1122「インターネットホストに関する要件」および関連する RFC
- HTTP Get/POST/HEAD/PUT/DELETE コマンドをサポートします。これらのコマンドは、NetX Web HTTP/HTTPs クライアント API によって内部的に生成されることに留意してください。

- 基本認証をサポートします。HTTPs に TLS を使用する場合、HTTP 認証が引き続き使用されることがあります。
- SSP で NetX Secure を使用する安全な通信のために TLS を有効または無効にするオプションを提供します。

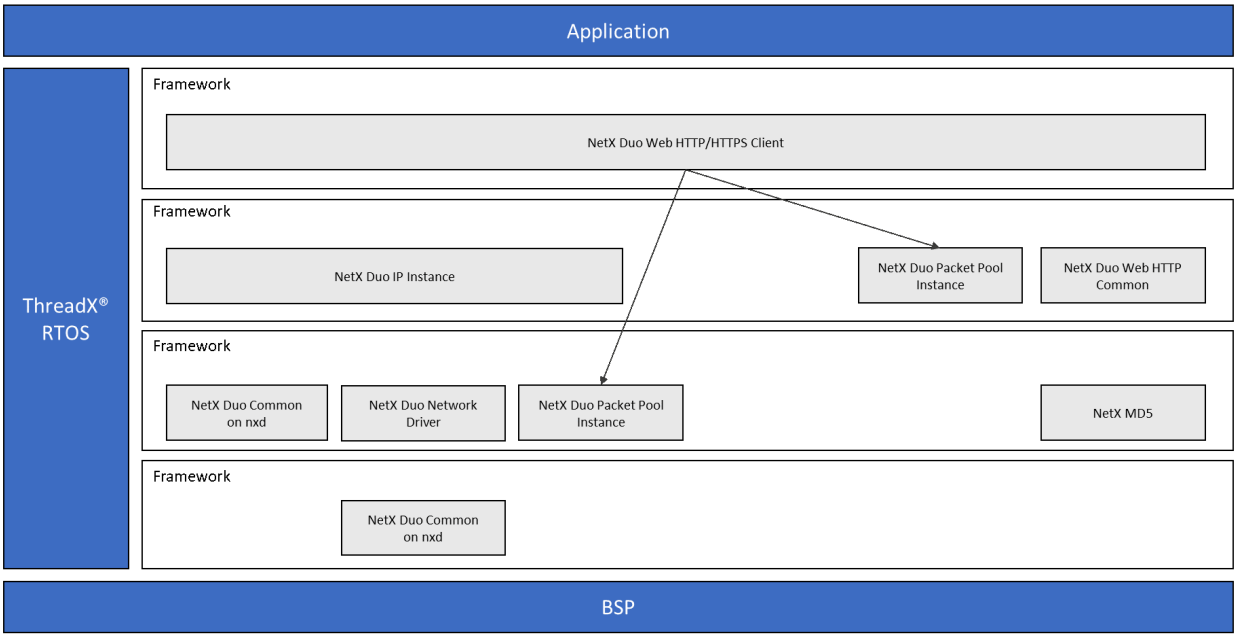


図 362:NetX Duo Web HTTP/HTTPs クライアントモジュールのブロック図

注 : 上の図で、NetX Duo ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.22.2 NetX Duo Web HTTP/HTTPs クライアントモジュールの API の概要

NetX Duo Web HTTP/HTTPs クライアントモジュールでは、作成、削除、GET および PUT のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX Duo Web HTTP/HTTPs クライアントモジュールの API の要約

Function Name	Example API Call and Description
nx_web_http_client_connect	nx_web_http_client_connect(NX_WEB_HTTP_CLIENT *client_ptr, NXD_ADDRESS *server_ip, UINT server_port, ULONG wait_option); Open a plaintext socket to an HTTP server for custom requests.
nx_web_http_client_create	nx_web_http_client_create(NX_WEB_HTTP_CLIENT *client_ptr, CHAR *client_name, NX_IP *ip_ptr, NX_PACKET_POOL *pool_ptr, ULONG window_size); Create an HTTP Client Instance.
nx_web_http_client_delete	nx_web_http_client_delete(NX_WEB_HTTP_CLIENT *client_ptr); Delete an HTTP Client Instance.
nx_web_http_client_delete_start	nx_web_http_client_delete_start(NX_WEB_HTTP_CLIENT *client_ptr, NXD_ADDRESS ip_address, UINT server_port, CHAR *resource, CHAR *host, CHAR *username, CHAR *password, ULONG wait_option); Start a plaintext HTTP DELETE request.
nx_web_http_client_delete_secure_start	nx_web_http_client_delete_secure_start(NX_WEB_HTTP_CLIENT *client_ptr, NXD_ADDRESS ip_address, UINT server_port, CHAR *resource, CHAR *host, CHAR *username, CHAR *password, UINT (*tls_setup)(NX_WEB_HTTP_CLIENT *client_ptr, NX_SECURE_TLS_SESSION *tls_session), ULONG wait_option); Start an encrypted HTTPS DELETE request.

Function Name	Example API Call and Description
nx_web_http_client_get_start	nx_web_http_client_get_start(NX_WEB_HTTP_CLIENT *client_ptr, NXD_ADDRESS ip_address, UINT server_port, CHAR *resource, CHAR *host, CHAR *username, CHAR *password, ULONG wait_option); Start a plaintext HTTP GET request.
nx_web_http_client_get_secure_start	nx_web_http_client_get_secure_start(NX_WEB_HTTP_CLIENT *client_ptr, NXD_ADDRESS ip_address, UINT server_port, CHAR *resource, CHAR *host, CHAR *username, CHAR *password, UINT (*tls_setup)(NX_WEB_HTTP_CLIENT *client_ptr, NX_SECURE_TLS_SESSION *tls_session), ULONG wait_option); Start an encrypted HTTPS GET request.
nx_web_http_client_head_start	nx_web_http_client_head_start(NX_WEB_HTTP_CLIENT *client_ptr, NXD_ADDRESS ip_address, UINT server_port, CHAR *resource, CHAR *host, CHAR *username, CHAR *password, ULONG wait_option); Start a plaintext HTTP HEAD request.
nx_web_http_client_head_secure_start	nx_web_http_client_head_secure_start(NX_WEB_HTTP_CLIENT *client_ptr, NXD_ADDRESS ip_address, UINT server_port, CHAR *resource, CHAR *host, CHAR *username, CHAR *password, UINT (*tls_setup)(NX_WEB_HTTP_CLIENT *client_ptr, NX_SECURE_TLS_SESSION *tls_session), ULONG wait_option); Start an encrypted HTTPS HEAD request.

Function Name	Example API Call and Description
nx_web_http_client_post_start	nx_web_http_client_post_start(NX_WEB_HTTP_CLIENT *client_ptr, NXD_ADDRESS ip_address, UINT server_port, CHAR *resource, CHAR *host, CHAR *username, CHAR *password, ULONG total_bytes, ULONG wait_option); Start an HTTP POST request.
nx_web_http_client_post_secure_start	nx_web_http_client_post_secure_start(NX_WEB_HTTP_CLIENT *client_ptr, NXD_ADDRESS ip_address, UINT server_port, CHAR *resource, CHAR *host, CHAR *username, CHAR *password, ULONG total_bytes, UINT (*tls_setup)(NX_WEB_HTTP_CLIENT *client_ptr, NX_SECURE_TLS_SESSION *tls_session), ULONG wait_option); Start an encrypted HTTPS POST request.
nx_web_http_client_put_start	nx_web_http_client_put_start(NX_WEB_HTTP_CLIENT *client_ptr, NXD_ADDRESS ip_address, UINT server_port, CHAR *resource, CHAR *host, CHAR *username, CHAR *password, ULONG total_bytes, ULONG wait_option); Start an HTTP PUT request.
nx_web_http_client_put_secure_start	nx_web_http_client_put_secure_start(NX_WEB_HTTP_CLIENT *client_ptr, NXD_ADDRESS ip_address, UINT server_port, CHAR *resource, CHAR *host, CHAR *username, CHAR *password, ULONG total_bytes, UINT (*tls_setup)(NX_WEB_HTTP_CLIENT *client_ptr, NX_SECURE_TLS_SESSION *tls_session), ULONG wait_option); Start an encrypted HTTPS PUT request.

Function Name	Example API Call and Description
nx_web_http_client_put_packet	nx_web_http_client_put_packet(NX_WEB_HTTP_CLIENT *client_ptr, NX_PACKET *packet_ptr, ULONG wait_option); Send next resource data packet.
nx_web_http_client_request_header_add	nx_web_http_client_request_header_add(NX_WEB_HTTP_CLIENT *client_ptr, CHAR *field_name, UINT name_length, CHAR *field_value, UINT value_length, UINT wait_option); Add a custom header to a custom HTTP request.
nx_web_http_client_request_initialize	nx_web_http_client_request_initialize (NX_WEB_HTTP_CLIENT *client_ptr, UINT method, CHAR *resource, CHAR *host, UINT input_size, UINT transfer_encoding_trunked, CHAR *username, CHAR *password, UINT wait_option); Initialize a custom HTTP request.
nx_web_http_client_request_send	nx_web_http_client_request_send(NX_WEB_HTTP_CLIENT *client_ptr, UINT wait_option); Send a custom HTTP request.
nx_web_http_client_response_body_get	nx_web_http_client_response_body_get(NX_WEB_HTTP_CLIENT *client_ptr, NX_PACKET **packet_ptr, ULONG wait_option); Get next resource data packet.

Function Name	Example API Call and Description
nx_web_http_client_response_header_callback_set	nx_web_http_client_response_header_callback_set(NX_WEB_HTTP_CLIENT *client_ptr, VOID (*callback_function)(NX_WEB_HTTP_CLIENT *client_ptr, CHAR *field_name, UINT field_name_length, CHAR *field_value, UINT field_value_length)); Set callback to invoke when processing HTTP headers.
nx_web_http_client_secure_connect	nx_web_http_client_secure_connect(NX_WEB_HTTP_CLIENT *client_ptr, NXD_ADDRESS *server_ip, UINT server_port, UINT (*tls_setup)(NX_WEB_HTTP_CLIENT *client_ptr, NX_SECURE_TLS_SESSION *tls), ULONG wait_option); Open a TLS session to an HTTPS server for custom requests.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

ステータス戻り値

Name	Description
NX_SUCCESS	Successful connection of TCP socket.
NX_PTR_ERROR	Invalid pointer input.
NX_WEB_HTTP_NOT_READY	Another request is already in progress.
NX_WEB_HTTP_POOL_ERROR	Invalid payload size in packet pool
NX_CALLER_ERROR	Invalid caller of this service.
NX_HTTP_PASSWORD_TOO_LONG	Password exceeded expected length
NX_WEB_HTTP_ERROR	Internal HTTP Client error.
NX_WEB_HTTP_NOT_READY	HTTP Client not ready.

Name	Description
NX_WEB_HTTP_FAILED	HTTP Client error communicating with the HTTP Server.
NX_WEB_HTTP_AUTHENTICATION_ERROR	Invalid name and/or password.
NX_WEB_HTTP_USERNAME_TOO_LONG	Username too large for buffer.
NX_SIZE_ERROR	Invalid total size of resource.
NX_WEB_HTTP_REQUEST_UNSUCCESSFUL_CODE	Received Server error code.
NX_WEB_HTTP_BAD_PACKET_LENGTH	Invalid packet length.
NX_WEB_HTTP_INCOMPLETE_PUT_ERROR	Server responds before PUT is complete.
NX_INVALID_PACKET	Packet too small for TCP header.
NX_WEB_HTTP_METHOD_ERROR	Some required information was missing (e.g. input_size for PUT or POST).
NX_WEB_HTTP_GET_DONE	HTTP Client get packet is done.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.3.22.3 NetX Duo Web HTTP/HTTPs クライアントモジュールの動作の概要

HTTP（ハイパーテキストトランスファープロトコル）は、HTTP クライアントとサーバー間でハイパーテキストをやり取りするのに使用されます。HTTP クライアントは、HTTP サーバー上の特定のポート（ポート 80、ポート 443 など）に対して TCP 接続を確立することにより、Get/POST/HEAD/PUT/DELETE などの HTTP 要求を開始します。そのポートで受信待ちしている HTTP サーバーは、クライアントの要求メッセージを待機します。要求を受信すると、サーバーは「HTTP/1.1 200 ok」のようなステータスメッセージを返信し、続いて自身のメッセージを送信します。

NetX Web HTTP/HTTPs クライアントモジュールは、ノーマルモードまたはセキュアモードで使用できます。

NetX Web HTTP/HTTPs クライアントモジュールのノーマルモードの動作の説明

ノーマルモードでは、HTTP クライアントとサーバーの間の通信はセキュアではありません。

NetX Web HTTP/HTTPs クライアントモジュールのセキュアモードの動作の説明

セキュアモードでは、TLS プロトコルによって HTTP クライアントとサーバーの間の通信がセキュアになっています。下の図に示すように、スレッドペインでは、[Add NetX Duo TLS common [Optional]] ブロックによって TLS プロトコルが表されます。

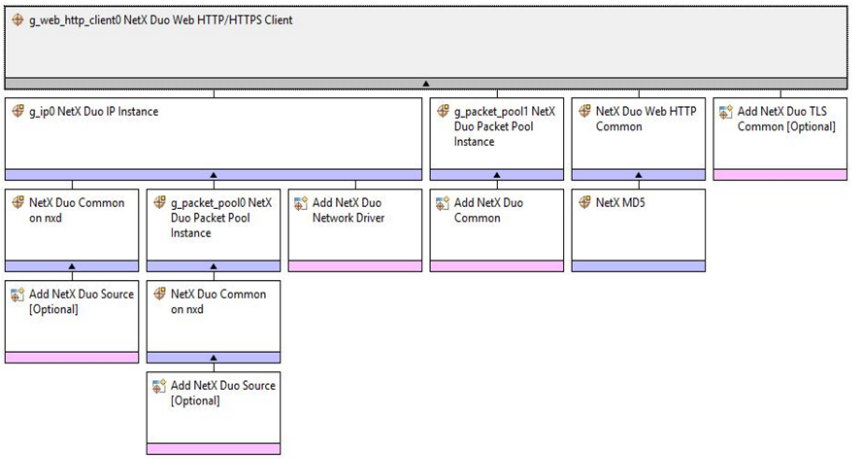


図 363:NetX Duo Web HTTP/HTTPs クライアントモジュールコンポーネントのスレッドペインビュー

NetX Duo TLS 共通ブロックを追加すると TLS のサポートが有効になり、NX_SECURE_ENABLE マクロが内部的に定義されます。次の図は、TLS のサポートが有効になっている HTTP クライアントのスレッドペインビューです。

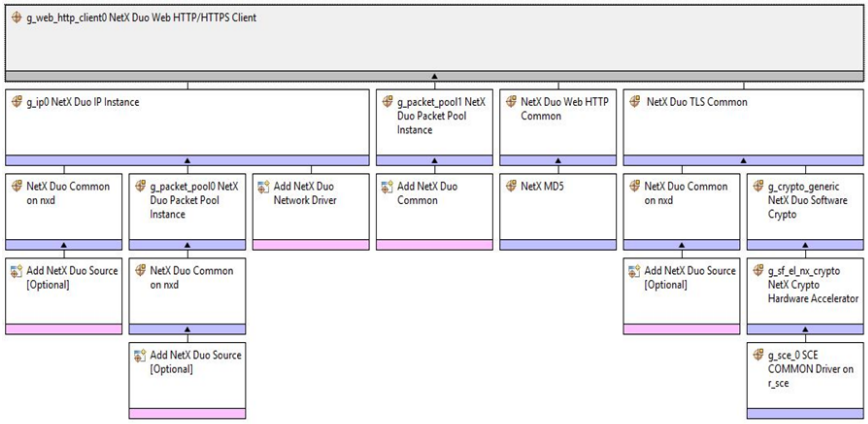


図 364:TLS のサポートが有効になっている NetX Duo Web HTTP/HTTPs クライアントモジュールコンポーネント

NetX Duo Web HTTP/HTTPs クライアントモジュールの動作に関する重要な注意事項と制限事項

スレッドペインウィンドウの [+] 記号 -> [X-Ware] -> [NetX Duo] -> [Protocols] -> [NetX Web HTTP/HTTPs Client] の順にクリックして、NetX Web HTTP/HTTPs クライアントコンポーネントを追加します。

NetX Web HTTP/HTTPs クライアントコンポーネントをプロジェクトに追加すると、セキュアな HTTP クライアントに必要な NetX Duo TLS コンポーネントを追加するためのオプションが自動的に追加されます。

NetX Web HTTP/HTTPs クライアントのプロパティを次の表に示します。

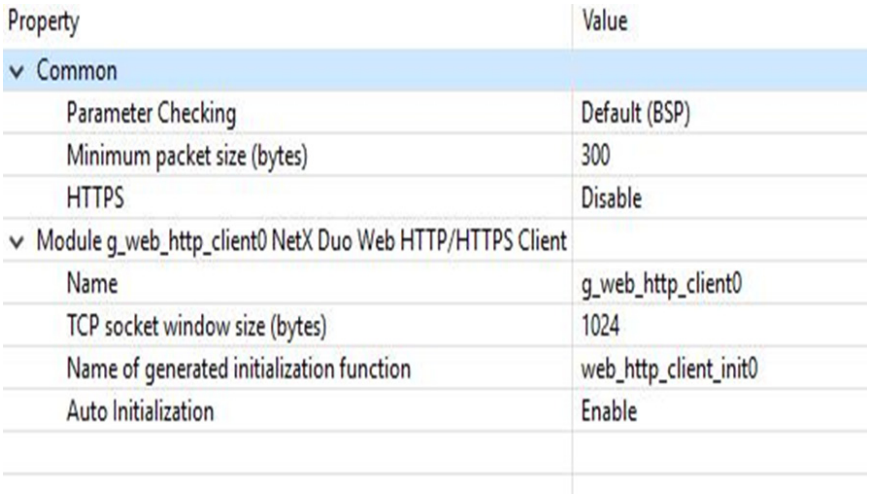


図 365:NetX Duo Web HTTP/HTTPs クライアントモジュールコンポーネントの構成可能なプロパティ

上の図で「Common」とされているプロパティは、プロジェクト内の HTTP/HTTPs クライアントのすべてのインスタンスで共通の、NetX Web HTTP/HTTPs クライアントの構成可能なオプションです。「Module」のプロパティは、プロジェクト内の HTTP/HTTPs クライアントの各インスタンスに固有です。

共通プロパティ

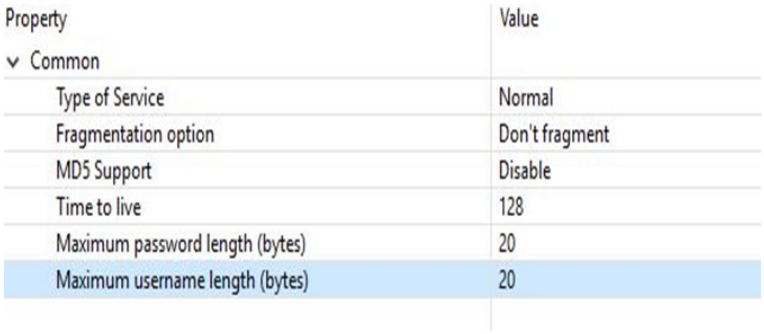
- Parameter Checking：基本的な HTTP エラーチェックを有効化 / 無効化します。通常、アプリケーションがデバッグされた後に使用します。デフォルト値は [BSP] です。
- Minimum Packet Size (bytes)：クライアント作成時に指定されたプール内のパケットの最小サイズです。最小サイズは、完全な HTTP ヘッダーを 1 つのパケットに確実に格納するため必要です。デフォルト値は 300 バイトです。
- HTTPS：セキュアなチャネルの確立に使用される TLS を有効化または無効化します。デフォルト値は無効です。

モジュールのプロパティ

- Name：HTTP/HTTPs クライアントインスタンスの名前です。
- TCP socket window size(bytes)：クライアントの TCP ソケット受信ウィンドウのサイズを設定します。デフォルトは 1024 バイトです。
- Name of generated initialization function：HTTP/HTTPs クライアントインスタンスを作成する初期化関数の名前です。デフォルトは、自動生成関数 web_http_client_init0. です。
- Auto Initialization：初期化関数のコールを有効化または無効化します。これは、[Name of generated initialization function] オプションで指定された関数がコールされるかどうかを決定します。有効に設定すると、その関数が呼び出されます。無効に設定した場合、アプリケーションは NetX Web HTTP/HTTPs クライアントサービスを使用する前に、nx_web_http_client_create() API をコールする必要があります。

NetX Duo Web HTTP 共通の構成可能なプロパティ

共通プロパティを次の表に示します。



Property	Value
Common	
Type of Service	Normal
Fragmentation option	Don't fragment
MD5 Support	Disable
Time to live	128
Maximum password length (bytes)	20
Maximum username length (bytes)	20

図 366:NetX Duo Web HTTP/HTTPs クライアントモジュールコンポーネント共通の構成可能なプロパティ

- Type of Service：HTTP TCP 要求に必要なネットワークサービスのタイプです。ネットワークサービスのタイプには、[Normal]、[Minimum delay]、[Maximum data]、[Maximum reliability]、[Minimum cost] があり、デフォルト値は [Normal] です。
- Fragmentation Option：HTTP クライアント要求の TCP フラグメンテーションを有効化 / 無効化します。デフォルト値は [Don't Fragment] です。
- MD5 Support：ダイジェスト認証に必要な MD5 ダイジェストサポートを有効化 / 無効化します。デフォルト値は無効です。
- Time to live：HTTP パケットが破棄されるまでに通過できるルータの最大数を指定します。デフォルトは 128 です。
- Maximum password length (bytes)：クライアントが提供するパスワードの最大長を定義します。デフォルトは 20 です。
- The Maximum username length (bytes)：クライアントが提供するユーザー名の最大長を定義します。デフォルトは 20 です。
- 要求のパイプライン化はサポートされません
- コンテンツ圧縮はサポートされていません。
- TRACE、OPTIONS、および CONNECT HTTP 要求はサポートされていません。
- クライアントに関連付けられたパケットプールには、完全な HTTP ヘッダーを保持するのに十分な大きさが必要です。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.22.4 アプリケーションへの NetX Duo Web HTTP/HTTPs クライアントモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX Duo Web HTTP/HTTPs クライアントモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX Duo Web HTTP/HTTPs クライアントモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX Duo Web HTTP/HTTPs クライアントモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_web_http_client0 NetX Duo Web HTTP/HTTPs Client	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo Web HTTP/HTTPs Client

次の図に示すように、NetX Duo Web HTTP/HTTPs クライアントモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

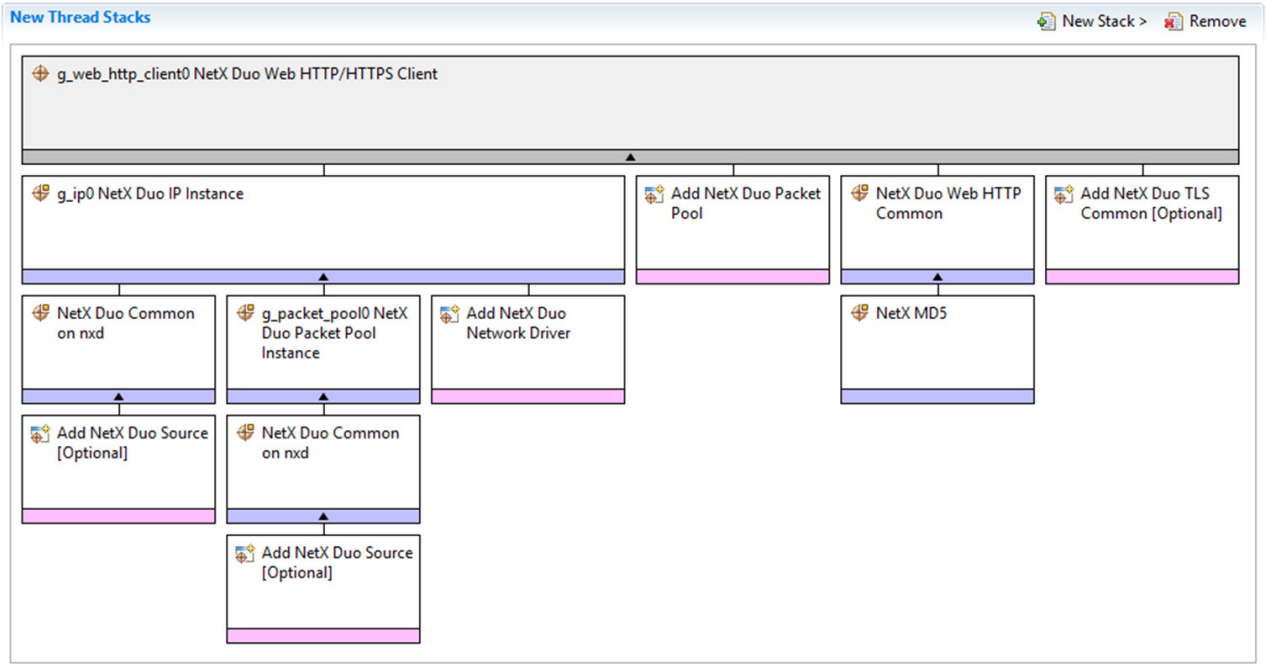


図 367:NetX Duo Web HTTP/HTTPs クライアントモジュールのスタック

上記のスタックでは、NetX Duo ネットワークドライバーがまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER

- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.22.5 NetX Duo Web HTTP/HTTPs クライアントモジュールの構成

ユーザーは必要な動作に合わせて NetX Duo Web HTTP/HTTPs クライアントモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX Duo Web HTTP/HTTPs クライアントモジュールの構成設定

ISDE Property	Value	Description
Parameter Checking	Enable, Disable, BSP Default: BSP	Selects if code for parameter checking is to be included in the build.
Minimum packet size (bytes)	300	Select the minimum packet size in bytes.
HTTPS Support	Enable, Disable Default: Disable	Select whether to enable HTTPS support.
Name	g_web_http_client0	Module name.
TCP socket window size (bytes)	1024	Select the TCP socket window size in bytes.
Name of generated initialization	web_http_client_init0	Name of generated initialization selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、イーサネットペリフェラル用に異なるピンの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX Duo Web HTTP/HTTPs クライアントのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
Default Gateway Address (use commas for separation)	0,0,0,0	Default gateway address selection
IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection

ISDE Property	Value	Description
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function
Link status change callback	NULL	Link status change callback selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo Web HTTP 共通インスタンスの構成設定

ISDE Property	Value	Description
Type of Service	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost Default: Normal	Type of service selection.
Fragmentation option	Don't fragment, Fragment okay Default: Don't fragment	Fragmentation option selection.
MD5 Support	Enable, Disable Default: Disable	MD5 support selection.
Time to live	128	Time to live selection.
Maximum password length (bytes)	20	Maximum password length in bytes.
Maximum username length (bytes)	20	Maximum username length in bytes,

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX MD5 インスタンスの構成設定

ISDE Property	Value	Description
No configurable properties()	-	-

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo Web HTTP/HTTPs クライアントモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX Duo Web HTTP/HTTPs クライアントモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin Selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.22.6 アプリケーションでの NetX Duo Web HTTP/HTTPs クライアントモジュールの使用

モジュールを構成してファイルが自動生成されると、NetX Web HTTP/HTTPs クライアントモジュールはアプリケーションで使用できる状態になります。自動生成されるコードには、[Name of generated initialization function] プロパティで指定した名前の初期化関数が含まれます。この関数は nx_web_http_client_create() API を内部的にコールして、[Name] プロパティで指定されている名前の HTTP/HTTPs クライアントインスタンスを作成します。この初期化関数の呼び出しは、[Auto Initialization] プロパティの値に応じて有効または無効になります。クライアントインスタンスが作成された後、アプリケーションでこのモジュールを使用する際の一般的な手順は次のとおりです。

1) アプリケーションは、nx_web_http_client_connect(). をコールすることでプレーンテキスト HTTP サーバーに接続できます。セキュアな接続を使用する必要がある場合、nx_web_http_client_secure_connect(). を呼び出すことで、セキュアな HTTP サーバーに接続できます。セキュアな接続のためには、アプリケーションに TLS セットアップコールバック関数を実装する必要があることに留意してください。付録の図 1 に、TLS セットアップコールバック関数のリファレンス実装を示します。これらの API は、サーバーへのプレーンな接続またはセキュアな接続を開くだけで要求は送信しません。
2) サーバーに接続した後、クライアントは nx_web_http_client_request_initialize(). をコールすることで、GET などのカスタム HTTP 要求を作成できます。
3) クライアントは、nx_web_http_client_request_header_add() API を呼び出すことで、手順 2 で作成した要求にカスタムヘッダーを追加できます。
4) サーバーに接続した後、クライアントは nx_web_http_client_request_send() API を使用してサーバーに要求を送信できます。
5) アプリケーションは、レスポンス全体が取得されるまで、nx_web_http_client_response_body_get() API を繰り返しコールすることによって、サーバーから送信されたレスポンスを取得できます。
6) アプリケーションは、nx_web_http_client_delete() を呼び出して、HTTP クライアントインスタンスに関連するすべてのリソースを削除する必要があります。

注 : アプリケーションでカスタム要求を作成したくない場合、nx_web_http_*_start() API を使用して標準の要求を送信できます。たとえば、標準の GET 要求をプレーンテキストサーバーに送信する場合、アプリケーションは nx_web_http_client_get_start(). を使用できます。同様に、nx_web_http_client_get_secure_start() を使用すれば、セキュアなチャネルを介して標準の GET 要求をサーバーに送信できます。nx_web_http_*_start() API は、内部的に nx_web_http_client_request_initialize() を使用して、目的の HTTP 要求を作成および送信します。

次の図は、以上の一般的な手順を示した通常の動作フロー図です。

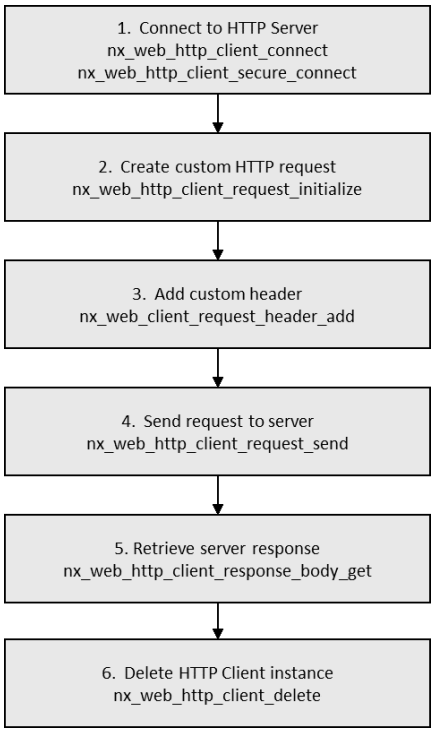


図 368:一般的な NetX Duo Web HTTP/HTTPs クライアントモジュールアプリケーションのフロー図

4.3.23 NetX Web HTTP/HTTPs サーバーフレームワーク

NetX Web HTTP/HTTPs サーバーフレームワークの概要

NetX Web HTTP/HTTPs サーバーモジュールは、Web 上のコンテンツを転送するための標準のハイパーテキストトランスファープロトコル（HTTP）にハイレベルな API 関数を提供します。HTTP プロトコルは、伝送制御プロトコル（TCP）サービスを使用してコンテンツ転送機能を実行します。HTTPs は、セキュアな HTTP プロトコルであり、基礎となる TCP 接続を保護するトランスポート層セキュリティ（TLS）プロトコルの上に HTTP を使用したものです。

NetX Web HTTP/HTTPs サーバーモジュールは SSP の試験的モジュールです。この試験的モジュールは Synergy プラットフォームではまだテストされておらず、開発での使用は推奨されません。試験的モジュールにはサポートが提供されていません。

NetX Web HTTP/HTTPs サーバーモジュールを試験的に使用するには、SSP 構成ツール（configuration.xml）の [component] タブに移動して、NetX Web HTTP サーバーモジュールを選択します。

4.3.24 NetX/NetX Duo SMTP クライアント

シンプルメールトランスファープロトコル（SMTP）は、ネットワークとインターネットを介してメールを送信するためのプロトコルで、信頼できる伝送制御プロトコル（TCP）サービスを使用してコンテンツ転送機能を実行します。

注 :NetX Duo ™ SMTP クライアントモジュールは、内部処理を除けば、SMTP クライアントセッションの応用、設定、実行が NetX ™ SMTP クライアントモジュールと同じです。NetX Duo で IPv6 用に IP インスタンスを設定する方法については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。

4.3.24.1 NetX/NetX Duo SMTP クライアントモジュールの特長

- NetX SMTP クライアント API は、RFC2821「シンプルメールトランスファープロトコル」および RFC 2554「SMTP 認証用サービス拡張機能」に準拠しています。
- また、以下を行うためのハイレベル API を提供します。
  - SMTP クライアントの作成と削除
  - メールメッセージの送信

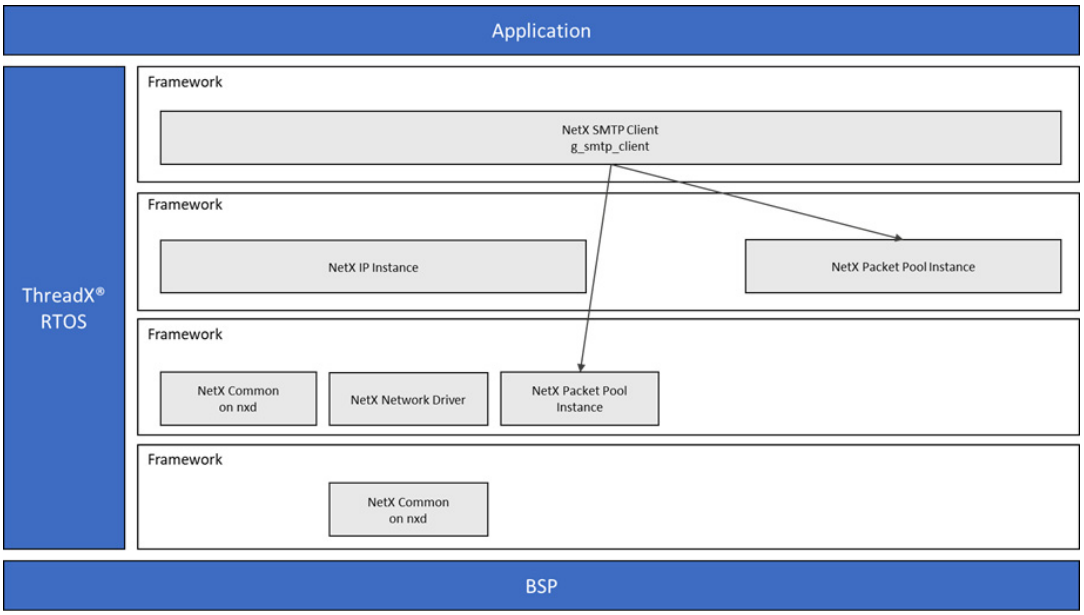


図 369:NetX/NetX Duo SMTP クライアントモジュールのブロック図

注 : 上の図で、NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.24.2 NetX/NetX Duo SMTP クライアントモジュールの API の概要

NetX SMTP クライアントでは、作成、削除、メール送信のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX/NetX Duo SMTP クライアントモジュールの API の要約

Function Name	Example API Call and Description
nx_smtp_client_create	nxd_smtp_client_create(&demo_client, &client_ip, &client_packet_pool,USERNAME,PASSWORD, FROM_ADDRESS, LOCAL_DOMAIN, CLIENT_AUTHENTICATION_TYPE, server_ip_address, SERVER_PORT); Create an SMTP Client Instance.
**nxd_smtp_client_create	nxd_smtp_client_create(&demo_client, &client_ip, client_packet_pool, USERNAME, PASSWORD, from_address, client_domain, authentication_type, &server_address, port); Create an SMTP Client instance for IPv4 or IPv6 networks.
nx_smtp_client_delete	nx_smtp_client_delete(&demo_client); Delete an SMTP Client instance.
nx_smtp_mail_send	nx_smtp_mail_send(&demo_client, recipient_address, NX_SMTP_MAIL_PRIORITY_NORMAL, SUBJECT_LINE, MAIL_BODY, strlen(MAIL_BODY)); Create and send an SMTP Mail item.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :** この API は、NetX Duo SMTP クライアントでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

ステータス戻り値

Name	Description
NX_SUCCESS	SMTP call successful

Name	Description
NX_CALLER_ERROR*	Invalid caller of this service
NX_PTR_ERROR*	Invalid input pointer parameter
NX_SMTP_INVALID_PARAM*	Invalid non-pointer input
NX_IP_ADDRESS_ERROR*	Invalid IP address type
NX_SMTP_CLIENT_NOT_INITIALIZED	SMTP Client instance initialized for SMTP session

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

注 : これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX と NetX Duo でのエラーチェックサービスの詳細については、それぞれ『NetX User Guide for the Renesas Synergy ™ Platform』または『NetX Duo User's Guide for the Renesas Synergy ™ Platform』を参照してください。*

4.3.24.3 NetX/NetX Duo SMTP クライアントモジュールの動作の概要

NetX IP インスタンスが作成され、その TCP サービスが有効化されます。SMTP クライアントは、ホストの電子メールアドレス、メールドメイン、認証タイプ、およびサーバー IP アドレス **（下の SMTP クライアントモジュール構成表のサーバー IPv4 アドレス）およびポート（下の SMTP クライアントモジュール構成表のサーバーポート）などのホストからの情報に加えて、以前に作成されたパケットプールを入力として作成されます。サーバーポートのデフォルトは、よく知られた SMTP ポート 25 です。

*注 :nx_smtp_client_create は NetX Duo POP3 クライアントでも使用できます。この場合、サーバー IPv4 アドレスを設定できます。ただし、SMTP クライアントを IPv6 サーバーとともに使用するには、アプリケーションで IPv6 アドレスを指定する必要があります。*SMTP クライアントが使用するパケットプールのペイロードは、一般的な SMTP データサイズとネットワークヘッダ（IP、TCP、および物理フレーム）に合わせて最適化する必要があります。そのパケットペイロードを超えるメッセージの場合、SMTP クライアントはパケットペイロードを超えているメッセージに追加のパケットを割り当てます。下のパケットプールインスタンス構成表では、パケットプールのペイロードの設定はデフォルトの 512、パケット数はデフォルトの 16 になっています。

*注 :SMTP クライアントインスタンスには、完全修飾メールアドレス（下の SMTP クライアントモジュール構成表のクライアントアドレス）が必要です。完全修飾ドメイン名には、「@」で区切られたローカル部分とドメイン名が含まれます。通常、クライアントアドレスの「@」記号の後ろの部分であるドメイン名も指定する必要があります（下の構成表のクライアントドメイン）。これは、SMTP クライアントのサーバーに対する HELO および EHLO グリーティングで使用されます。*NetX SMTP 認証

SMTP クライアントの作成には、認証タイプ、クライアント名、およびクライアントパスワードの設定も必要です（下の SMTP クライアントモジュールの構成表に記載されています）。クライアント名は、完全修飾ドメイン名または表示ユーザー名のいずれかにできます。

認証は、SMTP クライアントが SMTP サーバーに自身の身元を証明し、信頼できるユーザーとしてメールを配信する方法です。ほとんどの商用の SMTP サーバーでは、クライアントを認証する必要があります。

一般に、認証データは送信者のユーザー名とパスワードで構成されます。認証チャレンジの際、サーバーはこの情報を要求し、クライアントは要求されたデータをエンコードされたフォーマットで送信することでこれに応答します。サーバーはデータをデコードし、ユーザーデータベース内で一致するものを検索します。見つかった場合、サーバーは認証が成功したことを示します。

認証には基本認証とダイジェスト認証の 2 種類があります。ダイジェスト認証は現在の NetX SMTP クライアントではサポートされていないため、ここでは説明しません。基本認証は、前述の名前とパスワードを使った認証に相当します。SMTP の基本認証では、名前とパスワードが base64 でエンコードされています。基本認証の利点は実装が容易で、広く使用されていることです。基本認証の主な欠点は、要求で名前とパスワードのデータが隠されることなく送信されることです。

プレーン認証

NetX SMTP クライアントは PLAIN パラメータを指定して AUTH コマンドを送信します。NetX SMTP サーバーはこのタイプの認証をサポートしているかどうかを通知します。サポートしている場合、クライアントは base64 でエンコードされた単一のユーザー名およびパスワードメッセージをサーバーに返します。サーバーは、クライアント認証が成功したかどうかを示すステータスコードを返します。

ログイン認証

NetX SMTP クライアントは LOGIN パラメータを指定して AUTH コマンドを送信します。SMTP サーバーはこのタイプの認証をサポートしているかどうかを通知し、認証「チャレンジ」を開始します。サーバーは、base64 でエンコードされたプロンプトをクライアントに返します。これは通常「ユーザー名」です。クライアントはプロンプトをデコードし、base64 でエンコードされたユーザー名を返します。サーバーは、クライアントのユーザー名を受け入れると、クライアントのパスワードを求める base64 でエンコードされたプロンプトを送信します。クライアントは base64 でエンコードされたパスワードを返します。サーバーはクライアントの認証が成功したかどうかを通知します。

認証なし

SMTP サーバーは認証なしで構成されることがあります。その場合、クライアントの EHLO メッセージへの 250 レスポンスに認証タイプがリストされません。ただし、認証タイプがリストされないからといって、必ずしもサーバーが認証を要求またはサポートしないという意味ではありません。このような場合、クライアントに PLAIN または LOGIN の認証が構成されていると、NetX クライアントのスレッドタスクはデフォルトで PLAIN になります。クライアントが NONE に構成されている場合、認証手順はスキップされ、SMTP の状態は MAIL 状態になります。

クライアントが認証なしで構成されており、SMTP サーバーが認証をサポートしている場合、クライアントの認証タイプは PLAIN に切り替えられることに留意してください。

メールメッセージの送信

SMTP クライアントの作成後、SMTP クライアントアプリケーションは nx_smtp_mail_send サービスをコールすることでメッセージの送信を開始できます。NetX SMTP クライアントは、このサービスがコールされるたびに SMTP サーバーとの間に新しい TCP 接続を作成し、SMTP セッションを開始します。このセッションで、クライアントは SMTP プロトコルの一環として一連のコマンドを SMTP サーバーに送信します。これには、認証の受け渡しや実際のメールメッセージ送信の完了が含まれます。次に、SMTP セッションの結果にかかわらず、TCP 接続が終了します。メールメッセージが正常に送信された場合は、NX_SUCCESS が返されます。そうでない場合、SMTP クライアントエラー（認証失敗など）であるか、基礎となる NetX エラー（サーバーへの接続失敗など）であるかに応じて、エラーコードが表示されます。

SMTP セッションの後は、メールメッセージ送信の成否にかかわらず、SMTP クライアントが「初期」状態に戻り、別の SMTP セッション（同じ SMTP サーバー）に備えます。

NetX/NetX Duo SMTP クライアントモジュールの動作に関する重要な注意事項と制限事項

- NetX SMTP クライアント API は、RFC2821「シンプルメールトランスファープロトコル」および RFC 2554「SMTP 認証用サービス拡張機能」に準拠しています。
- NetX SMTP クライアントは、CRAM-MD5 ダイジェスト認証をサポートしていません。
- NetX SMTP クライアントメッセージは、メールアイテムごとに 1 人の受信者、SMTP サーバーとの TCP 接続ごとに 1 件のメールメッセージのみに制限されます。
- SMTP コマンドの VRFY、SEND、SOML、EXPN、SAML、ETRN、TURN、SIZE SMTP オプションはサポートされていません。

- SMTP クライアントは、一般にメールメッセージの作成に使用されるメールブラウザ（「メールユーザーエージェント」）とは異なります。「メールトランスファーエージェント」だけを指します。RFC 2821 で指定されているとおり、SMTP 転送のために必要なメールメッセージ本文の処理を提供します。内容の構文（受信者とリバース経路など）が正しいかどうかはチェックしません。メールバッファの内容（MIME データまたはクリアテキストメッセージ）に制限はありません。RFC 2822 で指定されている、ヘッダーとメッセージ本文を含めるためのメールメッセージフォーマットは、SMTP クライアント API の範囲外です。
- 制限事項の追加情報については『NetX ™ Simple Mail Transfer Protocol (SMTP) Client User Guide for the Renesas Synergy ™ Platform』を参照してください。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.24.4 アプリケーションへの NetX/NetX Duo SMTP クライアントモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo SMTP クライアントモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo SMTP クライアントモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo SMTP クライアントモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_smtp_client0 NetX SMTP Client	Threads	New Stack> X-Ware> NetX> Protocols> NetX SMTP Client
g_smtp_client0 NetX Duo SMTP Client	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo SMTP Client

次の図に示すように、NetX/NetX Duo SMTP クライアントモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

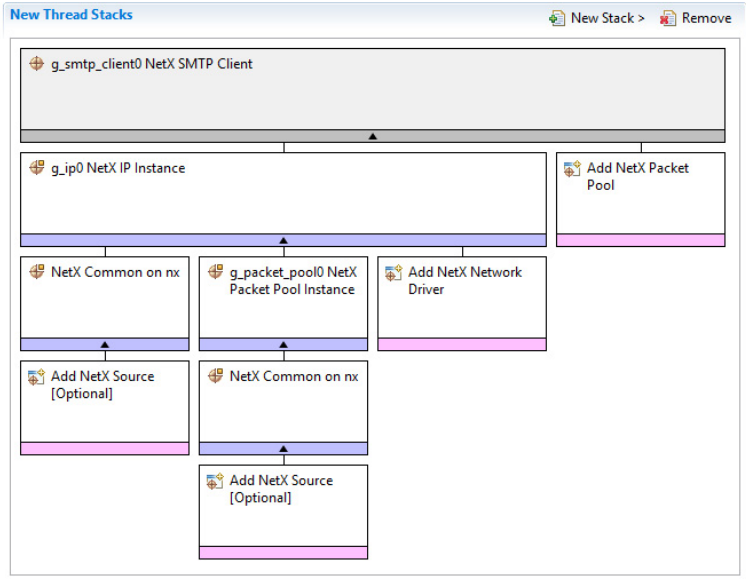


図 370:**NetX/NetX Duo SMTP クライアントモジュールのスタック**

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.24.5 NetX/NetX Duo SMTP クライアントモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo SMTP クライアントモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注：次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo SMTP クライアントモジュールの構成設定

ISDE Property	Value	Description
TCP window size (bytes)	1460	TCP window size selection
Packet allocation timeout (seconds)	2	Packet allocation timeout selection
TCP socket connect timeout (seconds)	10	TCP socket connect timeout selection
TCP socket disconnect timeout (seconds)	5	TCP socket disconnect timeout selection
Server greeting reply timeout	10	Server greeting reply timeout selection
Command timeout (seconds)	10	Command timeout selection
Mail data request timeout (seconds)	30	Mail data request timeout selection
TCP socket send completion timeout (seconds)	5	TCP socket send completion timeout selection
Server challenge maximum string length (bytes)	200	Server challenge maximum string length selection
Maximum password length (bytes)	20	Maximum password length selection
Maximum username length (bytes)	40	Maximum username length selection
Name	g_smtp_client0	Name selection
**Use server address type	IPv4, IPv6 Default: IPv6	Use server address type selection
Server IPv4 Address (use commas for separation)	192, 168, 0, 2	Server IPv4 Address selection
**Server IPv6 Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	Server IPv6 Address selection
Server Port	25	Server Port selection

ISDE Property	Value	Description
Client Name	username	Client Name selection
Client Password	password	Client Password selection
Client Address	mailto:username@domain.com	Client Address selection
Client Domain	domain.com	Client Domain selection
Authentication Type	Login	Authentication Type selection
Name of generated initialization function	smtp_client_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は NetX Duo でのみ使用可能なプロパティを表します。*

場合によっては、スタックモジュール用デフォルト以外の設定が望ましいこともあります。たとえば、イーサネットポートに対して異なるアドレスを選択するときに役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo SMTP クライアントのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	0,0,0,0	IPv4 Address selection

ISDE Property	Value	Description
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection

ISDE Property	Value	Description
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection

ISDE Property	Value	Description
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo SMTP クライアントモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo SMTP クライアントモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I^2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定表

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.24.6 アプリケーションでの NetX/NetX Duo SMTP クライアントモジュールの使用

一般的なアプリケーションで NetX/NetX Duo SMTP クライアントモジュールを使用する際の手順は次のとおりです。

1) nx_smtp_mail_send で送信するメールメッセージを作成します。
2) 必要に応じてメールメッセージの送信を続けます。
3) メッセージの送信が終了したら、nx_smtp_client_delete. をコールして SMTP クライアントを削除します。

次の図は、一般的な手順を示した通常の動作フロー図です。

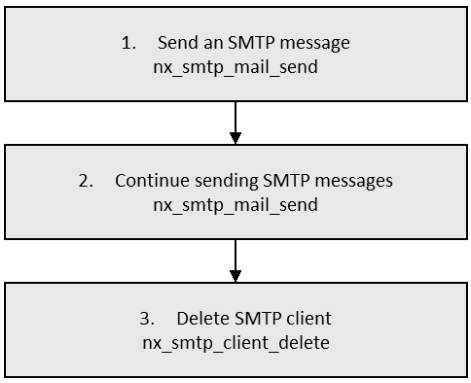


図 371:一般的な NetX/NetX Duo SMTP クライアントモジュールアプリケーションのフロー図

4.3.25 NetX/NetX Duo SNMP エージェント

NetX/NetX Duo SNMP エージェントモジュールは、SSP X-Ware 統合に含まれている NetX および NetX Duo アプリケーションバンドルの一部であり、SSP アーキテクチャ用の SNMP エージェントを実装するためのハイレベルな API を提供するものです。MCU に依存しないモジュールなので、NetX および NetX Duo をサポートしているすべての MCU に SNMP エージェントを実装できます。

注 :NetX Duo ™ SNMP エージェントモジュールは、内部処理を除けば、SNMP エージェントセッションの応用、設定、実行が NetX ™ SNMP エージェントモジュールと同じです。

4.3.25.1 NetX/NetX Duo SNMP エージェントモジュールの特長

- NetX/NetX Duo SNMP エージェントモジュールは、RFC1155、RFC1157、RFC1215、RFC1901、RFC1905、RFC1906、RFC1907、RFC1908、RFC2571、RFC2572、RFC2574、RFC2575、RFC 3414、および関連する RFC に準拠しています。
- SNMP エージェントモジュールは UDP でのみ動作します。TCP はサポートされていません。
- SNMP エージェントモジュールでは、トランスポート層セキュリティ（TLS）とデータグラムトランスポート層セキュリティ（DTLS）はサポートされていません。
- NetX/NetX Duo の SNMP プロトコルは、SNMP バージョン 1、2、3 を実装しています。SNMPv3 の実装では、MD5、セキュアハッシュアルゴリズム 1（SHA-1）認証、データ暗号化標準（DES）暗号化がサポートされます。このバージョンの NetX および NetX Duo SNMP エージェントには、以下の制約があります。
  - 1 つの NetX IP インスタンスに対して 1 つの SNMP エージェント。
  - RMON はサポート対象外。
  - SNMP v3 のインフォームメッセージはサポート対象外。
- SNMP エージェントの作成時に username、get、set、getnext を処理するためのコールバックを登録するメカニズムが提供されます。

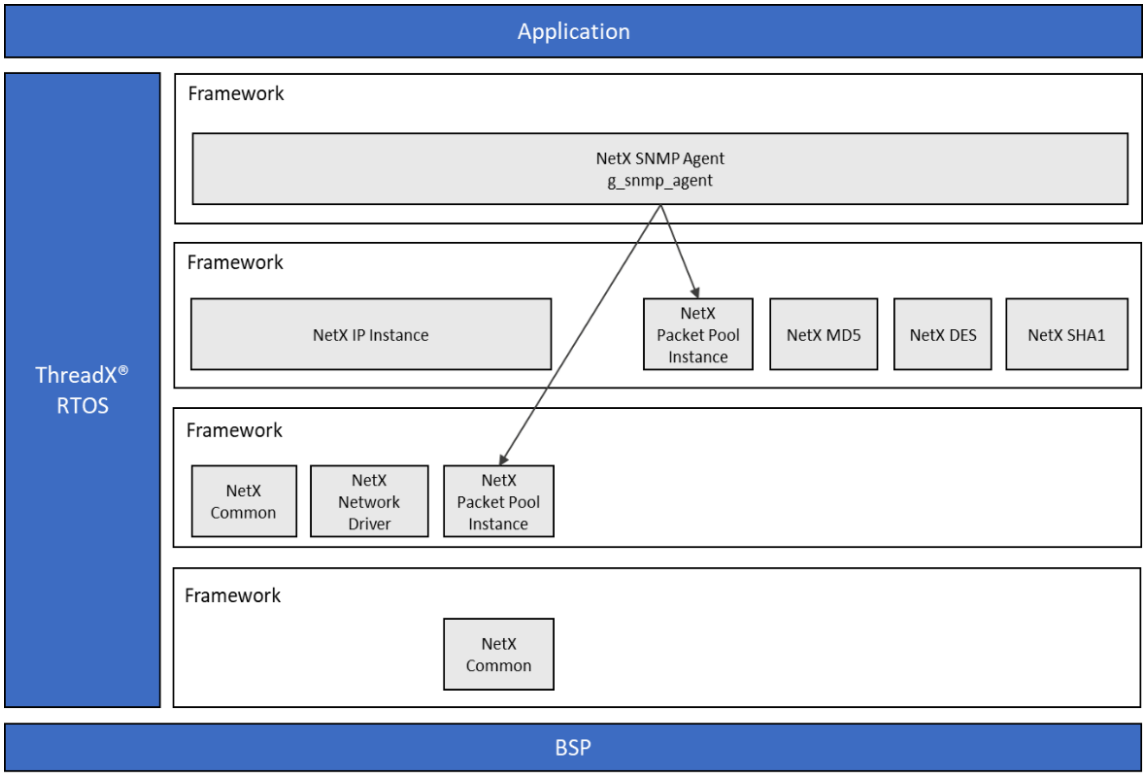


図 372:NetX/NetX Duo SNMP エージェントモジュールのブロック図

注 : 上の図で、NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.25.2 NetX/NetX Duo SNMP エージェントモジュールの API の概要

NetX Duo SNMP エージェントでは、SNMP エージェントインスタンスの作成、削除、およびそのメッセージの取得のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX/NetX Duo SNMP エージェントモジュールの API の要約（パート 1）

Function Name	Example API Call and Description
nx_snmp_agent_authenticate_key_use	(NX_SNMP_AGENT *agent_ptr, NX_SNMP_SECURITY_KEY *key); Register a previously created authentication key with the SNMP Agent for SNMP Agent responses
nx_snmp_agent_auth_trap_key_use	(NX_SNMP_AGENT *agent_ptr, NX_SNMP_SECURITY_KEY *key); Register a previously created authentication key with the SNMP Agent for SNMP trap messages.
nx_snmp_agent_community_get	(NX_SNMP_AGENT *agent_ptr, UCHAR **community_string_ptr); Retrieve the community string from the SNMP manager GET or GETNEXT request. The SNMP Agent should compare that to its own community string (see the nx_snmp_agent_public_string_test API description).
nx_snmp_agent_context_engine_set	(NX_SNMP_AGENT *agent_ptr, UCHAR *context_engine, UINT context_engine_size); Set the context engine ID of the SNMP Agent. Must specify the size of the ID string. Only used in SNMPv3 to identify the Agent to the SNMP Manager.
nx_snmp_agent_context_name_set	(NX_SNMP_AGENT *agent_ptr, UCHAR *context_name, UINT context_name_size); Set the context name of the SNMP Agent. This string must be null terminated and the size of the name must be specified. This name must be known to the SNMP Manager. Only used in SNMPv3.

Function Name	Example API Call and Description
nx_snmp_agent_create	(NX_SNMP_AGENT *agent_ptr, CHAR *snmp_agent_name, NX_IP *ip_ptr, VOID *stack_ptr, ULONG stack_size, NX_PACKET_POOL *pool_ptr, UINT (*snmp_agent_username_process)(struct NX_SNMP_AGENT_STRUCT *agent_ptr, UCHAR *username),UINT (*snmp_agent_get_process)(struct NX_SNMP_AGENT_STRUCT *agent_ptr, UCHAR *object_requested, NX_SNMP_OBJECT_DATA *object_data),UINT (*snmp_agent_getnext_process)(struct NX_SNMP_AGENT_STRUCT *agent_ptr, UCHAR *object_requested, NX_SNMP_OBJECT_DATA *object_data), UINT (*snmp_agent_set_process)(struct NX_SNMP_AGENT_STRUCT *agent_ptr, UCHAR *object_requested, NX_SNMP_OBJECT_DATA *object_data)); Create the SNMP Agent and set the Agent thread task size, packet pool for transmitting SNMP messages, and user defined callbacks for handling GET, GETNEXT, SET and username requests.
nx_snmp_agent_current_version_get	(NX_SNMP_AGENT *agent_ptr, UINT *version); Obtain the current version of SNMP based on the last message received.
nx_snmp_agent_delete	(NX_SNMP_AGENT *agent_ptr); Delete the SNMP instance
nx_snmp_agent_md5_key_create	(NX_SNMP_AGENT *agent_ptr, UCHAR *password, NX_SNMP_SECURITY_KEY *destination_key); Create a key based on a supplied password and SNMP Agent context engine ID using the MD5 algorithm. This key can be used for authentication or encryption.

Function Name	Example API Call and Description
nx_snmp_agent_privacy_key_use	(NX_SNMP_AGENT *agent_ptr, NX_SNMP_SECURITY_KEY *key); Register a previously created security key with the SNMP Agent for encrypting and decrypting SNMPv3 messages.
nx_snmp_agent_priv_trap_key_use	(NX_SNMP_AGENT *agent_ptr, NX_SNMP_SECURITY_KEY *key); Register a previously created security key with the SNMP Agent for encrypting SNMPv3 trap messages.
nx_snmp_agent_private_string_set	(NX_SNMP_AGENT *agent_ptr, UCHAR *private_string); Register a null terminated privacy string to the SNMP Agent. Only used in SNMP1 and SNMPv2.
nx_snmp_agent_private_string_test	(NX_SNMP_AGENT *agent_ptr, UCHAR *community_string, UINT *is_private); The SNMP Agent compares the privacy string in a SET request with the its own privacy string to determine if the SET request will be permitted.
nx_snmp_agent_public_string_set	(NX_SNMP_AGENT *agent_ptr, UCHAR *public_string); Register a null terminated public string to the SNMP Agent. Only used in SNMP1 and SNMPv2.
nx_snmp_agent_public_string_test	(NX_SNMP_AGENT *agent_ptr, UCHAR *community_string, UINT *is_public); The SNMP Agent compares the public string in a GET or GETNEXT request with the its own public string to determine if the request will be permitted.

Function Name	Example API Call and Description
nx_snmp_agent_request_get_type_test	(NX_SNMP_AGENT *agent_ptr, UINT *is_get_type); Determine if the last SNMP packet received was a GET, GETNEXT, or GET_BULT_REQUEST request type (returns TRUE) or a SET request (returns FALSE). Intended for use in the username callback for type of request received and checking public or private strings in the request message.
nx_snmp_agent_set_interface	(NX_SNMP_AGENT *agent_ptr, UINT if_index); Determine on which network interface to run the SNMP Agent protocol. The default interface is the primary (0) interface.
nx_snmp_agent_sha_key_create	(NX_SNMP_AGENT *agent_ptr, UCHAR *password, NX_SNMP_SECURITY_KEY *destination_key); Create a key based on a supplied password and SNMP Agent context engine ID using the SHA1 algorithm. This key can be used for authentication or encryption.
nx_snmp_agent_start	(NX_SNMP_AGENT *agent_ptr); Start the SNMP Agent thread task. This task waits to receive SNMP messages and formulates the response to the SNMP Manager.
nx_snmp_agent_stop	(NX_SNMP_AGENT *agent_ptr); Stop the SNMP Agent task thread. The SNMP Agent thread can be restarted by calling the nx_snmp_agent_start API.

Function Name	Example API Call and Description
nx_snmp_agent_trap_send	(NX_SNMP_AGENT *agent_ptr, ULONG ip_address, UCHAR * username, UCHAR *enterprise, UINT trap_type, UINT trap_code, ULONG elapsed_time, NX_SNMP_TRAP_OBJECT *object_list_ptr); Send a trap message in SNMPv1. This does not result from a request from the SNMP Manager. The SNMP application sends out traps as needed.
nx_snmp_agent_trapv2_send	(NX_SNMP_AGENT *agent_ptr, ULONG ip_address, UCHAR *username, UINT trap_type, ULONG elapsed_time, NX_SNMP_TRAP_OBJECT *object_list_ptr); Send a trap message in SNMPv2. This does not result from a request from the SNMP Manager. The SNMP application sends out traps as needed.
nx_snmp_agent_trapv3_send	(NX_SNMP_AGENT *agent_ptr, ULONG ip_address, UCHAR * username, UINT trap_type, ULONG elapsed_time, NX_SNMP_TRAP_OBJECT *object_list_ptr); Send a trap message in SNMPv3. This does not result from a request from the SNMP Manager. The SNMP application sends out traps as needed. Both the SNMP agent and browser must previously agree on the security (authentication and encryption) settings.
nx_snmp_agent_trapv2_oid_send	(NX_SNMP_AGENT *agent_ptr, ULONG ip_address, UCHAR *username, UCHAR *oid, ULONG elapsed_time, NX_SNMP_TRAP_OBJECT *object_list_ptr); Send a trap message in SNMPv2. This differs from nx_snmp_agent_trapv2_send in that it allows the caller to specify the OID directly.

Function Name	Example API Call and Description
nx_snmp_agent_trapv3_oid_send	(NX_SNMP_AGENT *agent_ptr, ULONG ip_address, UCHAR * username, UCHAR *oid, ULONG elapsed_time, NX_SNMP_TRAP_OBJECT *object_list_ptr); Send a trap message in SNMPv3. This differs from nx_snmp_agent_trapv3_send in that it allows the caller to specify the OID directly.
nx_snmp_agent_v3_context_boots_set	(NX_SNMP_AGENT *agent_ptr, UINT boots); Set the number of times the SNMP Agent has rebooted since the last communication with the SNMP Manager. Used in SNMPv3 only.
nx_snmp_agent_version_set	(NX_SNMP_AGENT *agent_ptr, UINT enabled_v1, UINT enable_v2, UINT enable_v3); Determine which type of SNMP packets the SNMP Agent will process. The application can choose V1, V2 and/or V3. Packets received from which the SNMP Agent is not enabled are dropped.
**nxd_snmp_agent_trap_send	(NX_SNMP_AGENT *agent_ptr, NXD_ADDRESS *ip_address, UCHAR * username, UCHAR *enterprise, UINT trap_type, UINT trap_code, ULONG elapsed_time, NX_SNMP_TRAP_OBJECT *object_list_ptr); Send a trap message in SNMPv1 over IPv6. Note that this takes an NXD_ADDRESS* ip_address instead of ULONG ip_address.

Function Name	Example API Call and Description
**nxd _snmp_agent_trapv2_send	(NX_SNMP_AGENT *agent_ptr, NXD_ADDRESS *ip_address, UCHAR * username, UINT trap_type, ULONG elapsed_time, NX_SNMP_TRAP_OBJECT *object_list_ptr); Send a trap message in SNMPv2. Note that this takes an NXD_ADDRESS* ip_address instead of ULONG ip_address
**nxd _snmp_agent_trapv3_send	(NX_SNMP_AGENT *agent_ptr, NXD_ADDRESS *ip_address, UCHAR *username, UINT trap_type, ULONG elapsed_time, NX_SNMP_TRAP_OBJECT *object_list_ptr); Send a trap message in SNMPv3. Note that this takes an NXD_ADDRESS* ip_address instead of ULONG ip_address
nx_snmp_agent_trapv2_oid_send	(NX_SNMP_AGENT *agent_ptr, NXD_ADDRESS *ip_address,, UCHAR *username, UCHAR *oid, ULONG elapsed_time, NX_SNMP_TRAP_OBJECT *object_list_ptr); Send a trap message in SNMPv2. Note that this takes an NXD_ADDRESS* ip_address instead of ULONG ip_address
**nxd _snmp_agent_trapv3_oid_send	(NX_SNMP_AGENT *agent_ptr, NXD_ADDRESS *ip_address, UCHAR * username, UCHAR *oid, ULONG elapsed_time, NX_SNMP_TRAP_OBJECT *object_list_ptr); Send a trap message in SNMPv3. Note that this takes an NXD_ADDRESS* ip_address instead of ULONG ip_address

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :** この API は、NetX Duo SNMP エージェントでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

次の API コールは、SNMP エージェントのレスポンス用にデータ項目を処理するためのものです。

NetX/NetX Duo SNMP エージェントモジュールの API の要約（パート 2）

Function Name	Example API Call and Description
nx_snmp_object_copy	(UCHAR *source_object_name, UCHAR *destination_object_name); This copies the string pointed to by source_object_name into the destination_object_name buffer (typically a trap message).
nx_snmp_object_counter_get	(VOID *source_ptr, NX_SNMP_OBJECT_DATA *object_data); This extracts the counter data from the location pointed to by source_ptr into the object data. Also used internally to copy internal counters of SNMPv3 statistics into error messages in SNMPv3 messages
nx_snmp_object_counter_set	(VOID *destination_ptr, NX_SNMP_OBJECT_DATA *object_data); This sets the value of data extracted from the object_data into the location pointed to by the destination_ptr.
nx_snmp_object_counter64_get	(VOID *source_ptr, NX_SNMP_OBJECT_DATA *object_data); This extracts the counter data from the location pointed to by source_ptr into the object data. The difference with nx_snmp_objext_counter_get is the value is two words instead of one.
nx_snmp_object_counter64_set	(VOID *source_ptr, NX_SNMP_OBJECT_DATA *object_data); This sets the value of data extracted from the object_data into the location pointed to by the destination_ptr. The difference with nx_snmp_objext_counter_set is the value is two words instead of one.

Function Name	Example API Call and Description
nx_snmp_object_compare	(UCHAR *requested_object, UCHAR *reference_object); This compares the two input objects and if equal returns NX_SUCCESS.
nx_snmp_object_copy	(UCHAR *source_object_name, UCHAR *destination_object_name) This copies the data pointed to by the source object pointer into memory pointed to by the destination object pointer. Also used internally to copy internal counters of SNMPv3 statistics into error messages in SNMPv3 messages.
nx_snmp_object_end_of_mib	(VOID *not_used_ptr, NX_SNMP_OBJECT_DATA *object_data); This appends an END_OF_MIB_VIEW macro as the input object's value. This signals the end of the MIB. See the snmp_mib_helper.h for an example.
nx_snmp_object_gauge_get	(VOID *source_ptr, NX_SNMP_OBJECT_DATA *object_data); This sets the input object to type SNMP GAUGE and places the value pointed to by the source_ptr into the object value.
nx_snmp_object_gauge_set	(VOID *destination_ptr, NX_SNMP_OBJECT_DATA *object_data); This verifies the input object is an SNMP GAUGE data type, and extracts the value into the location pointed to by the destination pointer.

Function Name	Example API Call and Description
nx_snmp_object_id_get	(VOID *source_ptr, NX_SNMP_OBJECT_DATA *object_data); This function retrieves the object ID from the specified source location and writes it into the object data value.
nx_snmp_object_id_set	(VOID *destination_ptr, NX_SNMP_OBJECT_DATA *object_data); This function retrieves the ASCII string from the input object and writes it to the area pointed to by the destination pointer.
nx_snmp_object_integer_get	(VOID *source_ptr, NX_SNMP_OBJECT_DATA *object_data); This retrieves the object integer from the specified source location and stores it to the object data.
nx_snmp_object_integer_set	(VOID *destination_ptr, NX_SNMP_OBJECT_DATA *object_data); This retrieves the integer from the input object and places it in the destination.
nx_snmp_object_ip_address_get	(VOID *source_ptr, NX_SNMP_OBJECT_DATA *object_data); This retrieves the IP address from the specified source location and stores it to the object data.
nx_snmp_object_ip_address_set	(VOID *destination_ptr, NX_SNMP_OBJECT_DATA *object_data); This retrieves the IP address from the input object and places it in the destination.

Function Name	Example API Call and Description
**nx_snmp_object_ipv6_address_get	(VOID *source_ptr, NX_SNMP_OBJECT_DATA *object_data); This retrieves the IPv6 address from the specified source location and stores it to the object data.
**nx_snmp_object_ipv6_address_set	(VOID *destination_ptr, NX_SNMP_OBJECT_DATA *object_data); This retrieves the IPv6 address from the input object and places it in the destination.
nx_snmp_object_octet_string_get	(VOID *source_ptr, NX_SNMP_OBJECT_DATA *object_data, UINT length); This retrieves the string data from the specified source location and stores it to the object data.
nx_snmp_object_octet_string_set	(VOID *destination_ptr, NX_SNMP_OBJECT_DATA *object_data); This retrieves the string from the input object and places it in the destination.
nx_snmp_object_no_instance	(VOID *not_used_ptr, NX_SNMP_OBJECT_DATA *object_data); This function places a no-instance value (NX_SNMP_ANS1_NO_SUCH_INSTANCE) in the object data.
nx_snmp_object_not_found	(VOID *not_used_ptr, NX_SNMP_OBJECT_DATA *object_data); This function places a not-found value (NX_SNMP_ANS1_NO_SUCH_OBJECT) in the object data.

Function Name	Example API Call and Description
nx_snmp_object_string_get	(VOID *source_ptr, NX_SNMP_OBJECT_DATA *object_data, UINT length); This function retrieves the ASCII string from the specified source location and stores it to the object data.
nx_snmp_object_string_set	(VOID *destination_ptr, NX_SNMP_OBJECT_DATA *object_data); This retrieves the ASCII string from the input object and stores it to the destination.
nx_snmp_object_timetics_get	(VOID *source_ptr, NX_SNMP_OBJECT_DATA *object_data); This function retrieves the data of type timer ticks from the specified source location and stores it to the object data.
nx_snmp_object_timetics_set	(VOID *destination_ptr, NX_SNMP_OBJECT_DATA *object_data); This retrieves the timer tick from the input object and stores it to the destination.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :** この API は、NetX Duo SNMP エージェントでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

ステータス戻り値

Name	Description
NX_SUCCESS	API Call Successful
NX_PTR_ERROR*	Invalid input pointer parameter

Name	Description
NX_SNMP_UNSUPPORTED_AUTHENTICATION*	The authentication key is of an unknown or unsupported type (for example, not MD5 or SHA).
NX_SNMP_INVALID_PDU_ENCRYPTION*	The encryption key is of an unknown or unsupported type (for example, not MD5 or SHA).
NX_IP_ADDRESS_ERROR*	IP address supplied in a trap send API is null or invalid.
NX_SNMP_ERROR	Internal processing error, for example, not able to append data into the SNMP response.
NX_NOT_ENABLED	SNMP Agent is not enabled with the correct security settings for certain operations such as sending messages or processing authentication and encryption keys.
NX_SNMP_ERROR_TOOBIG	Data exceeds the size of the response buffer or exceed the allowable size of the parameter e.g. NX_SNMP_MAX_USER_NAME.
**NX_SNMP_INVALID_IP_PROTOCOL_ERROR	An IPv6 address is received in a trap send API but the NetX Duo library is not enabled for IPv6.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

注 : これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX と NetX Duo でのエラーチェックサービスの詳細については、それぞれ『NetX User Guide for the Renesas Synergy ™ Platform』または『NetX Duo User's Guide for the Renesas Synergy ™ Platform』を参照してください。*

4.3.25.3 NetX/NetX Duo SNMP エージェントモジュールの動作の概要

SNMP エージェントモジュールは、基礎となる NetX/NetX Duo スタックを使用して動作を実行します。SNMP v3 動作での認証と暗号化には、IP スタックに加えて、NetX MD5、NetX SHA1、NetX DES モジュールを使用します。

SNMP エージェントモジュールは nx_snmp_agent_create API を使用して作成できます。SNMP エージェントの実装のために、ユーザーは username、get、getnext および set 動作のためのハンドラ関数を定義する必要があります。

NetX/NetX Duo SNMP エージェントモジュールの動作に関する重要な注意事項と制限事項

SNMP バージョン 1 の無効化

SNMP エージェントモジュールでは、「SNMP バージョン 1」の SNMP エージェントの構成プロパティで [Disable] を選択することで、バージョン 1 の要求の処理を無効にすることができます。デフォルトでは有効になっています。無

効にすると、SNMP エージェントはバージョン 1 のパケットを単にドロップし、SNMP マネージャでタイムアウトが発生します。

SNMP バージョン 2 の無効化

SNMP エージェントモジュールでは、「SNMP バージョン 2」の SNMP エージェントの構成プロパティで [Disable] を選択することで、バージョン 2 の要求の処理を無効にすることができます。デフォルトでは有効になっています。無効にすると、SNMP エージェントはバージョン 2 のパケットを単にドロップし、SNMP マネージャでタイムアウトが発生します。

SNMP バージョン 3 の無効化

SNMP エージェントでは、「SNMP バージョン 3」の SNMP エージェントの構成プロパティで [Disable] を選択することで、バージョン 3 の要求の処理を無効にすることができます。デフォルトでは有効になっています。無効にすると、SNMP エージェントはバージョン 3 のパケットを単にドロップし、SNMP マネージャでタイムアウトが発生します。SNMP バージョン 3 を有効化するには MD5、SHA1 認証、および DES 暗号化が必要です。

- 1 つの NetX IP インスタンスに対して 1 つの SNMP エージェント。
- RMON はサポート対象外。
- SNMP v3 のインフォームはサポート対象外。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.25.4 アプリケーションへの NetX/NetX Duo SNMP エージェントモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo SNMP エージェントモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo SNMP エージェントモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo SNMP エージェントモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_snmp_agent0 NetX SNMP Agent	Threads	New Stack> X-Ware> NetX> Protocols> NetX SNMP Agent
g_snmp_agent0 NetX Duo SNMP Agent	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo SNMP Agent

次の図に示すように、NetX/NetX Duo SNMP エージェントモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボッ

クスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

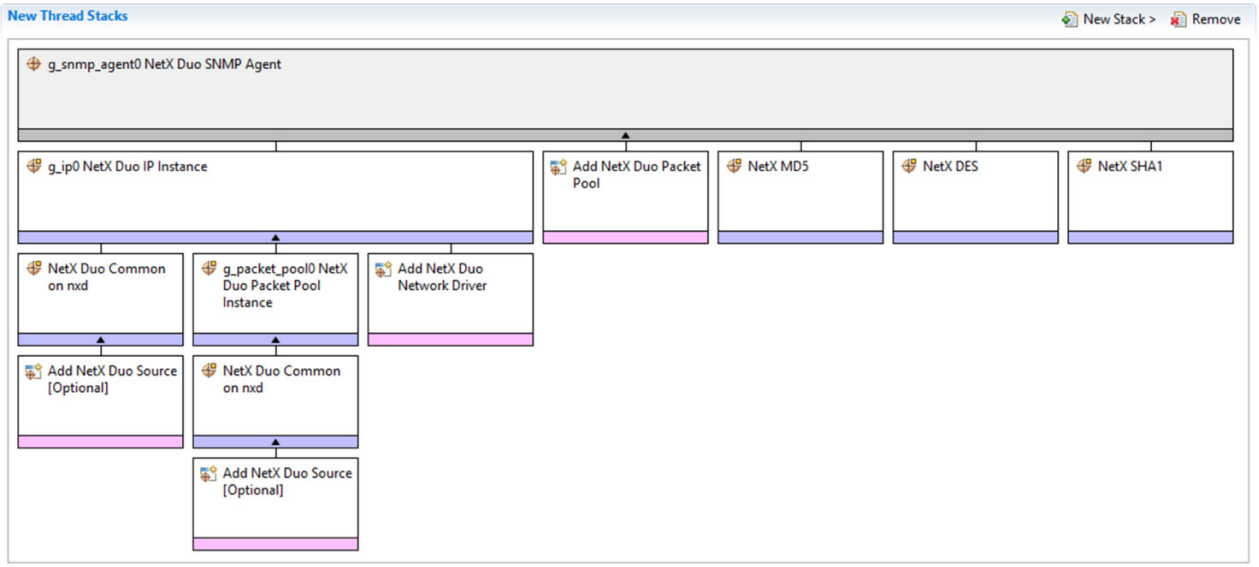


図 373:NetX/NetX Duo SNMP エージェントモジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.25.5 NetX/NetX Duo SNMP エージェントモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo SNMP エージェントモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo SNMP エージェントモジュールの構成設定

ISDE Property	Value	Description
Internal thread stack size (bytes)	4096	Internal thread stack size selection
SNMP agent priority	16	SNMP agent priority selection
Type of service for SNMP responses	Normal, Minimum Delay, Maximum Delay, Maximum Reliability, Minimum Cost Default: Normal	Type of service for SNMP responses selection
Fragment enable for SNMP PDU requests	Fragment, Don't Fragment Default: Don't Fragment	Fragment enable for SNMP PDU requests selection
SNMP socket time to live	128	SNMP socket time to live selection
Agent timeout	100	Agent timeout selection
Max octet string size	255	Max octet string size selection
Max content string size	32	Max content string size selection
Max User Name Size	64	Max user name size selection
Max security Key Size	64	Max security key size selection
Minimum SNMP packet size	560	Minimum SNMP packet size selection (Value must be between 560 to 1500)

ISDE Property	Value	Description
UDP port number	161	UDP port number selection (Value must be between 1 to 65535)
Trap destination port	162	Trap destination port selection
Max trap Name Size	64	Max trap Name Size selection
Max trap Key Size	64	Max trap Key Size selection
Interval between SNMP packet processing timer ticks	100	Interval between SNMP packet processing timer ticks selection
SNMP Version 1	Enable, Disable Default: Enable	SNMP Version 1 selection
SNMP Version 2	Enable, Disable Default: Enable	SNMP Version 2 selection
SNMP Version 3	Enable, Disable Default: Enable	SNMP Version 3 selection
Name	g_snmp_agent0	Name selection
Read Community String	public	Read Community String selection (Must be less than 64 chars)
Write Community String	private	Write Community String selection (Must be less than 64 chars)
Name of SNMP Username Handler	sf_snmp0_username_handler	Name of SNMP Username Handler selection
Name of SNMP GET Handler	sf_snmp0_get_handler	Name of SNMP GET Handler selection
Name of SNMP GETNEXT Handler	sf_snmp0_getnext_handler	Name of SNMP GETNEXT Handler selection

ISDE Property	Value	Description
Name of SNMP SET Handler	sf_snmp0_set_handler	Name of SNMP SET Handler selection
Name of generated initialization function	snmp_agent_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto Initialization selection
SNMP agent instance id	0	SNMP agent instance id selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

スタックモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、イーサネットポートに対して異なるアドレスを選択するときに役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo SNMP エージェントのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	0,0,0,0	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection

ISDE Property	Value	Description
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection

ISDE Property	Value	Description
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo MD5 インスタンスの構成設定

ISDE Property	Value	Description
No configurable properties()	-	-

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo DES インスタンスの構成設定

ISDE Property	Value	Description
No configurable properties()	-	-

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo SHA1 インスタンスの構成設定

ISDE Property	Value	Description
No configurable properties()	-	-

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo SNMP エージェントモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo SNMP エージェントモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I^2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択表

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.25.6 アプリケーションでの NetX/NetX Duo SNMP エージェントモジュールの使用

一般的なアプリケーションで NetX/NetX Duo SNMP エージェントモジュールを使用する際の手順は次のとおりです。

1) nx_snmp_agent_create API でエージェントを作成します。
2) ハンドラ関数を定義します。
   a) username ハンドラ関数 nx_snmp_agent_username_process（ユーザーコード内）
   b) get ハンドラ関数 nx_snmp_agent_get_process（ユーザーコード内）
   c) get next ハンドラ関数 nx_snmp_agent_getnext_process（ユーザーコード内）
   d) set ハンドラ関数 nx_snmp_agent_set_process（ユーザーコード内）
3) OID をアクションハンドラにマップする構造体をユーザーコード内に作成します。
4) リード、ライトのコミュニティストリング nx_snmp_agent_public_string_set および nx_snmp_agent_private_string_set を構成します。
5) ユーザーが SNMPv3 を使用したい場合、nx_snmp_agent_md5_key_create API または nx_snmp_agent_sha_key_create を使用してキーを作成し、そのキーを認証、暗号化、トラップサービス nx_snmp_agent_authenticate_key_use,、nx_snmp_agent_privacy_key_use,、nx_snmp_agent_auth_trap_key_use、nx_snmp_agent_priv_trap_key_use に関連付ける必要があります。
6) nx_snmp_agent_start API を使用して SNMP エージェントを開始します。

次の図は、一般的な手順を示した通常の動作フロー図です。

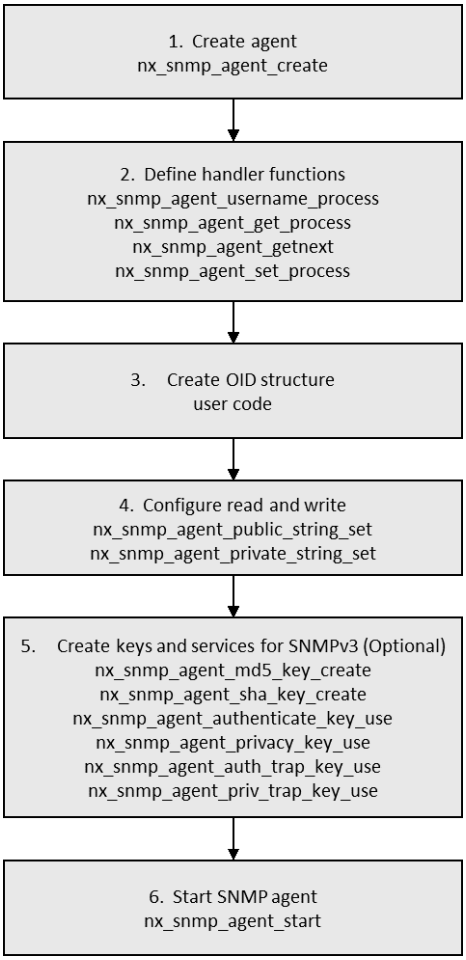


図 374:一般的な NetX/NetX Duo SNMP エージェントモジュールアプリケーションのフロー図

4.3.26 NetX/NetX Duo SNTP クライアント

シンプルネットワークタイムプロトコル（SNTP）は、インターネットを介してクロックを同期するために設計されたプロトコルです。SNTP バージョン 4 は、ネットワークタイムプロトコル（NTP）に基づく簡易プロトコルで、ユーザーデータグラムプロトコル（UDP）サービスを使用して、簡単でステートレスなプロトコルで時刻更新を実行します。SNTP は、インターネット上で、同期ソースとネットワークパスの特性に応じて 1 ～ 50 ミリ秒の精度を提供します。SNTP は、時刻更新の受信に信頼性を確保するためのオプションを多数備えています。代替サーバーへの切り替え、バックオフポーリングアルゴリズムの適用、自動タイムサーバー検出などの機能があり、これらは SNTP クライアントがさまざまなインターネットタイムサービス環境を処理する手段のほんの一部にすぎません。精度に劣りますが、その分、実装の簡素さや容易さに優れています。SNTP の主な目的は、NTP サーバーサービスの国内時刻および周波数伝達にアクセスするための包括的なメカニズムを提供することです。

注 :NetX Duo ™ SNTP クライアントは、内部処理を除けば、SNTP スレッドの応用、設定、実行が NetX ™ SNTP クライアントと同じです。このドキュメントでは、NetX と NetX Duo の SNTP クライアントの使用方法の違いを明確に示します。

4.3.26.1 NetX/NetX Duo SNTP クライアントモジュールの特長

- インターネット上で時刻を取得するため SNTP 標準をサポート
- NetX SNTP クライアントは、RFC4330「IPv4、IPv6、OSI のためのシンプルネットワークタイムプロトコル（SNTP）バージョン 4」および関連する RFC に準拠
- ユニキャストとブロードキャストの両方の SNTP メッセージングをサポート
- NTP 時刻を日付と時刻のフォーマットに変換するためのユーティリティ
- 有効な時刻受信のサニティチェックを実行
- セカンダリインタフェースで SNTP クライアントをサポート
- 時刻更新の受信通知を受け取るためのコールバック

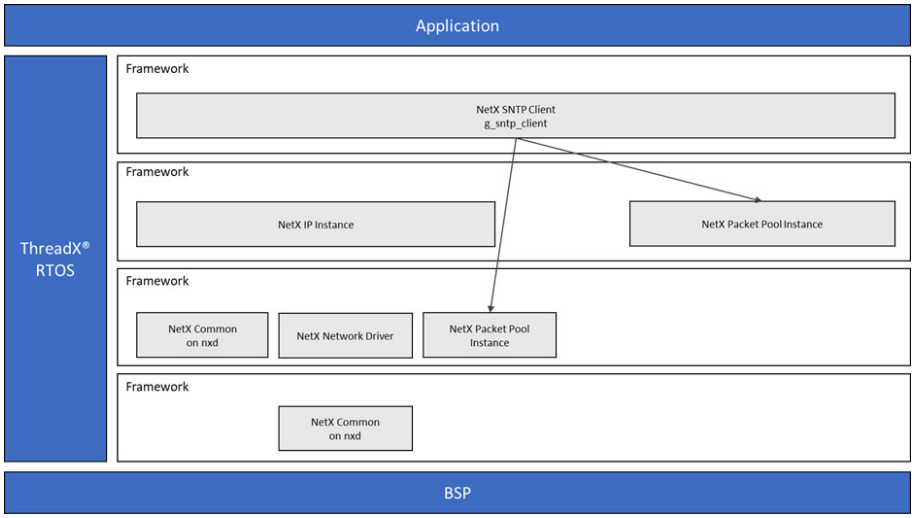


図 375:NetX/NetX Duo SNTP クライアントモジュールのブロック図

注：上の図で、NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.26.2 NetX/NetX Duo SNTP クライアントモジュールの API の概要

NetX SNTP クライアントモジュールでは、作成、削除、レスポンスパケットの生成、レスポンス送信、受信パケットからの情報の取得のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX/NetX Duo SNTP クライアントモジュールの API の要約

Function Name	Example API Call and Description
nx_sntp_client_create	nx_sntp_client_create(&demo_client, iface_index,&client_ip, &client_packet_pool, leap_second_handler, kiss_of_death_handler,NULL /* no random_number_generator callback */); This service creates an SNTP Client instance.
nx_sntp_client_delete	nx_sntp_client_delete(&demo_client); Delete the SNTP Client.
nx_sntp_client_get_local_time	nx_sntp_client_get_local_time(&demo_client, &base_seconds,&base_milliseconds, NX_NULL); Get SNTP Client local time.
nx_sntp_client_initialize_broadcast	nx_sntp_client_initialize_broadcast(&demo_client,0x0, NX_NULL, IP_ADDRESS(192,2,2,255); Initialize Client for IPv4 broadcast operation.
**nxd_sntp_client_initialize_broadcast	nxd_sntp_client_initialize_broadcast(&demo_client,0x0, NX_NULL, &broadcast_server) Initialize Client for IPv6 or IPv4 broadcast operation.
nx_sntp_client_initialize_unicast	nx_sntp_client_initialize_unicast(&demo_client, IP_ADDRESS(192,2,2,1)); Initialize Client for IPv4 unicast operation.
**nxd_sntp_client_initialize_unicast	nxd_sntp_client_initialize_unicast(&demo_client, *unicast_server); Initialize Client for IPv4 or IPv6 unicast operation.
nx_sntp_client_receiving_updates	nx_sntp_client_receiving_updates(&demo_client, &receive_status); Client is currently receiving valid SNTP updates.

Function Name	Example API Call and Description
nx_sntp_client_request_unicast_time	nx_sntp_client_request_unicast_time(&demo_client, 400); Send a request asynchronously to NTP server.
nx_sntp_client_run_broadcast	nx_sntp_client_run_broadcast(&demo_client); Receive time updates from server.
nx_sntp_client_run_unicast	nx_sntp_client_run_unicast(&demo_client); Send requests and receive time updates from server.
nx_sntp_client_set_local_time	nx_sntp_client_set_local_time(&demo_client, base_seconds, base_fraction); Set SNTP Client initial local time.
nx_sntp_client_set_time_update_notify	nx_sntp_client_set_time_update_notify(&demo_client, time_update_cb); Sets callback to notify the application when the SNTP Client receives a valid time update.
nx_sntp_client_stop	nx_sntp_client_stop(&demo_client); Stop the SNTP Client thread.
nx_sntp_client_utility_display_date_and_time	nx_sntp_client_utility_display_date_time(&demo_client , buffer, sizeof(buffer)); Display NTP time in date and time format
nx_sntp_client_utililty_msecs_to_fraction	nx_sntp_client_utility_msecs_to_fraction(miliseconds, &fraction); Convert milliseconds to NTP fraction component.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :** この API は、NetX Duo SNTP クライアントでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

ステータス戻り値

Name	Description
NX_SUCCESS	API Call Successful
NX_SNTP_INSUFFICIENT_PACKET_PAYLOAD	Invalid non-pointer input
NX_PTR_ERROR **	Invalid pointer input
NX_CALLER_ERROR **	Invalid caller of service
NX_INVALID_INTERFACE	Invalid network interface
NX_INVALID_PARAMETERS	Invalid non-pointer input
NX_SNTP_PARAM_ERROR	Invalid non-pointer input
NX_SNTP_CLIENT_NOT_STARTED	SNTP Client not running
NX_SNTP_CLIENT_ALREADY_STARTED	Client already running
NX_SNTP_CLIENT_NOT_INITIALIZED	Client not initialized
NX_SNTP_ERROR_CONVERTING_DATETIME	NX_SNTP_CURRENT_YEAR not defined or no local time established
NX_SNTP_INVALID_DATETIME_BUFFER	Insufficient buffer length
NX_SNTP_OVERFLOW_ERROR	Error converting time to a date
NX_SNTP_INVALID_TIME	Invalid SNTP data input

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

*注 :** これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX と NetX Duo でのエラーチェックサービスの詳細については、それぞれ『NetX User Guide for the Renesas Synergy ™ Platform』または『NetX Duo User's Guide for the Renesas Synergy ™ Platform』を参照してください。*

4.3.26.3 NetX/NetX Duo SNTP クライアントモジュールの動作の概要

NetX/NetX Duo SNTP クライアントモジュールは、IP インスタンスとパケットプールを作成します。さらに、その IP インスタンスで UDP が有効化され、UDP ソケットが作成されます。UDP ソケットは、SNTP サーバーに時刻更新要求を送信して時刻データを受け取るためにウェルノウンポート 123 にバインドされますが、代替ポートでも機能します。ブロードキャストクライアントは、ブロードキャストサーバーからの送信時に UDP ポート（通常は 123）をバインドする必要があります。NetX/NetX Duo SNTP クライアントアプリケーションには、ユニキャスト動作またはブロードキャスト動作用に SNTP クライアントスレッドタスクを初期化するためのサーバーアドレスが必要です。

NetX/NetX Duo SNTP クライアントは、ユニキャストまたはブロードキャストの 2 つの基本モードのいずれかで動作し、インターネット上で時刻を取得できます。ユニキャストモードの場合、クライアントは定期的に SNTP サーバーをポーリングし、そのサーバーからの返信を受信するまで待機します。返信を受け取ると、クライアントは RFC 4330 で推奨されるサニティチェックのセットを適用し、その返信に有効な時刻更新が含まれているかどうかを検証します。次に、サーバークロックとローカルクロックの時刻に差がある場合には、時刻の差をローカルに適用します。ブロードキャストモードの場合、クライアントは単に時刻更新のブロードキャストを受信待ちし、同様のサニティチェックセットを適用して更新時刻データを検証した後、ローカルクロックを保守します。

クライアントがいずれかのモードで実行する前に、アプリケーションは絶対時間の推定値を取得できます。NTP では、1970 年 1 月 1 日午前 0 時からの秒数を意味します。例 :

```
base_seconds = 0xd2c96b90; /* This is the number of seconds since 1970
                              on Jan 24, 2012 UTC */
base_fraction = 0xa132db1e; /* between 0-1 seconds, not so critical */
```

これは必須ではありませんが、この時刻が（nx_sntp_client_set_local_time サービスを使用して）SNTP クライアントに登録されている場合、SNTP クライアントはこれを使用して最初に受信した時刻更新が妥当であるかどうかを判断できます。NTP の時計は、非常に要求の厳しいタイムベースのアプリケーション以外のすべてのアプリケーションより正確ですが、無効または不正な SNTP パケットをチェックする必要があります。SNTP クライアントを開始する前に、このベースライン時間を取得できず、ローカル時間が設定されていない場合、SNTP クライアントは時刻更新をそのままの値で受け入れます。

次に、アプリケーションは時刻更新を受信するために SNTP クライアントを準備します。そのためには、nx_sntp_client_initialize_unicast,（ユニキャストモード）または nx_sntp_client_initialize_broadcast（ブロードキャストモード）をコールします（これらは IPv4 ネットワークのみに制限されます）。NetX Duo SNTP クライアントモジュールを使用している場合、nxd_sntp_client_initialize_broadcastand、nxd_sntp_client_initialize_unicast は、IPv4 ネットワークと IPv6 ネットワークの両方をサポートするサービスとして使用できます。これらの初期化サービスコールが成功した場合、アプリケーションは nx_sntp_client_run_broadcast または nx_sntp_client_run_unicast サービスをコールすることで SNTP クライアントを開始できます。

アプリケーションでは、SNTP クライアントの動作を調整するために次のパラメータを設定できます。

- [Maximum time adjustment allowed to local clock time (milliseconds)] プロパティ。ローカルクロックの時刻を調整できる最大値です。ローカル時刻の概念がない場合、[Ignore maximum time adjust limit at startup] プロパティを有効にすることで、初回の時刻更新時にこのチェックをスキップすることができます。
- [Starting poll interval for unicast update request (seconds)] プロパティ。ユニキャストのポーリングレートです。SNTP クライアントがユニキャストモードで動作している場合に、更新のために NTP タイムサーバーをポーリングするレートです。
- [Maximum time lapse without valid update (seconds)] プロパティ。サーバーに送信された時刻要求間の許容される最大インターバルです。時刻要求に応答がない場合、SNTP クライアントはこの最大インターバルに達するまでポーリングインターバルを倍にします。
- [Invalid message limit to mark server invalid] プロパティ。サーバーを無効としてマークするまでにサーバーから受信できる無効な SNTP パケットの上限数。

これらの制限を超過しても SNTP クライアントは引き続き動作しますが、現在の SNTP サーバーのステータスが無効に設定されます。アプリケーションは、nx_sntp_client_receiving_updates サービスを定期的に呼び出して、サーバーの状態が有効であるかどうかをチェックできます。このサービスによって更新が受信されないことが示された場合、アプリケーションは nx_sntp_client_stop サービスを使用して SNTP クライアントスレッドを停止し、別の SNTP サーバーを使用する必要があります。SNTP クライアントを再起動するには、IPv4 ネットワークの場合、nx_sntp_client_initialize_broadcast と nx_sntp_client_initialize_unicast（または nxd_sntp_client_initialize_broadcast と nxd_sntp_client_initialize_unicast),）の 2 つの初期化サービスのいずれかをコールし、nx_sntp_client_run_broadcast または nx_sntp_client_run_unicast サービスで SNTP クライアントを再起動します。

ローカルクロックの動作、SNTP サニティチェック、SNTP 非同期およびユニキャスト要求の詳細については、このドキュメントの概要で説明されている『*NetX Simple Network Time Protocol (SNTP) Client User Guide for the Renesas Synergy ™ Platform*』を参照してください。

NetX/NetX Duo SNTP クライアントモジュールの動作に関する重要な注意事項と制限事項

- SNTP クライアントにコールバックを登録すると、アプリケーションで時刻更新の通知を受け取ることができます。これには、_nx_sntp_client_set_time_update_notify サービスを使用してコールバックを指定します。
- SNTP クライアントが時刻更新を受信した後、アプリケーションは SNTP クライアントに時刻更新のステータスをクエリすることができます。nx_sntp_client_get_local_time サービスを使用して NTP 時間を直接受け取ることも、_nx_sntp_client_utility_display_date_time サービスを使用してフォーマットされた日付時刻の出力を受け取ることもできます。後者の場合、SNTP クライアントが NTP 時間データを正しく処理できるよう、アプリケーションは SNTP クライアントモジュールの [Current calendar year] プロパティを設定する必要があります。
- RFC 4330 では、SNTP クライアントは、ローカルネットワークの最上位層のみにおいて、可能であれば同期のためにこれらに依存する NTP または SNTP クライアントがない構成で、運用することが推奨されています。階層レベルは、NTP 時間階層内でのホストの位置を示します。第 1 階層が最上位（ルートタイムサーバー）で第 15 階層が（クライアントごとの）許可される最下位です。SNTP クライアントのデフォルトの最小階層は 2 です。
- SNTP クライアント API によって処理される NTP 時刻更新でのローカル時刻表現の精度は、ミリ秒単位に制限されています。
- NetX/NetX Duo SNTP クライアントは、常に単一の SNTP サーバーアドレスだけを保持します。そのサーバーが有効ではなくなった場合、アプリケーションは SNTP クライアントタスクを停止し、ブロードキャストまたはユニキャストの SNTP 通信を使用して、別の SNTP サーバーアドレスでこれを再初期化する必要があります。
- NetX/NetX Duo SNTP クライアントは、多数のキャストをサポートしていません。
- NetX/NetX Duo SNTP クライアントは、受信したパケットデータを検証するための認証メカニズムをサポートしていません。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.26.4 アプリケーションへの NetX/NetX Duo SNTP クライアントモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo SNTP クライアントモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo SNTP クライアントモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo SNTP クライアントモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sntp_client0 NetX SNTP Client	Threads	New Stack> X-Ware> NetX> Protocols> NetX SNTP Client

Resource	ISDE Tab	Stacks Selection Sequence
g_sntp_client0 NetX Duo SNTP Client	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo SNTP Client

次の図に示すように、NetX/NetX Duo SNTP クライアントモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

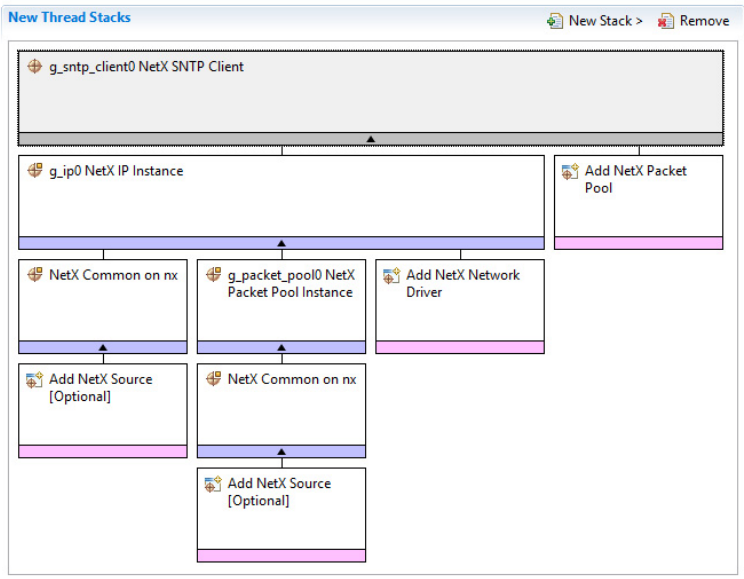


図 376:NetX/NetX Duo SNTP クライアントモジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.26.5 NetX/NetX Duo SNTP クライアントモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo SNTP クライアントモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注：次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo SNTP クライアントモジュールの構成設定

ISDE Property	Value	Description
Internal thread stack size (bytes)	2048	Internal thread stack size selection
SNTP client thread time slicing interval (ticks)	TX_NO_TIME_SLICE	SNTP client thread time slicing interval selection
Internal thread priority	2	Internal thread priority selection
UDP socket name	SNTP Client socket	UDP socket name selection
UDP port number	123	UDP port number selection
Server UDP port	123	Server UPD port selection
Time to live	128	Time to live selection
Maximum UDP packets queue depth (units)	5	Maximum UDP packets queue depth selection
Packet allocation timeout (seconds)	1	Packet allocation timeout selection
SNTP version to use	3	SNTP version to use selection
NTP minimum version	3	NTP minimum version selection

ISDE Property	Value	Description
Lowest level server stratum client accepts	2	Lowest level server stratum client accepts selection
Minimum time difference that triggers adjustment (milliseconds)	10	Minimum time difference that triggers adjustment selection
Maximum time adjustment allowed to local clock time (milliseconds)	180000	Maximum time adjustment allowed to local clock time selection
Ignore maximum time adjust limit at startup	True, False Default: True	Ignore maximum time adjust limit at startup selection
Maximum time lapse without valid update (seconds)	7200	Maximum time lapse without valid update selection
Update time remaining timer update interval (seconds)	1	Update time remaining timer update interval selection
Starting poll interval for unicast update request (seconds)	3600	Starting poll interval for unicast update request selection
Poll interval increment after failed time update	2	Poll interval increment after failed time update selection
Calculate round trip time of messages	True, False Default: False	Calculate round trip time of messages selection
**Maximum server clock inaccuracy to accept (to disable set to 0)	50000	Maximum server clock inaccuracy to accept selection
Invalid message limit to mark server invalid	3	Invalid message limit to mark server invalid selection
Randomize update request interval on startup	True, False Default: False	Randomize update request interval on startup selection
Internal Task sleep interval (ticks)	1	Internal task sleep interval selection

ISDE Property	Value	Description
Current calendar year	2016	Current calendar year selection
Name	g_sntp_client0	Module name
Index to SNTP Network Interface	0	Index to SNTP network interface selection
Name of Leap Second Handler	leap_second_handler	Name of leap second handler selection
Name of Kiss of Death Handler	kiss_of_death_handler	Name of kiss of death handler selection
Name of Random Number Generator Function (optional)	NULL	Name of random number generator function selection
Name of generated initialization function	sntp_client_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は NetX Duo でのみ使用可能なプロパティを表します。*場合によっては、スタックモジュール用デフォルト以外の設定が望ましいこともあります。たとえば、イーサネットポートに対して異なるアドレスを選択するときに役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo SNTP クライアントのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定表

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	0,0,0,0	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection

ISDE Property	Value	Description
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo SNTP クライアントモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo SNTP クライアントモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I^2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.26.6 アプリケーションでの NetX/NetX Duo SNTP クライアントモジュールの使用

一般的なアプリケーションで NetX/NetX Duo SNTP クライアントモジュールを使用する際の手順は次のとおりです。

1) nx_ip_status_check API を使用して、有効な IP アドレスを待機します。
2) nx_sntp_client_initialize_unicast API または nx_sntp_client_initialize_broadcast API（NetX Duo を使用している場合は nxd_sntp_client_initialize_unicast API または nxd_sntp_client_initialize_broadcast API）を使用して、SNTP サーバーへのアクセスモードを設定します。
3) nx_sntp_client_set_local_time API を使用してローカル時刻を設定します（オプション）。

4) nx_sntp_client_run_unicast API または nx_sntp_client_run_broadcast API を使用してクライアントを実行します。
5) nx_sntp_client_receiving_updates API を使用して SNTP が有効で、実行されていることを確認します（推奨）。
6) nx_sntp_client_get_local_time API を使用してローカル時刻を取得するか、nx_sntp_client_utility_display_date_time API を使用してローカル時刻を日付時刻フォーマットで取得します（オプション）。
7) nx_sntp_client_stop API を使用して SNTP サービスを停止します。

次の図は、一般的な手順を示した通常の動作フロー図です。

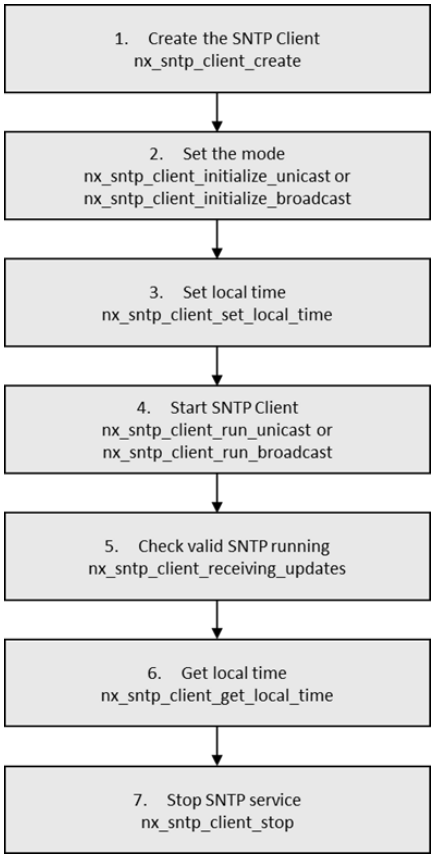


図 377:一般的な NetX/NetX Duo SNTP クライアントモジュールアプリケーションのフロー図

4.3.27 NetX/NetX Duo POP3 クライアント

ポストオフィスプロトコルバージョン 3（POP3）は、小規模なワークステーションが POP3 サーバー上のクライアントメールドロップにアクセスするためのメール転送システムを提供するために設計されたプロトコルです。POP3 は、伝送制御プロトコル（TCP）サービスを利用してメール転送を実行します。

クライアントメールは、POP3 サーバー上のメールボックス（「メールドロップ」）に格納されます。メールドロップは、インデックスが 1 から始まるメールアイテムの配列として表されます。各メールは、メールドロップのインデックスを使用して取得および削除されます。メールメッセージがダウンロードされると、通常、クライアントはサーバーのメールボックスでそのメールアイテムを削除対象としてマークします（削除はサーバーが自動的に実行すると考えられます）。メールアイテムの取得が完了したら、アプリケーションは POP3 クライアントインスタンスを削除し、TCP 接続を閉じる必要があります。メールを再取得するには、別の POP3 クライアントを作成しなければなりません。

注 : 特に説明がないかぎり、NetX Duo POP3 クライアントモジュールは POP3 セッションの応用、設定、実行が NetX POP3 クライアントモジュールと同じです。NetX Duo で IPv6 用に IP インスタンスを設定する方法については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。

4.3.27.1 NetX/NetX Duo POP3 クライアントモジュールの特長

- NetX Client POP3 は RFC 1939 に準拠しています。
- 以下を行うためのハイレベル API を提供します。
  - POP3 クライアントインスタンスを作成および削除する
  - 受信するメールメッセージの数とサイズをクエリする
  - 個々のメールメッセージを受信する
  - サーバーのメールドロップボックスからメールメッセージを削除する

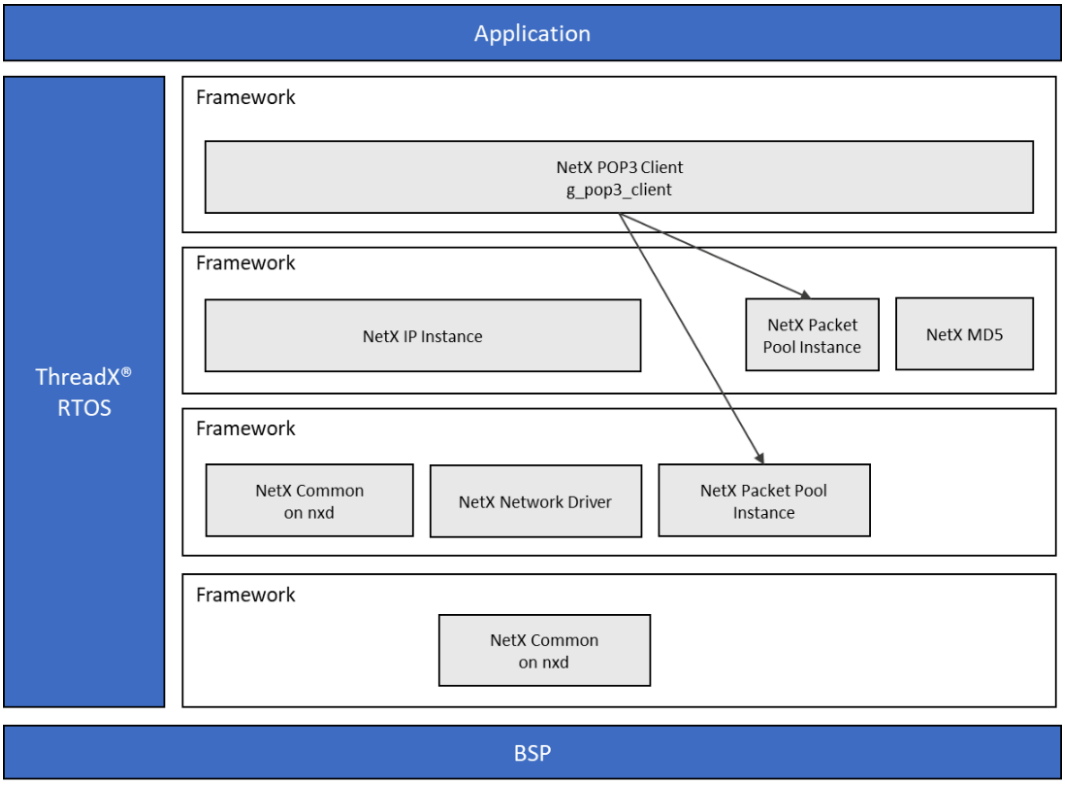


図 378:NetX/NetX Duo POP3 クライアントモジュールのブロック図

注 : 上の図で、NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.27.2 NetX/NetX Duo POP3 クライアントモジュールの API の概要

NetX POP3 クライアントでは、POP3 インスタンスの作成、削除、およびそのメッセージの取得のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。

NetX/NetX Duo POP3 クライアントモジュールの API の要約

Function Name	Example API Call and Description
nx_pop3_client_create	nx_pop3_client_create(&demo_client, NX_FALSE /* disable APOP authentication */, &client_ip, &client_packet_pool, POP3_SERVER_ADDRESS, POP3_SERVER_PORT, LOCALHOST, LOCALHOST_PASSWORD); Create a POP3 Client Instance for IPv4 only.
**nxd_pop3_client_create	nxd_pop3_client_create(&demo_client, NX_FALSE /* disable APOP authentication */, &client_ip, &client_packet_pool, *server_ip_address, ULONG server_port, CHAR *client_name, CHAR *client_password); Create a POP3 Client Instance for either IPv4 or IPv6
nx_pop3_client_delete	nx_pop3_client_delete (&demo_client); Delete a POP3 Client Instance
nx_pop3_client_mail_item_delete	nx_pop3_client_mail_item_delete(&demo_client, item_index); Delete a Client mail item for a Server maildrop.
nx_pop3_client_mail_item_get	nx_pop3_client_mail_item_get (&demo_client, 1, &item_size); Retrieve a specific mail message size.

Function Name	Example API Call and Description
nx_pop3_client_mail_items_get	nx_pop3_client_mail_item_get (&demo_client, 1, &number_mail_items, &maildrop_total_size); Obtain the number of mail items in a maildrop.
nx_pop3_client_mail_item_message_get	nx_pop3_client_mail_item_message_get (&demo_client, &recv_packet_ptr, &bytes_retrieved, &final_packet); Download a specific mail message.
nx_pop3_client_mail_item_size_get	nx_pop3_client_mail_item_size_get (&demo_client, mail_item, &size); Obtain the size of a specific mail item.
nx_pop3_client_quit	nx_pop3_client_quit(&demo_client); Sends a QUIT command to the POP3 server.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :** この API は、NetX Duo POP3 クライアントでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

ステータス戻り値

Name	Description
NX_SUCCESS	API Call Successful
NX_PTR_ERROR*	Invalid input pointer parameter
NX_POP3_CLIENT_INVALID_INDEX*	Null mail index input
NX_POP3_PARAM_ERROR	Invalid non-pointer input
NX_POP3_APOP_FAILED_MD5_DIGEST	POP3 Client failed APOP authentication
NX_POP3_INVALID_MAIL_ITEM	Invalid mail item index (exceeds number of items in the maildrop box)

Name	Description
NX_POP3_INSUFFICIENT_PACKET_PAYLOAD	Client packet payload too small for POP3 request
NX_POP3_SERVER_ERROR_STATUS	Server replies with error status
NX_POP3_CLIENT_INVALID_STATE	Client not initialized to receive mail messages

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

注 : これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX でのエラーチェックサービスの詳細については、『NetX User Guide for the Renesas Synergy ™ Platform』を参照してください。*

4.3.27.3 NetX/NetX Duo POP3 クライアントモジュールの動作の概要

POP3 プロトコルでは、クライアントが POP3 セッションの状態を維持する必要があります。POP3 クライアントには、RFC 1939 で定義される 3 つの異なる状態があります。初期状態は認証状態で、サーバーに自身を識別させる必要があります。次がトランザクション状態で、クライアントはメールをダウンロードします。POP3 クライアントがセッションの終了を選択すると更新状態となり、サーバーから切断します。

NetX POP3 クライアントでは、POP3 メッセージをサーバーに送信するために、以前に作成した NetX IP インスタンスとパケットプールが必要です。NetX POP3 クライアントは、POP3 メッセージを送信するためのパケットプールを必要とします。IP インスタンスが使用するのと同じパケットプールを使用することも、NetX で別のパケットプールを作成することもできます。サーバーへの POP3 クライアントメッセージは、単純な POP3 コマンドとログイン認証データに限定されます。つまり、NetX POP3 クライアントが独自のパケットプールを使用している場合、POP3 クライアントのユーザー名とパスワードの長さに応じて、ペイロードサイズを 150 ～ 200 バイトをはるかに超えるほど大きくする必要はありません。これに対し、IP インスタンスが使用するパケットプールには、最大デバイス MTU（通常は 1518 バイト）の POP3 メールデータを受信するのに十分な大きさのペイロードが必要です。

NetX POP3 クライアントは TCP サービスを利用するため、IP インスタンスで TCP を有効化する必要があります。

NetX/NetX Duo POP3 クライアントモジュールの動作に関する重要な注意事項と制限事項

NetX Client POP3 は RFC 1939 に準拠しています。

自動生成された Synergy コードは、起動 / 初期化時にこれらのタスクを処理します。IP インスタンスとパケットプールの作成、および TCP サービスの有効化は自動的に行われます。

POP3 クライアントプロパティ *[Auto initialization]* のデフォルトは [Disable] です。このプロパティを有効にしないでください。このプロパティが有効になっていると、初期化中、ネットワークリンクが有効になる前に、自動生成コードがサーバーに接続しようとして失敗します。これにより、アプリケーションの初期化が中止されます。

[Auto initialization] が無効の場合、アプリケーションがサーバーへの接続とメールの取得を処理します。

サーバーへの接続には次のプロパティが必要です。

- APOP authentication：デフォルトは無効。ユーザー名とパスワードを暗号化されたテキストでやり取りできるようにします。パスワードなどのデータが公開されるのを防ぐため、利用が推奨されます。
- Server port number：デフォルトは RFC 1939 で推奨される 110 ですが、任意の値を指定できます。
- *Server_IPv4 Address：IPv4 接続を使用している場合に接続するサーバーです。*
- Server_IPv6 *Address：IPv6 接続を使用している場合に接続するサーバーです。*

- User server address type：アプリケーションでの使用に応じて IPv4 または IPv6 に設定します。
- Client Name：サーバーに POP3 クライアントを識別させるためのユーザー名。
- Client Password：サーバーが必要とする場合に、以前に合意したパスワード。

もっとも、POP3 クライアントの構造上、これらのプロパティの設定にはアプリケーションからアクセスできません。アプリケーションは nx_pop3_client_create API を使用してサーバーに接続し、これらのプロパティを直接入力する必要があります。

```
UINT  nx_pop3_client_create(NX_POP3_CLIENT *client_ptr,
                            UINT APOP_authentication, NX_IP *ip_ptr,
                            NX_PACKET_POOL *packet_pool_ptr,
                            ULONG server_ip_address,
                            ULONG server_port,
                            CHAR *client_name,
                            CHAR *client_password)
```

ほとんどのアプリケーションでの一般的なコールは次のようになります。

```
Status =  nx_pop3_client_create(&g_pop3_client0, NX_TRUE, &g_ip0,
                            &g_packet_pool0, /* If sharing IP packet pool */
                            IP_ADDRESS(192,168,0,2,
                            110,
                            "myusername",
                            "mypassword")
```

NetX Duo の場合、ULONG server_ip_address は NXD_ADDRESS 型へのポインタに置き換えられます。

```
NXD_ADDRESS server_address;

server_address.nxd_ip_version = NX_IP_VERSION_IPv4;  /* e.g. 4 */

/* If the local IP address is 192.168.0.2, use the IP_ADDRESS macro to convert t
o a ULONG */
server_address.nxd_ip_address.v4 = IP_ADDRESS(192,168,0,);

status =  nxd_pop3_client_create(&g_pop3_client0, NX_TRUE, &g_ip0,
                            &g_packet_pool0, /* If sharing IP packet pool */
                            &server_address,
                            110,
                            "myusername",
                            "mypassword")
```

この関数は、POP3 クライアントインスタンスを作成し、TCP ソケットを作成してローカル TCP ポートにバインドします。次に、POP3 サーバーの TCP ソケットに接続を試みます。成功すると、ユーザーパスワードでサーバーに認証されます。POP3 認証の詳細については、このセクションで後ほど説明します。すべてが成功すると、POP3 クライアントはメールを受け取れる状態になります。

アプリケーションは、nx_pop3_client_mail_items_get API を使用して受信トレイにあるメールアイテムの数をサーバーにクエリすることができます。メールメッセージの合計バイト数も返されますが、これは概算と考えてください。アプリケーションは nx_pop3_client_mail_item_size_get API を使用して各メールアイテムのサイズをさらにクエリす

ることができます。実際のサイズは異なる可能性があるため、これも概算と考えてください。メールドロップの最初のメールアイテムのインデックスはゼロではなく、1 であることに注意してください。

メールアイテムを実際にダウンロードするときは、アプリケーションは、nx_pop3_client_get_mail_item サービスを使用してメールアイテムのインデックスを指定することで、受信したいメールアイテムをサーバーに示します。これによりメールアイテムのサイズも返されますが、POP3 クライアントはここで nx_pop3_client_mail_item_get_message_data. をコールしてメッセージテキストを取得できます。この API は一度に 1 つのパケットを受信します。このため、アプリケーションは、メッセージを含む最後のパケットが受信されるまで、1 回または複数回コールする必要があります。以下に示すように、サーバーは final_packet ポインタでクライアントに最後のパケットを示します。

```
/* Find out how many items are in our mailbox.  */
status = nx_pop3_client_mail_items_get(&demo_client, &number_mail_items,
                                       &total_size);
mail_item = 1;

/* Download all mail items.  */
while (mail_item <= number_mail_items)
{

    /* Submit a request for the next mail item. */
    status = nx_pop3_client_mail_item_get(&demo_client, mail_item,
                                          &mail_item_size);

    /* Loop to get the next mail message until it is completely downloaded. */
    do
    {
        status = nx_pop3_client_mail_item_message_get(&demo_client, &packet_ptr,
                                                      &bytes_retrieved, &final_packet
);

        /* Determine if this is the last data packet. */
        if (final_packet)
             status = nx_pop3_client_mail_item_delete(&demo_client, mail_item);

    } while (final_packet == NX_FALSE);

    /* Get the next mail item. */
    mail_item++;
}
```

各パケットを受信した後、アプリケーションはパケットデータをバッファにコピーし、パケットをパケットプールに戻す必要があります（上には示されていません）。これにより、パケットの受信に使用されるパケットプールが不足するのを防ぐことができます。メッセージをダウンロードした後は、メールアイテムを削除対象としてマークするのが一般的です。このため、アプリケーションは削除するメールアイテムのインデックスを指定して nx_pop3_client_mail_item_delete を呼び出します。

メールアイテムの送信が終了したら、クライアントは nx_pop3_client_quit. をコールしてセッションを終了します。これにより、TCP セッションが終了します。POP3 をこれ以上使用しない場合、アプリケーションは nx_pop3_client_delete. をコールして TCP ソケットを削除できます。他のタスクが POP3 クライアントパケットプールを使用していない場合、アプリケーションは nx_packet_pool_delete サービスを使用してパケットプールを削除できます。

POP3 認証

TCP 接続が確立された後、POP3 クライアントはメールボックスにアクセスするために認証によって POP3 サーバーに自身を識別させる必要があります。認証は、サーバーにユーザー名とパスワードを送信することによって実行されます。ユーザー名は通常、完全修飾ドメイン名（ローカル部分とドメイン名を「@」文字で区切ったもの）です。

APOP 認証を有効にするには、nx_pop3_client_create/nxd_pop3_client_create API（NetX Duo なら nxd_pop3_client_create API）で *APOP 認証*の入力を NX_TRUE に設定します。APOP 認証は、nx_pop3_client_create/nxd_pop3_client_create コールに指定されたユーザー名 / パスワードの MD5 ダイジェストを作成し、ユーザー名とパスワードをクリアテキストで送信するというセキュリティリスクを回避します。APOP 認証が失敗すると、NetX POP3 クライアントはユーザー名とパスワードを暗号化なしで使用してログインしようとします。

- NetX POP3 クライアントは AUTH コマンドをサポートしていませんが、ダイジェスト MD5 を使用して APOP 認証をサポートしています。
- NetX POP3 クライアントは、すべての POP3 コマンド（TOP コマンドや UIDL コマンドなど）を実装しているわけではありません。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.27.4 アプリケーションへの NetX/NetX Duo POP3 クライアントモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo POP3 クライアントモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo POP3 クライアントモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo POP3 クライアントモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_pop3_client0 NetX POP3 Client	Threads	New Stack> X-Ware> NetX> Protocols> NetX POP3 Client
g_pop3_client0 NetX POP3 Client	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo POP3 Client

次の図に示すように、NetX/NetX Duo POP3 クライアントモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。

（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

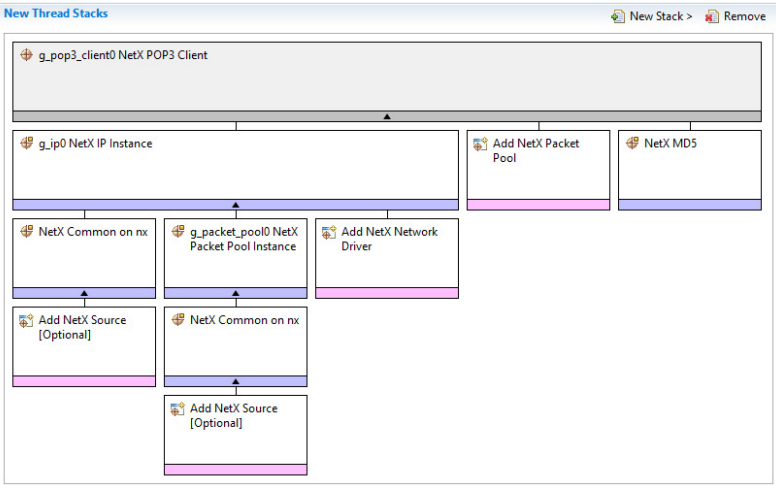


図 379:NetX/NetX Duo POP3 クライアントモジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.27.5 NetX/NetX Duo POP3 クライアントモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo POP3 クライアントモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注：次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo POP3 クライアントモジュールの構成設定

ISDE Property	Value	Description
Maximum buffer size to store messages (bytes)	2000	Maximum buffer size to store messages selection
Packet time out (seconds)	1	Packet time out selection
Connection time out (seconds)	30	Connection time out selection
Disconnect time out (seconds)	2	Disconnect time out selection
TCP socket send wait (seconds)	2	TCP socket send wait selection
Server reply timeout (seconds)	10	Server reply timeout selection
TCP window size (bytes)	1460	TCP window size selection
Maximum user name length (bytes)	40	Maximum user name length selection
Maximum password length (bytes)	20	Maximum password length selection
Name	g_pop3_client0	Module name
APOP Authentication	Enable, Disable Default: Disable	Apop authentication selection
**Use server address type	IPv4, IPv6 Default: IPv6	Use server address type selection
Server IPv4 Address (use commas for separation)	192, 168, 0, 2	Server IPv4 Address selection
**Server IPv6 Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	Server IPv6 Address selection
Server Port number	110	Server Port number selection

ISDE Property	Value	Description
Client Name	mailto:username@domain.com	Client name selection
Client Password	password	Client password selection
Auto initialization	Enable, Disable Default: Disable	Auto initialization selection
Name of generalized initialization function	pop3_client_init0	Name of generalized initialization function selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

ローレベルモジュールのデフォルト以外の設定が望ましいこともあります。たとえば、異なる MAC アドレスまたは IP アドレスの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo POP3 クライアントのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
Default Gateway Address (use commas for separation)	0,0,0,0	Default gateway address selection

ISDE Property	Value	Description
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection

ISDE Property	Value	Description
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function
Link status change callback	NULL	Link status change callback selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection

ISDE Property	Value	Description
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX MD5 インスタンスの構成設定

ISDE Property	Value	Description
No configurable properties()	-	-

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo POP3 クライアントモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo POP3 クライアントモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.27.6 アプリケーションでの NetX/NetX Duo POP3 クライアントモジュールの使用

アプリケーションで NetX および NetX Duo BSD サポートモジュールを使用する際の一般的な手順は次のとおりです。

手順 1：nx_ip_status_check API を使用して、ネットワークリンクが有効になるのを待機します。

手順 2：nx_pop3_client_create API（NetX Duo POP3 クライアントでは (nxd_pop3_client_create）を使用して、POP3 クライアントを作成します。

手順 3：nx_pop3_client_mail_items_get API を使用して POP3 サーバーにメールボックス内のメールアイテム数を要求します。

手順 4：POP3 クライアントのメールドロップに 1 つ以上のアイテムがあると想定し、nx_pop3_client_mail_item_get API をコールすることで POP3 サーバーに 1 つまたはすべてのアイテムを要求します。

手順 5：各 nx_pop3_client_mail_item_get, をコールした後、nx_pop3_client_mail_message_get API を使用して実際のメールメッセージデータをダウンロードします。

手順 6：パケットデータを別のバッファにコピーし、nx_packet_release API を使用して受信 packet(s) を解放します（パケットプールの不足を避けるため強く推奨されます）。

手順 7：これがメッセージの最後のパケットであるかどうかをチェックします。最後のパケットであれば、nx_pop3_client_mail_message_get の final_packet ポインタ入力が TRUE になります。

FALSE の場合、再び nx_pop3_client_mail_message_get をコールします。TRUE の場合、手順 8 に進みます。

手順 8：nx_pop3_client_mail_item_delete API を使用して、現在のメールアイテムを削除対象としてマークします。これにより、DELE メッセージがサーバーに送信され、サーバーによって後でメールアイテムが削除されます。

手順 9：ダウンロードするメールアイテムがさらにあるかどうかをチェックします。ある場合は、nx_pop3_client_mail_item_get. をコールして次のメールアイテムを取得します。ない場合は、手順 10 に進みます。

手順 10：nx_pop3_client_delete API を使用してサーバーに QUIT メッセージを送信します。

手順 11：nx_pop3_client_delete をコールして POP3 インスタンスを削除します。

次の図は、一般的な手順を示した通常の動作フロー図です。

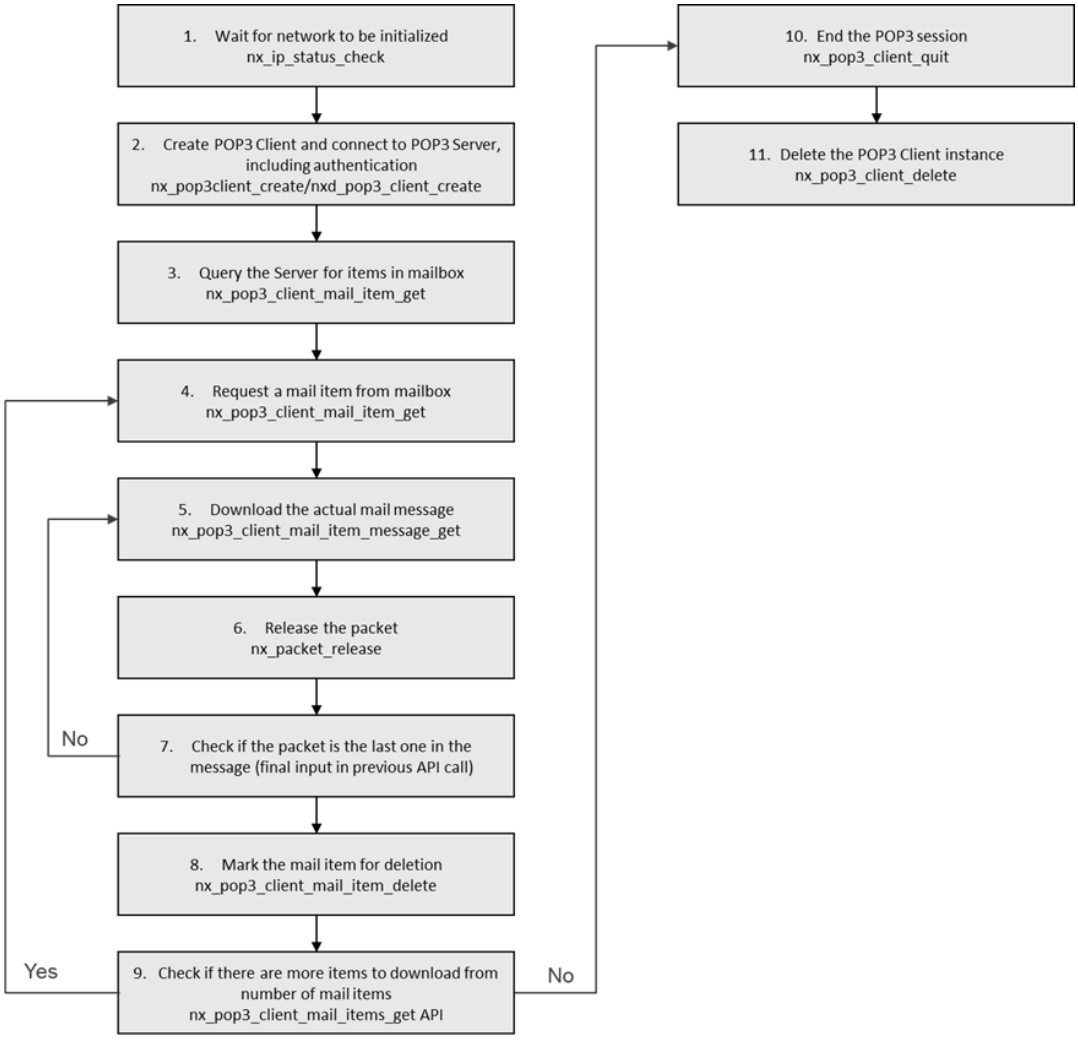


図 380:一般的な NetX/NetX Duo POP3 クライアントモジュールアプリケーションのフロー図

4.3.28 NetX/NetX Duo Telnet クライアント

Telnet プロトコル（Telnet）は、インターネット上の 2 つのノード間でコマンドとレスポンスを転送するために設計されています。Telnet は、信頼性の高い伝送制御プロトコル（TCP）サービスを利用して転送機能を実行するシンプルなプロトコルです。NetX ™ Telnet クライアントは、Telnet クライアントの実装を必要とするアプリケーションにハイレベルの API を提供します。

注 :NetX Duo ™ Telnet クライアントは、内部処理を除けば Telnet セッションの応用、設定、実行が NetX ™ Telnet クライアントとほぼ同じです（特に説明がない場合）。

4.3.28.1 NetX/NetX Duo Telnet クライアントモジュールの特長

- NetX ™ Telnet クライアントモジュールは、RFC854 および関連する RFC に準拠しています。

- 以下を行うためのハイレベル API を提供します。
  - Telnet クライアントインスタンスの作成および削除
  - Telnet クライアントインスタンスの接続および切断
  - Telnet サーバーに対するパケットの送信と受信
  - マルチスレッド動作のサポート

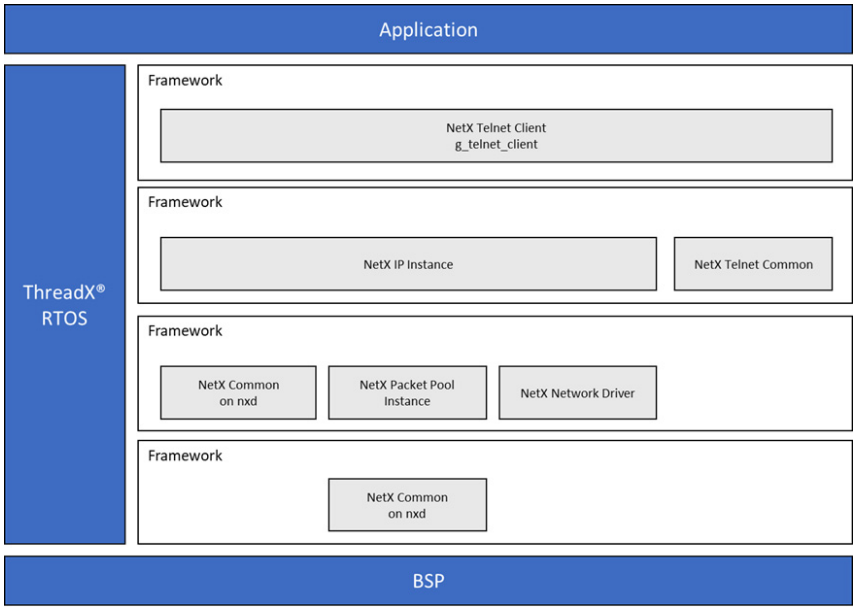


図 381:NetX/NetX Duo Telnet クライアントモジュールのブロック図

注 : 上の図で、NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.28.2 NetX/NetX Duo Telnet クライアントモジュールの API の概要

NetX ™ Telnet クライアントモジュールでは、Telnet 通信の作成、削除、接続、切断、受信、送信のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX/NetX Duo Telnet クライアントモジュールの API の要約

Function Name	Example API Call and Description
nx_telnet_client_connect	nx_telnet_client_connect(&g_telnet_client0, server_ip_address, server_port, NX_WAIT_FOREVER); Connect a Telnet Server. Supports only IPv4.
**nxd_telnet_client_connect	nxd_telnet_client_connect(&g_telnet_client0, &server_ip_address_v6, server_port, NX_WAIT_FOREVER); Connect to a Telnet Server. Supports IPv4 and IPv6.
nx_telnet_client_create	nx_telnet_client_create(&g_telnet_client0, “Telnet Client", &g_ip0, 1024); Create a Telnet Client.
nx_telnet_client_delete	nx_telnet_client_delete(&g_telnet_client0); Delete a Telnet Client.
nx_telnet_client_disconnect	nx_telnet_client_disconnect(&g_telnet_client0, NX_WAIT_FOREVER); Disconnect from the Telnet Server (IPv4 or IPv6).
nx_telnet_client_packet_receive	nx_telnet_client_packet_receive(&g_telnet_client0, &packet_ptr, NX_WAIT_FOREVER); Receive packet from the Telnet Server.
nx_telnet_client_packet_send	nx_telnet_client_packet_send(&g_telnet_client0, packet_ptr, NX_WAIT_FOREVER); Send packet to the Telnet Server (IPv4 or IPv6).

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :** この API は、NetX Duo Telnet クライアントでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

ステータス戻り値

Name	Description
NX_SUCCESS	API Call Successful
NX_TELNET_ERROR	Error while creating TCP socket as part of creating the Telnet Client instance
NX_TELNET_NOT_CONNECTED	Client not disconnected
NX_TELNET_NOT_DISCONNECTED	Client socket is not in closed state (cannot make a TCP connection; cannot delete the Telnet Client if the socket is still connected).
NX_TELNET_INVALID_PARAMETER*	Invalid non-pointer input to Telnet Client create
NX_PTR_ERROR*	Invalid pointer input
NX_IP_ADDRESS_ERROR*	Invalid IP address to connect to Telnet Server
NX_CALLER_ERROR*	Invalid caller of this service

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

注 : これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX Duo でのエラーチェックサービスの詳細については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

4.3.28.3 NetX/NetX Duo Telnet クライアントモジュールの動作の概要

NetX ™ Telnet クライアントモジュールでは、NetX ™の IP スレッドタスクの作成および実行、Telnet クライアントの作成、Telnet サーバーに接続するための TCP ソケットの使用準備が自動的に行われます。Telnet クライアントは、nx_telnet_client_connect API をコールすることでサーバーに接続します（この API は、NetX Duo ™ Telnet で使用でき、IPv4 TCP 接続に限定されています。NetX Duo ™では、アプリケーションは IPv4 と IPv6 の両方をサポートする nxd_telnet_client_connect も使用できます）。これが成功したら、Telnet クライアントは nx_telnet_client_packet_receive API を使用して自身を通知するサーバーの「バナー」を受け取るのを待つ必要があります。その後、Telnet クライアントは、nx_packet_allocate および nx_packet_data_append API を使用して、1 文字のデータを持つパケットを作成できます。これらのパケットは、nx_telnet_client_packet_send API をコールすることで Telnet サーバーに送信されます。

NetX ™ Telnet クライアントおよび NetX Duo ™ Telnet クライアントの動作の詳細については、関連するユーザーガイドを参照してください。このガイドは、Synergy ギャラリーで SSP の [Documentation] タブに「X-Ware ™ and NetX ™ Component Documentation for Renesas Synergy」として掲載されています。詳細については関連するユーザーガイ

ドにアクセスしてください。このドキュメントでは、コンポーネントの動作の概要のみ説明します。モジュール選択、構成、使用例の詳細については、以下のセクションを参照してください。

NetX/NetX Duo Telnet クライアントモジュールの動作に関する重要な注意事項と制限事項

- NetX Duo ™ Telnet クライアントモジュールサービスは、複数のスレッドから同時にコールすることができます。ただし、特定の Telnet クライアントインスタンスに対するリードまたはライト要求は、同じスレッドから順番に行う必要があります。
- Telnet クライアントは、Telnet ネゴシエーションをサポートしていないか、IAC およびコマンドコードシーケンスを送信しません。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.28.4 アプリケーションへの NetX/NetX Duo Telnet クライアントモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo Telnet クライアントモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo Telnet クライアントモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo Telnet クライアントモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_telnet_client0 NetX™ Telnet Client	Threads	New Stack> X-Ware> NetX™> Protocols> NetX™ Telnet Client
g_telnet_client0 NetX Duo™ Telnet Client	Threads	New Stack> X-Ware> NetX Duo™> Protocols> NetX Duo™ Telnet Client

次の図に示すように、NetX/NetX Duo Telnet クライアントモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

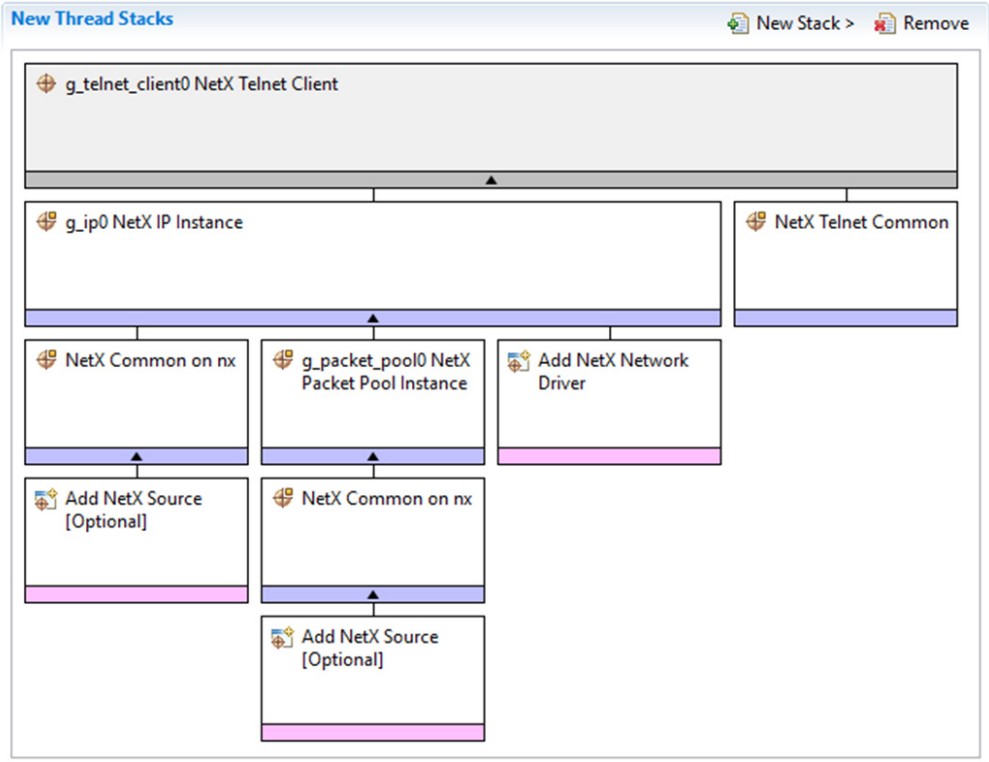


図 382:NetX/NetX Duo Telnet クライアントモジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.28.5 NetX/NetX Duo Telnet クライアントモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo Telnet クライアントモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo Telnet クライアントモジュールの構成設定

ISDE Property	Value	Description
Name	g_telnet_client0	Module name
TCP Socket Window Size in Bytes	1024	TCP socket window size selection
Name of generated initialization function	telnet_client_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、異なる IP アドレスおよびサブネットマスクの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo Telnet クライアントのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection

ISDE Property	Value	Description
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
Default Gateway Address (use commas for separation)	0,0,0,0	Default gateway address selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection

ISDE Property	Value	Description
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function
Link status change callback	NULL	Link status change callback selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo Telnet 共通インスタンスの構成設定

ISDE Property	Value	Description
Type of Service for UDP requests	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost Default: Normal	Type of service UDP requests selection
Fragmentation option	Don't fragment, Fragment okay Default: Don't fragment	Fragment option selection
Server TCP port number	23	Server TCP port number selection
Time to live	128	Time to live selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo Telnet クライアントモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo Telnet クライアントモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin Selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.28.6 アプリケーションでの NetX/NetX Duo Telnet クライアントモジュールの使用

NetX ™ Telnet クライアントモジュールは、アプリケーションによる通常の初期化を必要としません。コンフィギュレータが初期化プロセスを生成します。ユーザーアプリケーションに必要なのは Telnet 通信の処理だけです。

一般的なアプリケーションで NetX および NetX Duo Telnet クライアントモジュールを使用する際の手順は次のとおりです。

1) nx_ip_status_check API を使用して、IP インスタンスが初期化され、アプリケーションが NetX ™サービスの使用を開始できるかどうかを確認します。
2) nx_telnet_client_connect API を使用して、Telnet サーバーに接続します（注：NetX Duo ™の場合、nxd_telnet_client_connect). API が推奨されます）。
3) nx_telnet_client_packet_receive API を使用して、Telnet サーバーのウェルカムバナーを受け取ります（オプション）。
4) nx_packet_allocate および nx_packet_data_append API で送信するパケットを作成します。
5) nx_telnet_client_packet_send API を使用して、パケットを送信します。
6) nx_telnet_client_receive API を使用して、サーバーのレスポンスパケットを受信します。
7) nx_telnet_client_disconnect API を使用して通信を切断します。
8) 送信と受信が終了したら、nx_telnet_client_delete API を使用してインスタンスを削除します。

次の図は、一般的な手順を示した通常の動作フロー図です。

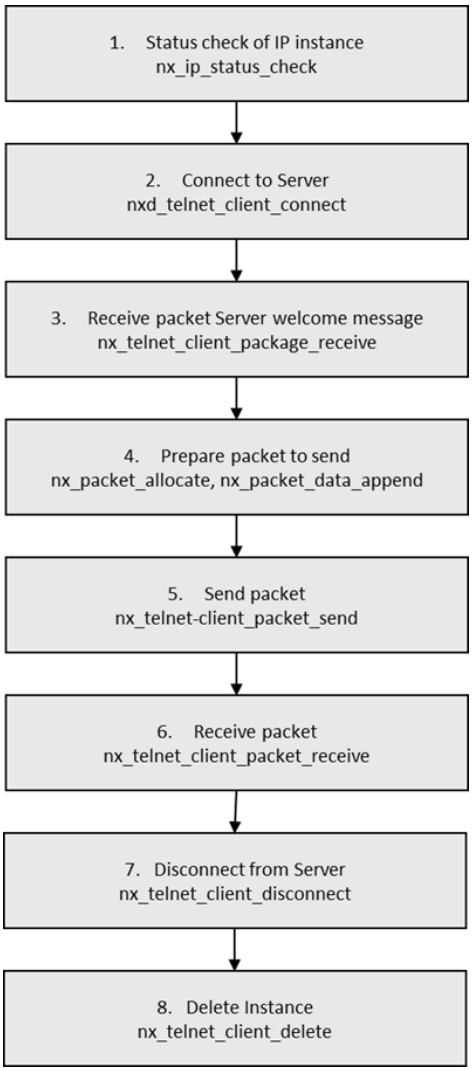


図 383:一般的な NetX/NetX Duo Telnet クライアントモジュールアプリケーションのフロー図

4.3.29 NetX/NetX Duo Telnet サーバー

Telnet プロトコル（Telnet）は、インターネット上の 2 つのノード間でコマンドとレスポンスを転送するために設計されたプロトコルです。Telnet は、信頼性の高い伝送制御プロトコル（TCP）サービスを利用して転送機能を実行するシンプルなプロトコルです。

注 :NetX Duo™ Telnet サーバーは、内部処理を除けば Telnet セッションの応用、設定、実行が NetX™ Telnet サーバーとほぼ同じです（特に説明がない場合）。

4.3.29.1 NetX/NetX Duo Telnet サーバーモジュールの特長

- NetX Telnet サーバーモジュールは、RFC854 および関連する RFC に準拠しています。
- 以下を行うためのハイレベル API を提供します。
  - マルチスレッド動作のサポート
  - 共通関数のコールバックをサポート
  - 認証
  - 新規接続
  - データ受信
  - 接続終了
- オプションネゴシエーションをサポートしています。
  - Echo
  - Go Ahead 抑止

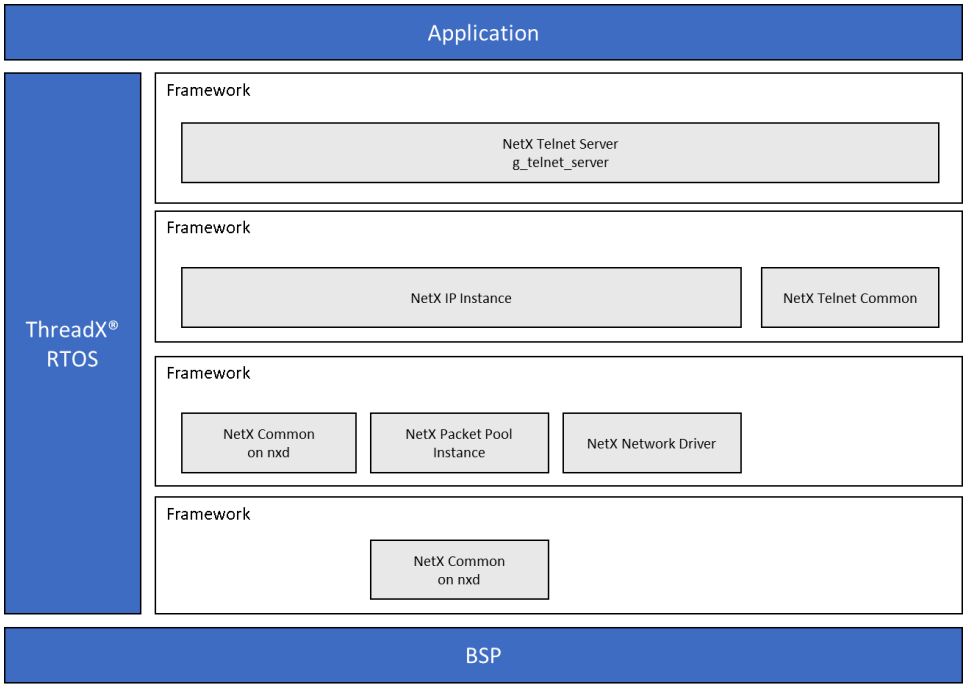


図 384:NetX/NetX Duo Telnet サーバーモジュールのブロック図

注:上の図で、NetX(または NetX Duo)ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.29.2 NetX/NetX Duo Telnet サーバーモジュールの API の概要

NetX Telnet サーバーモジュールでは、パケットの作成、削除、送信、および開始と停止のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX/NetX Duo Telnet サーバーモジュールの API の要約

Function Name	Example API Call and Description
nx_telnet_server_create	nx_telnet_server_create(&my_server, "Telnet Server", &ip_0,pointer, 2048, telnet_new_connection, telnet_receive_data, telnet_connection_end); Create a Telnet Server.
nx_telnet_server_delete	nx_telnet_server_delete(&my_server); Delete a Telnet Server.
nx_telnet_server_disconnect	nx_telnet_server_disconnect(&my_server, 2); Disconnect the Telnet Client specified by the client list index (2nd input).
nx_telnet_server_get_open_connection_count	nx_telnet_server_get_open_connection_count(&my_server, &conn_count); Retrieve the number of open connections.
nx_telnet_server_packet_send	nx_telnet_server_packet_send(&my_server, 2, my_packet, 100); Send packet to Telnet Client specified by client list index (second input).

Function Name	Example API Call and Description
nx_telnet_packet_pool_set	nx_telnet_server_packet_pool_set(&my_server, &telnet_server_packet_pool); Set packet pool as Telnet Server packet pool.
nx_telnet_server_start	nx_telnet_server_start(&my_server); Start theTelnet Server.
nx_telnet_server_stop	nx_telnet_server_stop(&my_server); Stop the Telnet Server.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

ステータス戻り値

Name	Description
NX_SUCCESS	Successful telnet function
NX_PTR_ERROR*	Invalid Server, IP, stack, or application callback pointers
NX_CALLER_ERROR*	Invalid caller of this service
NX_OPTION_ERROR*	Invalid logical connection
NX_IP_ADDRESS_ERROR*	Invalid IP address
NX_TELNET_FAILED	Server packet send failed
NX_TELNET_NO_PACKET_POOL	Cannot start Telnet Server, no packet pool available
NX_TELNET_NOT_CONNECTED	Cannot disconnect Telnet Server because it is not connected
NX_TELNET_NOT_DISCONNECTED	Cannot connect or delete Telnet Server because it is not disconnected

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

注 : これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX Duo でのエラーチェックサービスの詳細については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。

4.3.29.3 NetX/NetX Duo Telnet サーバーモジュールの動作の概要

NetX Telnet パッケージでは、NetX IP インスタンスが既に作成されている必要があります。加えて、同じ IP インスタンスで TCP が有効になっている必要があります。NetX Telnet パッケージのうち Telnet クライアントの部分には、追加要件はありません。また、すべての Telnet クライアント要求を処理するために、TCP のウェルノウンポート 23 への完全なアクセスが必要です。Telnet サーバーはクライアント接続のリストを保持し、このリストにインデックスを使用して、必要なときに特定のクライアントを指定します。このリストのサイズは、[Maximum clients to server simultaneously] プロパティで設定します。

Telnet サーバーは、Telnet クライアントとの限定されたオプションネゴシエーションを有効にすることができます。この機能を有効にするには、オプションネゴシエーションを有効に設定します。この機能を有効にした場合、Telnet サーバーにパケットプールが必要です。アプリケーションは、パケットペイロードとパケット数をそれぞれ packet_size_inthepool と Totalpacket_pool_sizeProperties, で設定し、Telnet サーバーにパケットプールを作成させることができます。Telnet サーバーが削除されると、パケットプールもともに削除されます。

また、パケットプールを直接作成し、Telnet サーバーパケットプールとして設定できます。これには、1）[Use application packet pool] を有効にするか、2）NetXnx_packet_pool_create API をコールしてパケットプールを作成し、3）nx_telnet_server_packet_pool_set API を使用して Telnet サーバーでパケットプールを設定します。Telnet サーバーが削除されたら、アプリケーションはパケットプールを直接削除する必要があります（nx_packet_pool_delete API）。

Telnet 新規接続コールバック

NetX Telnet サーバーは、新しい Telnet クライアント要求を受信すると新規接続コールバック関数をコールします。このコールバック関数は NetX Telnet サーバーの [Name of Client Connect Callback Function] プロパティで設定されます。新規接続コールバックのアクションには、クライアントへのバナーまたはプロンプトの送信が含まれます。認証が必要な場合は、ログイン情報を求めるプロンプトも含まれます。新規接続コールバックの 2 番目の引数は、クライアントが接続していることを指定します。

Telnet データ受信コールバック

NetX Telnet サーバーモジュールは、新しい Telnet クライアントデータを受信するとデータ受信コールバック関数をコールします。このコールバックの 2 番目の入力は、Telnet サーバーのクライアントリストへのインデックスです。このため、Telnet サーバーは切断すべきクライアントを認識できます。このコールバック関数は [Name of Receive Data Callback Function] プロパティで設定されます。データ受信コールバックの一般的なアクションには、クライアントからのコマンドを解釈するため、データのエコーバック、データの解析のいずれかまたは両方、およびデータの提供が含まれます。

注 : このコールバックルーチンは受信パケットを解放する必要があります。

Telnet 接続終了コールバック

NetX Telnet サーバーは、Telnet クライアント切断要求を受信すると接続終了コールバック関数をコールします。このコールバックの 2 番目の入力は、Telnet サーバーのクライアントリストへのインデックスです。このため、Telnet サーバーは切断すべきクライアントを認識できます。このコールバック関数は [Name of Client Disconnect Callback Function] プロパティで設定されます。接続終了コールバックの一般的なアクションには、Telnet クライアントセッションで使用されるリソースのクリーンアップが含まれます。

Telnet オプションネゴシエーション

Telnet クライアントとの接続時に、クライアントからオプション要求を受信していない場合、Telnet サーバーは、この一連の Telnet オプションをクライアントに送信します。

```
will echo
don't echo
```

```
will sga
```

クライアントから Telnet データを受信すると、Telnet サーバーは最初のバイトが IAC（Interpret as Command）コードであるかどうかを確認します。そうであれば、クライアントパケットのすべてのオプションを処理します。上のリストにないオプションはサポートされておらず、無視されます。

NetX/NetX Duo Telnet サーバーモジュールの動作に関する重要な注意事項と制限事項

- 接続コールバックの場合、アプリケーションは作成したパケットプールまたは IP のデフォルトパケットプールを使用できます。nx_telnet_server_packet_send が失敗した場合、コールバックはパケットを解放する必要があります。
- アクティブなクライアント接続の数は、nx_telnet_server_get_open_connection_count API をコールすることでいつでも取得できます。
- Telnet サーバースレッドタスクは、各クライアント接続の非アクティブタイムアウトまでの残り時間を定期的にチェックします。タイムアウトが期限切れになると、クライアント接続は切断されます。非アクティブタイムアウトの長さを設定するには、Telnet サーバーの [Client inactivity timeout] プロパティを目的の値に設定します。Telnet サーバースレッドタスクが非アクティブタイムアウトをチェックするインターバルは、[Timeout check period] プロパティで設定します。これは、クライアントの非アクティブタイムアウトよりも短くなければなりません。
- Telnet サーバーは nx_telnet_server_stop API を使用して停止でき、nx_telnet_server_start API を使用して再起動できます。Telnet サーバーが停止されると、すべてのクライアント接続が切断され、サーバーは Telnet ポートでの受信待ちを停止します。
- Telnet サーバーの削除は、Telnet サーバーの停止に似ていますが、加えて Telnet サーバーで使用されるすべてのリソース（タイマ、スレッド、TCP ソケット、および Telnet サーバーが作成した場合は Telnet パケットプール）が解放されます。
- 値 255 のバイトで示される Telnet クライアントコマンドに対する解釈とレスポンスは、アプリケーションが処理します。
- NetX Telnet サーバーは、一定の Telnet オプションコマンドのセットのみサポートします。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.29.4 アプリケーションへの NetX/NetX Duo Telnet サーバーモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo Telnet サーバーモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo Telnet サーバーモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo Telnet サーバーモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_telnet_server0 NetX Telnet Server	Threads	New Stack> X-Ware> NetX> Protocols> NetX Telnet Server
g_telnet_server0 NetX Duo Telnet Server	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo Telnet Server

次の図に示すように、NetX/NetX Duo Telnet サーバーモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

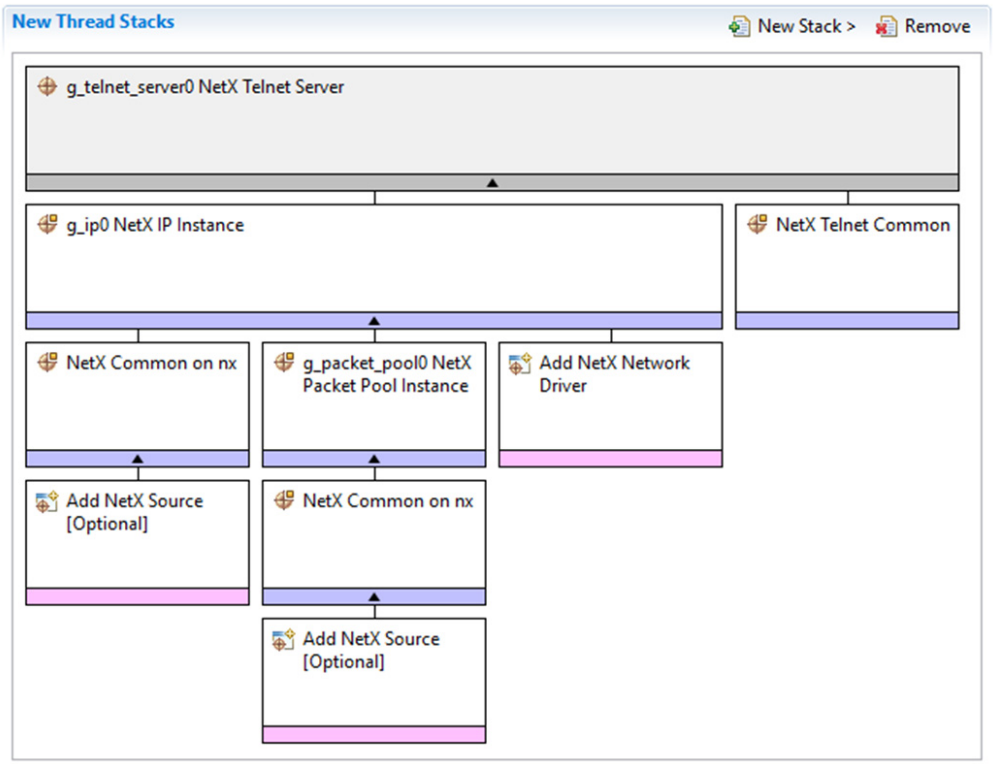


図 385:NetX/NetX Duo Telnet サーバーモジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.29.5 NetX および NetX Duo Telnet サーバーモジュールの構成

ユーザーは必要な動作に合わせて NetX および NetX Duo Telnet サーバーモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注：次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX および NetX Duo Telnet サーバーモジュールの構成設定

ISDE Property	Value	Description
Internal thread priority	16	Internal thread priority selection
Maximum clients to serve simultaneously	4	Maximum clients to serve simultaneously selection
Socket window size (bytes)	2048	Socket window size (bytes) selection
Server time out (seconds)	10	Server time out (seconds) selection
Client inactivity timeout (seconds)	600	Client inactivity timeout (seconds) selection
Timeout check period (seconds)	60	Timeout check period (seconds) selection

ISDE Property	Value	Description
Option negotiation	Enable, Disable Default: Enable	Option negotiation selection
Use application packet pool	Enable, Disable Default: Disable	Use application packet pool selection
Packet size in the pool (bytes)	300	Packet size in the pool (bytes) selection
Total packet pool size (bytes)	2048	Total packet pool size (bytes) selection
Name	g_telnet_server0	Name selection
Internal thread stack size (bytes)	2048	Internal thread stack size (bytes) selection
Name of Client Connect Callback Function	telnet_client_connect	Name of Client Connect Callback Function selection
Name of Receive Data Callback Function	telnet_receive_data	Name of Receive Data Callback Function selection
Name of Client Disconnect Callback Function	telnet_client_disconnect	Name of Client Disconnect Callback Function selection
Name of generated initialization function	telnet_server_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、異なる IP アドレスおよびサブネットマスクの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX および NetX Duo Telnet サーバーローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
Default Gateway Address (use commas for separation)	0,0,0,0	Default gateway address selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection

ISDE Property	Value	Description
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function
Link status change callback	NULL	Link status change callback selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo Telnet 共通インスタンスの構成設定

ISDE Property	Value	Description
Type of Service for UDP requests	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost Default: Normal	Type of service UDP requests selection
Fragmentation option	Don't fragment, Fragment okay Default: Don't fragment	Fragment option selection
Server TCP port number	23	Server TCP port number selection
Time to live	128	Time to live selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX および NetX Duo Telnet サーバーモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX および NetX Duo Telnet サーバーモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin Selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.29.6 アプリケーションでの NetX および NetX Duo Telnet サーバーモジュールの使用

一般的なアプリケーションでNetXおよびNetX Duo Telnetサーバーモジュールを使用する際の手順は次のとおりです。

1) nx_ip_status_check API を使用して、IP インスタンスが通信可能であることを確認します。
2) オプションネゴシエーションを行うように Telnet サーバーを構成します（オプション）。
3) オプションネゴシエーションが有効になっている場合、Telnet サーバーを構成して独自のパケットプールを作成する（デフォルト）か、アプリケーションから Telnet サーバープールを設定します。
4) nx_telnet_server_start API を使用して Telnet サーバーを起動します。
5) ビルド時に指定されたコールバックでクライアント要求を処理します。

6) 完了したら、Telnet サーバーを削除します。
7) アプリケーションによって作成された Telnet サーバーパケットプールは、他のスレッドで使用されていない場合、削除して構いません。

NetX Telnet サーバーモジュールアプリケーションは、登録されたコールバック関数内でレスポンス処理を実行する必要があります。通常、コールバック処理では、受信パケットの内容を確認し、nx_telnet_server_packet_send API でデータの内容を送信します。サーバー側から通信の切断を要求された場合は、nx_telnet_server_disconnect API をコールします。

次の図は、一般的な手順を示した通常の動作フロー図です。

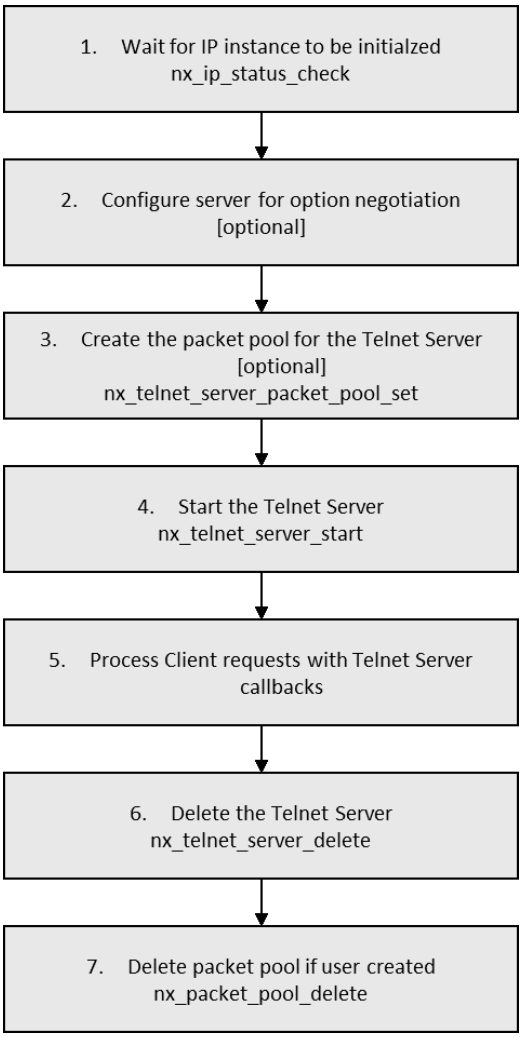


図 386:一般的な NetX および NetX Duo Telnet サーバーモジュールアプリケーションのフロー図

4.3.30 NetX/NetX Duo TFTP クライアント

Trivial File Transfer Protocol（TFTP）は、UDP によるファイル転送のために設計されたプロトコルです。堅牢な TCP ベースのプロトコルとは異なり、TFTP は stop-and-wait プロトコルなので、広範なエラーチェックは行わず、パフォーマンスは制限されていることがあります。TFTP データパケットが送信された後、送信者は受信者から ACK が返ってくるのを待機します。簡単な方法ですが、TFTP の全体的なスループットは制限されません。TFTP サーバーでは、クライアント要求の受信待ちのために、周知された UDP ポート 69 が使用されます。TFTP クライアントは、利用可能な任意の UDP ポートを使用できます。データパケットは、最後のパケットまで、512 バイトです。512 バイト未満のパケットは、ファイルの最後を示します。

注：特に説明がない限り、NetX Duo TFT クライアントはアプリケーション設定や TFTP セッションの実行に関して NetX TFTP クライアントと同じです。NetX Duo で IPv6 用に IP インスタンスを設定する方法については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。

4.3.30.1 NetX/NetX Duo TFTP クライアントモジュールの特長

- NetX TFTP クライアントモジュールは、RFC 1350 および関連する RFC に準拠しています。
- 以下を行うためのハイレベル API
  - TFTP クライアントの作成と削除
  - TFTP 受信者（サーバー）へのファイルのリードとライト
  - TFTP クライアントを実行するネットワークインタフェースの設定

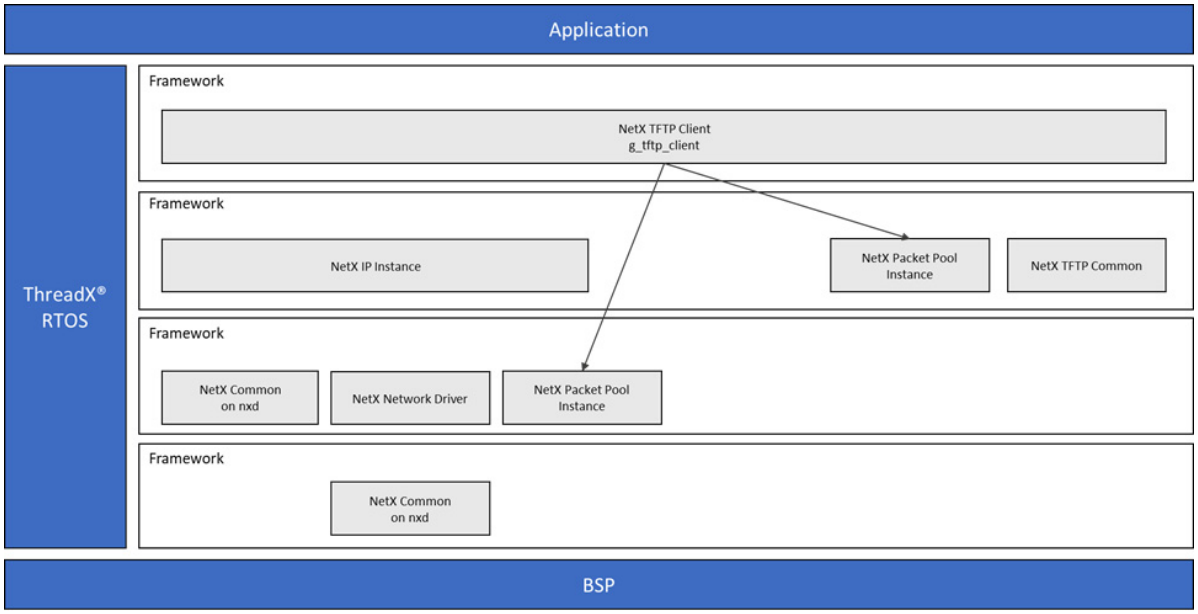


図 387:NetX/NetX Duo TFTP クライアントモジュールのブロック図

注：上の図で、NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.30.2 NetX/NetX Duo TFTP クライアントモジュールの API の概要

NetX/NetX Duo TFTP クライアントモジュールでは、TFTP クライアントの作成と開始のための API が定義されています。内部的には、TFTP クライアントが TFTP サーバーとのすべての通信を処理してファイルを転送します。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX/NetX Duo TFTP クライアントモジュールの API の要約

Function Name	Example API Call and Description
nx_tftp_client_file_open	nx_tftp_client_file_open(&g_tftp_client0, "test.txt", server_ip_address, NX_TFTP_OPEN_FOR_READ, NX_WAIT_FOREVER); Open TFTP client file (IPv4 only).
nxd_tftp_client_file_open**	nxd_tftp_client_file_open(&g_tftp_client0, "test.txt", &server_ip_address_v6, NX_TFTP_OPEN_FOR_READ, NX_WAIT_FOREVER, NX_IP_VERSION_V6); Open TFTP client file. Can be IPv4 type or IPv6
nx_tftp_client_create	nx_tftp_client_create(&g_tftp_client0, "TFTP Client", &g_ip0, &g_packet_pool0); Create a TFTP client instance
nxd_tftp_client_create**	nx_tftp_client_create(&g_tftp_client0, "g_tftp_client0 TFTP Client", &g_ip0, &g_packet_pool0, NX_IP_VERSION_V6); Create a TFTP client instance.
nx_tftp_client_delete	nx_tftp_client_delete(&g_tftp_client0); Delete a TFTP client instance.

Function Name	Example API Call and Description
nxd_tftp_client_delete**	nxd_tftp_client_delete(&g_tftp_client0); Delete a TFTP client instance.
nx_tftp_client_error_info_get	nx_tftp_client_error_info_get(&g_tftp_client0, &error_code, &error_string); Get client error information.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :** この API は、NetX Duo TFTP クライアントでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

ステータス戻り値

Name	Description
NX_SUCCESS	Successful TFTP create
NX_PTR_ERROR*	Invalid IP, pool, or TFTP pointer
NX_CALLER_ERROR *	Invalid caller of this service
NX_TFTP_NOT_CLOSED	Cannot open file; Client already has file open
NX_TFTP_CODE_ERROR	Error status received from TFTP server
NX_INVALID_TFTP_SERVER_ADDRESS	Invalid server address received
NX_TFTP_FAILED	Unknown code received from Server
NX_TFTP_NO_ACK_RECEIVED	No ACK received from server on read or write
NX_TFTP_INVALID_BLOCK_NUMBER	Block of data in TFTP server ACK does not match TFTP Client after read or write request.
NX_TFTP_NOT_OPEN	TFTP client not in the open state for file read or write

Name	Description
NX_IP_ADDRESS_ERROR*	Invalid Server IP address
NX_OPTION_ERROR*	Invalid open type
NX_INVALID_TFTP_SERVER_ADDRESS	Invalid server address received
NX_TFTP_END_OF_FILE	End of file detected (not an error) during file download (read) transfer
NX_TFTP_TIMEOUT	Timeout waiting for Server ACK of write request
NX_TFTP_INVALID_INTERFACE*	Invalid interface input

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

注 : これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX Duo でのエラーチェックサービスの詳細については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

4.3.30.3 NetX/NetX Duo TFTP クライアントモジュールの動作の概要

NetX/NetX Duo TFTP クライアントモジュールでは、クライアントは自動的に作成され、TFTP パケットを送受信するために UDP ソケットが作成されます。TFTP クライアントアプリケーション内では、TFTP クライアントは最初にファイルを「開き」ます。サーバーがアクノリッジを返すと、クライアントは TFTP サーバーとの間でのリード（受信）またはライト（送信）を行うためのファイル転送を要求できるようになります。各ファイル転送が完了したら、TFTP クライアントはファイルを閉じます。

TFTP サーバーは、周知されたポート 69 でクライアント要求を受信待ちします。TFTP クライアントソケットはどのポートにもバインドできます。TFTP でファイルを転送するときにパケットに追加されるデータの量は、未送信のデータが 512 バイト未満になるまでは、512 バイトのみです。512 バイト未満のパケットは、ファイルの最後を示します。イベントの一般的なシーケンスは次のとおりです。

TFTP リードファイル要求

1) クライアントはファイル名を含む「リード用オープン」要求を発行し、サーバーのレスポンスを待機します。
2) サーバーはファイルの最初の 512 バイトを「ブロック番号 1」として送信します。
3) クライアントは、データを受信すると、ブロック番号に「1」を指定した ACK を送信し、次のパケットを待機します。
4) サーバーはブロック番号を毎回インクリメントしながら残りのデータを送信し、クライアント ACK は対応するブロック番号の ACK を毎回送信します。
5) このシーケンスは、最後の 512 バイト未満のパケットを受信すると終了します。nx_tftp_client_file_read は、ファイル転送の最後のパケットを受信すると NX_TFTP_END_OF_FILE を返します。

TFTP ライト要求

1) クライアントはファイル名を含む「ライト用オープン」要求を発行し、サーバーからの ACK（ブロック番号 0）を待機します。
2) サーバーは ACK（ブロック番号 0）を送信します。

3) クライアントはファイルの最初の 512 バイトをサーバーに送信して ACK を待機します。
4) このバイトが書き込まれたら、サーバーは ACK を送信します。
5) クライアントが 512 バイト未満のパケットの送信を完了すると、このシーケンスは完了します。

NetX/NetX Duo TFTP クライアントモジュールの動作に関する重要な注意事項と制限事項

- nx_tftp_client_file_read が NX_SUCCESS, を返した場合、コール側は処理の完了後にパケットを開放する必要があります。
- nx_tftp_client_file_write が NX_TFTP_NOT_OPEN, を返した場合、コール側はパケットを開放する必要があります。それ以外の場合は、NetX または NetX TFTP クライアントモジュールがパケットを開放します。
- ファイルデータにはどのようなデータも使用できます。NetX TFTP クライアントモジュールは、送受信されるデータを一切変更しません。
- NetX TFTP クライアントモジュールは、RFC 1350 および関連する RFC に準拠しています。
- テキストファイルの改行コードは変更できません。ユーザーアプリケーションで処理する必要があります。
- NetX TFTP クライアントモジュールサービスは、複数のスレッドから同時にコールすることができます。ただし、特定のクライアントインスタンスに対するリードまたはライト要求は、同じスレッドから順番に行う必要があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.30.4 アプリケーションへの NetX/NetX Duo TFTP クライアントモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo TFTP クライアントモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo TFTP クライアントモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX/NetX Duo TFTP クライアントモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_tftp0 NetX TFTP Client	Threads	New Stack> X-Ware> NetX> Protocols> NetX TFTP Client
g_tftp0 NetX Duo TFTP Client	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo TFTP Client

次の図に示すように、NetX/NetX Duo TFTP クライアントモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

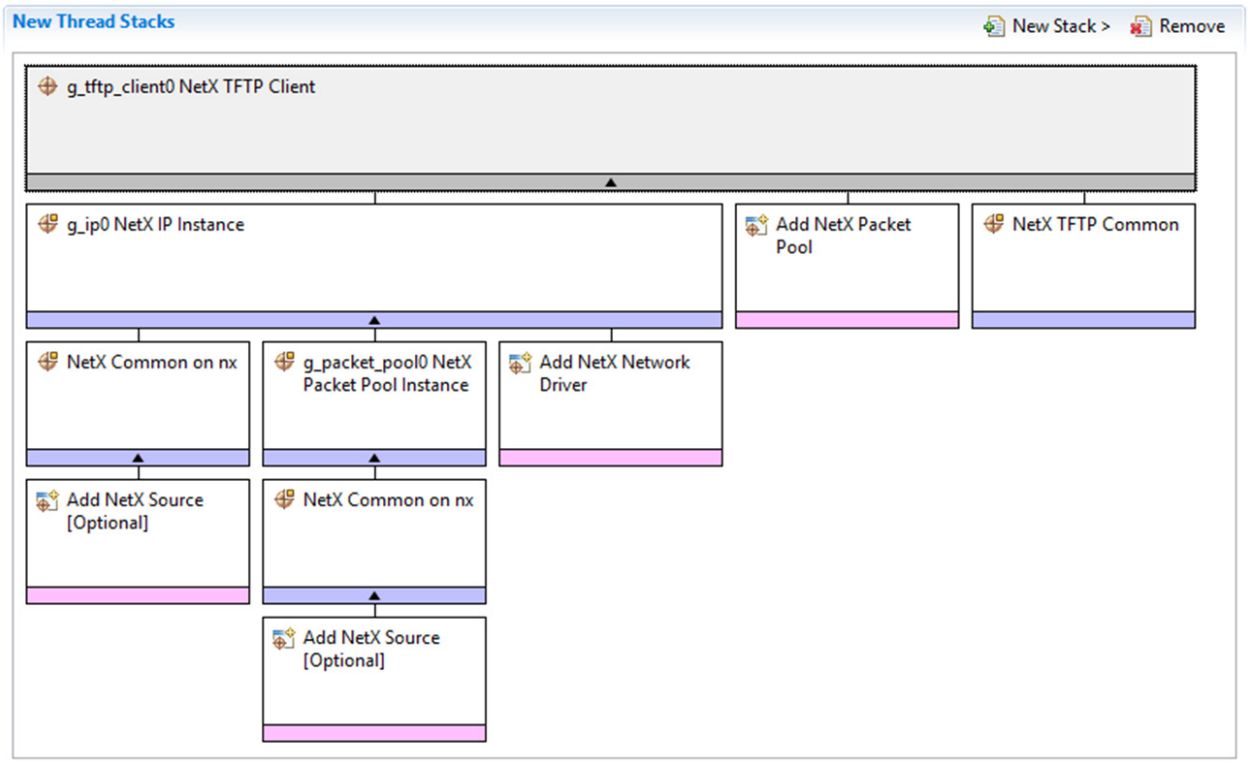


図 388:NetX/NetX Duo TFTP クライアントモジュールのスタック

上記のスタックで、NetX ネットワークドライバー（NetX Duo スタックの場合は NetX Duo ネットワークドライバー）はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.30.5 NetX/NetX Duo TFTP クライアントモジュールの構成

ユーザーは必要な動作に合わせて NetX/NetX Duo TFTP クライアントモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX/NetX Duo TFTP クライアントモジュールの構成設定

ISDE Property	Value	Description
Source port to use	NX_ANY_PORT	Source port to use selection
Name	g_tftp_client0	Module name
Name of generated initialization function	tftp_client_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、イーサネットペリフェラル用に異なるピンの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX/NetX Duo TFTP クライアントのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
Default Gateway Address (use commas for separation)	0,0,0,0	Default gateway address selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection

ISDE Property	Value	Description
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function
Link status change callback	NULL	Link status change callback selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX/NetX Duo TFTP 共通インスタンスの構成設定

ISDE Property	Value	Description
Maximum error string length (bytes)	64	Maximum error string length selection
Time to live	128	Time to live selection

ISDE Property	Value	Description
Type of Service for UDP requests	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost Default: Normal	Type of service UDP requests selection
Fragmentation option	Don't fragment, Fragment okay Default: Don't fragment	Fragment option selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection

ISDE Property	Value	Description
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo TFTP クライアントモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX/NetX Duo TFTP クライアントモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin Selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1

Property	Value	Description
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.30.6 アプリケーションでの NetX/NetX Duo TFTP クライアントモジュールの使用

次の例は、実際に使用できる有効な IP、ARP、UDP が確立済みで、リンクが実行されているシステムを前提にしています。

一般的なアプリケーションで NetX/NetX Duo TFTP クライアントモジュールを使用する際の手順は次のとおりです。

1) IP インスタンスの IP アドレスが有効な場合は、nx_ip_status_check API をポーリングします。
2) ファイルライトを要求するには、open_type 入力として NX_TFTP_OPEN_FOR_WRITE を指定して nx_tftp_client_file_open API をコールします。
3) nx_tftp_client_packet_allocate API を使用してパケットを割り当て、パケットバッファにファイルデータをライトします。これで、このパケットの送信準備が整います。
4) nx_tftp_client_file_write API をコールしてサーバーに送信します。
5) すべてのファイルデータが送信されるまで繰り返します。この API が NX_TFTP_NOT_OPEN, を返さない限り、アプリケーションではパケットを開放しないでください。
6) nx_tftp_client_file_close. をコールしてファイルを閉じます。

7) ファイルリードを要求するには、open_type 入力として NX_TFTP_OPEN_FOR_READ を指定して nx_tftp_client_file_open API をコールします。
8) nx_tftp_client_file_read API をコールしてファイルデータを受信します。
9) 512 バイト未満のパケットを受信するまで継続します。処理が完了したら、パケットをパケットプールに開放します。
10) nx_tftp_client_file_close API を使用してファイルを閉じます。
11) TFTP クライアントでの処理が完了したら、nx_tftp_client_delete API を使用してインスタンスを削除します。

次の図は、一般的な手順を示した通常の動作フロー図です。

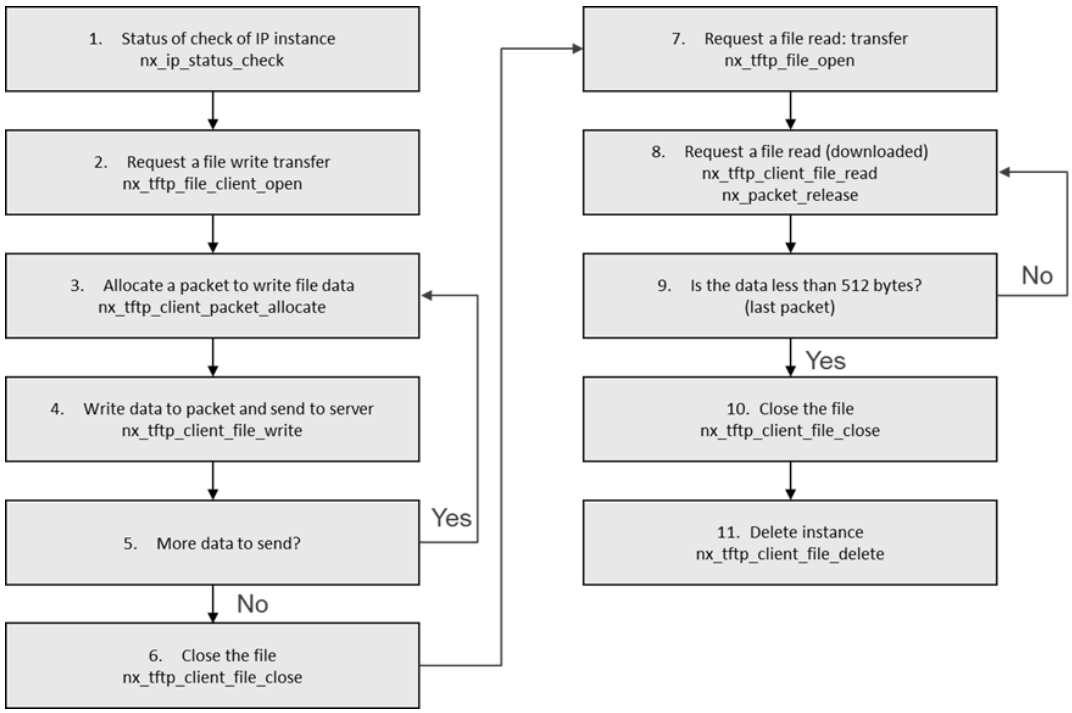


図 389:一般的な NetX/NetX Duo TFTP クライアントモジュールアプリケーションのフロー図

4.3.31 NetX/NetX Duo TFTP サーバー

NetX ™の Trivial File Transfer Protocol（TFTP）は、ファイル転送のために設計されたプロトコルです。堅牢なプロトコルとは異なり、TFTP は stop and-wait プロトコルなので、広範なエラーチェックは行わず、パフォーマンスは制限されていることがあります。TFTP データパケットが送信された後、送信者は受信者から ACK が返ってくるのを待機します。簡単な処理方法ですが、TFTP の全体的なスループットは制限されません。TFTP パッケージによって、ホストは IP ネットワークで TFTP プロトコルを使用できるようになります。

注 :NetX Duo ™ TFTP サーバーは、内部処理を除けば、TFTP セッションの応用、設定、実行が NetX TFTP と同じです。

4.3.31.1 NetX および NetX Duo TFTP サーバーモジュールの特長

- NetX サーバーの TFTP モジュールは、RFC 1350 および関連する RFC に準拠しています。
- また、以下を行うためのハイレベル API を提供します。
  - TFTP サーバーの作成と削除
  - TFTP サーバータスクスレッドの開始と停止

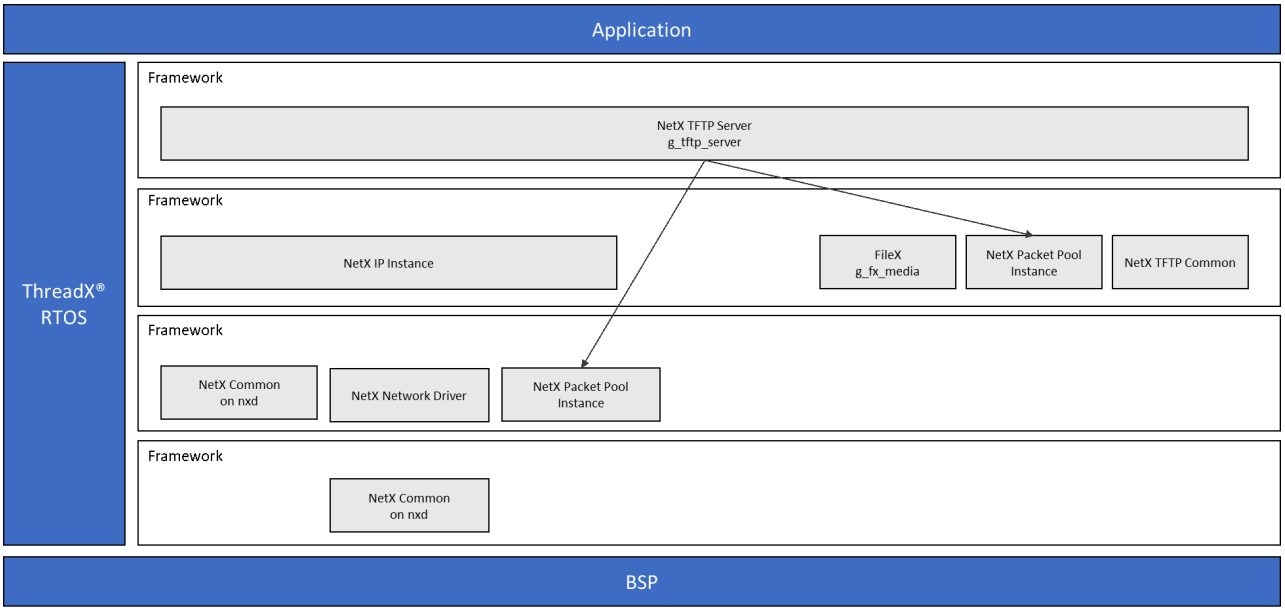


図 390:NetX/NetX Duo TFTP サーバーモジュールのブロック図

注 : 上の図で、FileX および NetX（または NetX Duo）ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.31.2 NetX および NetX Duo TFTP サーバーモジュールの API の概要

NetX TFTP サーバーモジュールでは、レスポンスパケットの作成、削除、生成、レスポンス送信、受信パケットからの情報の取得のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX および NetX Duo TFTP サーバーモジュールの API の要約

Function Name	Example API Call and Description
nx_tftp_server_create	nx_tftp_server_create(&my_server, "My TFTP Server", server_ip, &ram_disk, stack_ptr, 2048, pool_ptr); Create TFTP server (IPv4 only)
nxd_tftp_server_create**	nxd_tftp_server_create(&my_server, "My TFTP Server", &server_ip, &ram_disk, stack_ptr, 2048, pool_ptr); Create TFTP server (IPv4 and IPv6 supported).
nx_tftp_server_delete	nx_tftp_server_delete(&my_server); Delete TFTP server.
nxd_tftp_server_delete**	nxd_tftp_server_delete(&my_server); Delete TFTP server.
nx_tftp_server_start	nx_tftp_server_start(&my_server); Start the TFTP server.
nxd_tftp_server_start**	nxd_tftp_server_start(&my_server); Start the TFTP server.
nx_tftp_server_stop	nx_tftp_server_stop(&my_server); Stop the TFTP server.
nxd_tftp_server_stop**	nxd_tftp_server_stop(&my_server); Stop the TFTP server.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :** この API は、NetX Duo TFTP サーバーでのみ使用できます。NetX Duo 固有のデータ型の定義については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

ステータス戻り値

Name	Description
NX_SUCCESS	API Call Successful
NX_TFTP_POOL_ERROR*	Packet pool payload is too small for the 512 bytes of TFTP data.
NX_PTR_ERROR*	Invalid pointer input
NX_CALLER_ERROR*	Invalid caller of this service

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

注 : これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。NetX Duo でのエラーチェックサービスの詳細については、『NetX Duo User Guide for the Renesas Synergy ™ Platform』を参照してください。*

4.3.31.3 NetX および NetX Duo TFTP サーバーモジュールの動作の概要

NetX Duo TFTP パッケージの TFTP クライアントの部分を適切に機能させるためには、IP インスタンスが既に作成されている必要があります。

注 :TFTP サーバーインスタンスがプロジェクトに追加されると、ARP と TCP が有効化された TFTP サーバーの IP インスタンスが自動的に作成されます。

同じ IP インスタンスで UDP が有効になっている必要があります。NetX Duo TFTP パッケージのクライアントの部分には、追加要件はありません。

NetX TFTP パッケージの TFTP サーバーの部分では、すべてのクライアント TFTP 要求を処理するために、UDP ポート 69 への完全なアクセスが必要です。また、TFTP サーバーは、FileX® 組み込みファイルシステムと連携するように設計されています。FileX を使用できない場合、ユーザーは使用される FileX の一部を独自の環境に移行できます（詳細については、この後のモジュールガイドのセクションで解説します）。

ファイル名は、ターゲットファイルシステムのフォーマットに従う必要があります。ファイル名は末尾が NULL の ASCII 文字列である必要があり、必要に応じてフルパス情報を含みます。NetX サーバーの TFTP の実装では、TFTP ファイル名のサイズに制限は指定されていません。

TFTP メッセージ

TFTP では、ファイルのオープン、リード、ライト、クローズを行う際、UDP ヘッダーの下に 2 ～ 4 バイトの TFTP ヘッダーを追加するという単純なメカニズムを採用しています。

TFTP のファイルオープンメッセージの定義のフォーマットは以下のとおりです。

オペコード（2 バイト）	ファイル名（可変長）	NULL（1 バイト）	モード（可変長）	NULL（1 バイト）

オペコード（2 バイト）：

0x0001：Open for read

0x0002：Open for write

ファイル名（可変長）：netascii コード

NULL（1 バイト）：0x00

モード：バイナリ転送を指定する ascii コード "OCTET"

TFTP データ転送フォーマット、ACK、エラーメッセージの定義はやや異なります。

オペコード（2 バイト）	ブロック番号（2 バイト）	データ（512 バイト以下）

オペコード（2 バイト）：

0x0003：Data packet

0x0004：Ack for last read

0x0005：Error condition

ブロック番号：2 バイト

データ：512 バイト以下のデータ

TFTP 通信

アップロードまたはダウンロード対象のファイルを含むデータパケットのペイロードは、最後のパケットが 512 バイト未満になり、その 512 バイト未満のパケットによってファイルの最後が示されるまでは、512 バイトのチャンクとして送信されます。イベントの一般的なシーケンスは次のとおりです。

TFTP リードファイル要求

1) クライアントはファイル名を含むリード用オープン要求を発行し、サーバーからの応答を待機します。
2) サーバーはファイルの最初の 512 バイトを送信します（512 バイト未満のファイルの場合はそれよりも少なくなります）。
3) クライアントはデータを受信し、ACK を送信します。512 バイトを超えるファイルの場合は、サーバーからの次のパケットを待機します。
4) クライアントが 512 バイト未満のパケットを受信すると、このシーケンスは完了します。

TFTP ライト要求

1) クライアントはファイル名を含むライト用オープン要求を発行し、サーバーからの ACK（ブロック番号 0）を待機します。
2) サーバーは、ファイルを書き込む準備ができたら、ACK（ブロック番号 0）を送信します。
3) クライアントはファイルの最初の 512 バイトをサーバーに送信し（512 バイト未満のファイルの場合はそれよりも少なくなります）、ACK が返ってくるのを待機します。
4) このバイトが書き込まれたら、サーバーは ACK を送信します。
5) クライアントが 512 バイト未満のパケットのライトを完了すると、このシーケンスは完了します。

NetX および NetX Duo TFTP サーバーモジュールの動作に関する重要な注意事項と制限事項

NetX TFTP サーバーモジュールでは、FileX メディア（ブロックメディアまたは USB 大容量記憶装置）が必要です。TFTP サーバースタック要素がプロジェクトに追加されると、**[Add FileX]** ボックスがアタッチされます。サーバーが起動する前に、コンフィギュレータがサーバーに合わせて FileX メディアを自動的にセットアップして初期化します。FileX の構成の詳細については、『*FileX ™ User's Guide* for the Renesas Synergy ™ Platform』を参照してください。

NetX TFTP サーバーには、パケットを送信するためにパケットプールも必要です。IP のデフォルトパケットプールを共有することも、個別のパケットプールを作成することもできます（TFTP サーバーのパケットプールの設定の詳細については、セクション 4 を参照してください）。

- クライアント要求のサポートプロパティの再送が有効ではない場合、クライアントがレスポンスを停止した場合でも、TFTP サーバーは TFTP クライアントセッションを維持します。この場合、TFTP サーバーがドロップされたクライアント接続で一杯になり、新しいクライアント要求を受け入れられなくなる可能性があります。
- クライアント要求のサポートプロパティの再送が有効ではない場合、TFTP サーバーは重複したクライアントパケットに応答できません。重複パケットは単純にドロップされ、TFTP サーバーは ACK もデータパケットも送信せず、その結果、クライアントとサーバーがデッドロックされてしまいます。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.31.4 アプリケーションへの NetX および NetX Duo TFTP サーバーモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX と NetX Duo TFTP サーバーモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX/NetX Duo TFTP サーバーモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX および NetX Duo TFTP サーバーモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_tftp0 NetX TFTP Server	Threads	New Stack> X-Ware> NetX> Protocols> NetX TFTP Server
g_tftp0 NetX Duo TFTP Server	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo TFTP Server

次の図に示すように、NetX/NetX Duo TFTP サーバーモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

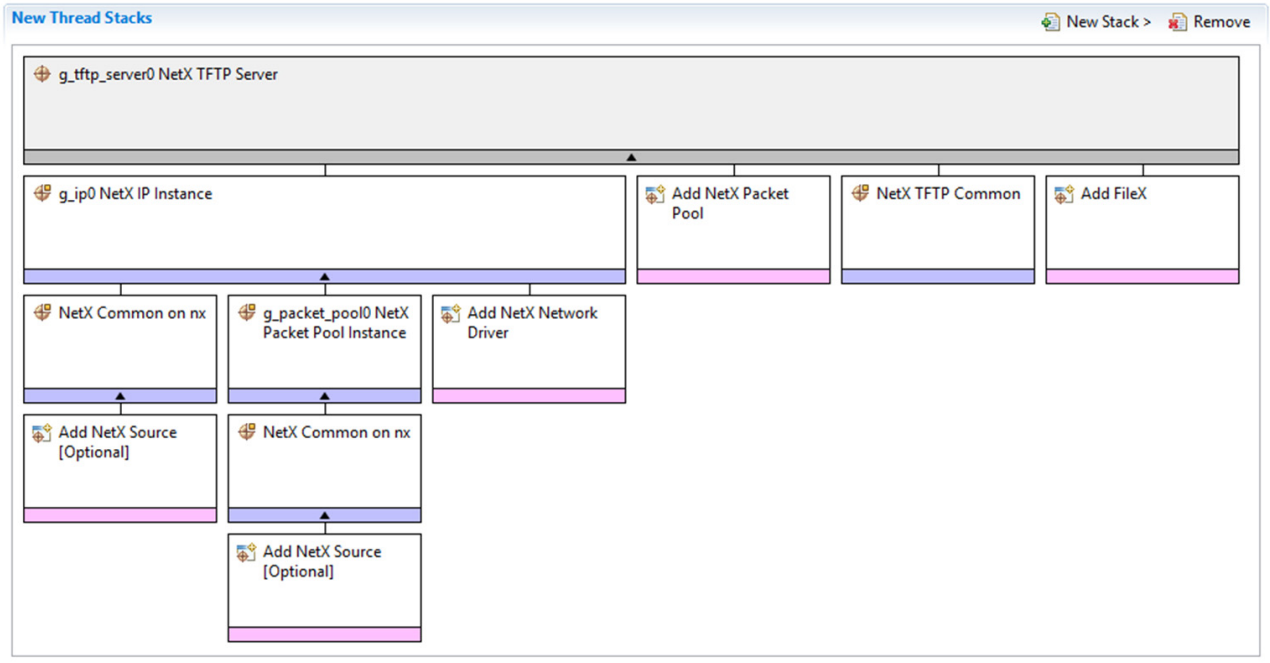


図 391:NetX および NetX Duo TFTP サーバーモジュールのスタック

上記のスタックで、NetX ネットワークドライバー (NetX Duo スタックの場合は NetX Duo ネットワークドライバー) はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上

に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

また、上記のスタックで FileX スタックはまだ設定されていません。FileX モジュールには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- FileX スタブ
- ブロックメディア上の FileX（sf_block_media_ram) 上のブロックメディアフレームワークに実装）
- USB 大容量記憶装置上の FileX（USBX ホストクラス大容量記憶装置上に実装）

4.3.31.5 NetX および NetX Duo TFTP サーバーモジュールの構成

ユーザーは必要な動作に合わせて NetX および NetX Duo TFTP サーバーモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX および NetX Duo TFTP サーバーモジュールの構成設定

ISDE Property	Value	Description
FileX Support	Enable, Disable Default: Enable	FileX support selection
Retransmission on client request support	Enable, Disable Default: Disable	Retransmission on client request support selection
Internal thread priority	16	Internal thread priority selection

ISDE Property	Value	Description
Maximum clients to serve simultaneously	10	Maximum clients to serve simultaneously selection
Time slice for internal thread	2	Time slice for internal thread selection
Client request activity timeout check interval (ticks)	20	Client request activity timeout (ticks) selection
Ack or data retransmission interval (ticks)	200	Ack or data retransmission interval (ticks) selection
Maximum retries for transmission without response	5	Maximum retries for transmission without response selection
Maximum retries for transmission with duplicate response	2	Maximum retries for transmission with duplicate response selection
Name	g_tftp_server0	Module name
Internal thread stack size (bytes)	2048	Internal thread stack size (bytes) selection
Name of generated initialization function	tftp_server_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ローレベルモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、異なる MAC アドレスの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX および NetX Duo TFTP サーバーのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX/NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
Default Gateway Address (use commas for separation)	0,0,0,0	Default gateway address selection
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection

ISDE Property	Value	Description
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization function
Link status change callback	NULL	Link status change callback selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

*注 :** は、NetX Duo でのみ使用できるプロパティを表します。*

NetX および NetX Duo TFTP 共通インスタンスの構成設定

ISDE Property	Value	Description
Maximum error string length (bytes)	64	Maximum error string length selection
Time to live	128	Time to live selection

ISDE Property	Value	Description
Type of Service for UDP requests	Normal, Minimum delay, Maximum data, Maximum reliabililty, Minimum cost Default: Normal	Type of service UDP requests selection
Fragmentation option	Don't fragment, Fragment okay Default: Don't fragment	Fragment option selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX/NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection

ISDE Property	Value	Description
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX および NetX Duo TFTP サーバーモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [Clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX および NetX Duo TFTP サーバーモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin Selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1

Property	Value	Description
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.31.6 アプリケーションでの NetX および NetX Duo TFTP サーバーモジュールの使用

一般的なアプリケーションでNetXおよびNetX Duo TFTPサーバーモジュールを使用する際の手順は次のとおりです。

1) nx_tftp_server_create API を使用して TFTP サーバーを作成します。
2) ブロックメディアまたは USB 大容量記憶装置上の FileX API を準備します。
3) nx_tftp_server_start API を使用して TFTP サーバーを起動します。
4) TFTP サーバーがアクティブなクライアント接続での非アクティブ状態を定期的にチェックするようになります。
5) すべての受信パケットで、サーバーが既に受信したパケットとの重複がチェックされます。
6) クライアントのリード用オープン要求を受信します（内部動作）。
7) サーバーが別のクライアント要求に対応できるかチェックします（[Maximum clients to server simultaneously] プロパティで設定）（内部動作）。
8) 512 チャンクのファイルデータのパケットを最後のパケットまでダウンロードします（内部動作）。
9) ファイルを閉じてクライアント要求を削除します（内部動作）。

次の図は、一般的な手順を示した通常の動作フロー図です。

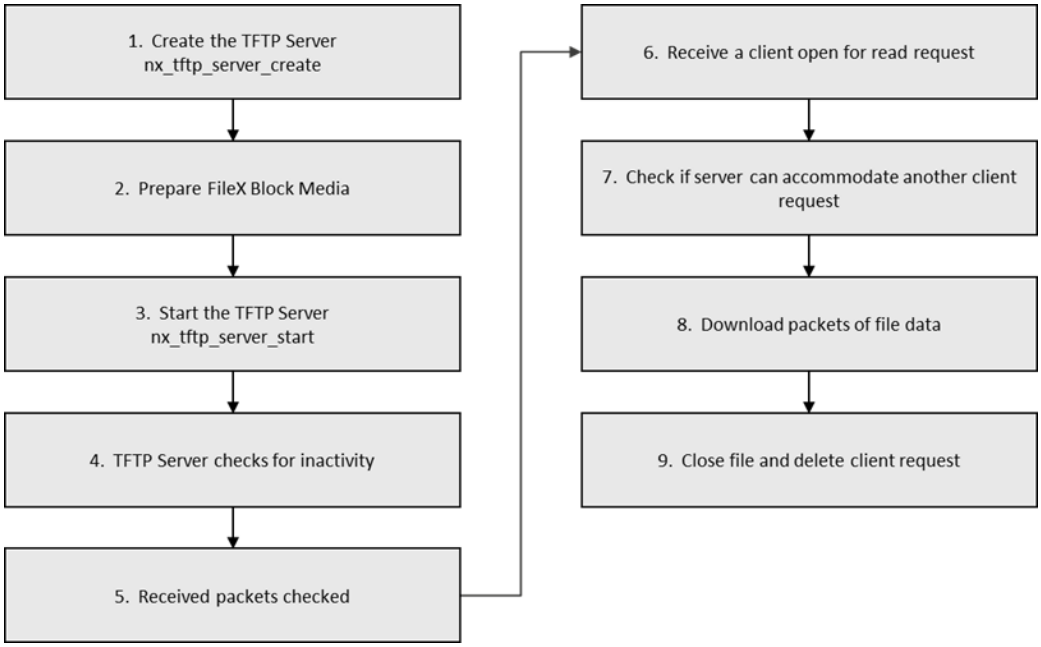


図 392:一般的な NetX および NetX Duo TFTP サーバーモジュールアプリケーションのフロー図

4.3.32 NetX Duo MQTT クライアント

MQTT（Message Queue Telemetry Transport）は、パブリッシャ / サブスクライバモデルに基づいた通信プロトコルです。データの作成元は、ブローカを介して他のクライアントに情報をパブリッシュできます。複数のデータコンシューマが、トピックに関心がある場合、ブローカを介してトピックをサブスクライブできます。ブローカは、クライアントの認証と承認を行い、パブリッシュされたメッセージを、トピックのサブスクライバに提供する必要があります。このパブリッシャ / サブスクライバモデルでは、複数のクライアントが同じトピックでデータをパブリッシュできます。クライアントは、同じトピックをサブスクライブしている場合、パブリッシュされたメッセージを受け取ります。

4.3.32.1 NetX Duo MQTT クライアントモジュールの特長

- OASIS MQTT に準拠しています。
  2014 年 10 月 29 日のバージョン 3.1.1 の仕様は次の場所で確認できます。http://mqtt.org/
- SSP で NetX Secure を使用する安全な通信のために TLS を有効または無効にするオプションを提供します。
- QoS をサポートし、メッセージをパブリッシュするときにレベルを選択できます
- 内部バッファを備え、受信したメッセージのキューを保持します
- 新しいメッセージを受信したときのコールバックを登録するメカニズムを提供します。
- ブローカとの接続が終了されたときのコールバックを登録するメカニズムを提供します。

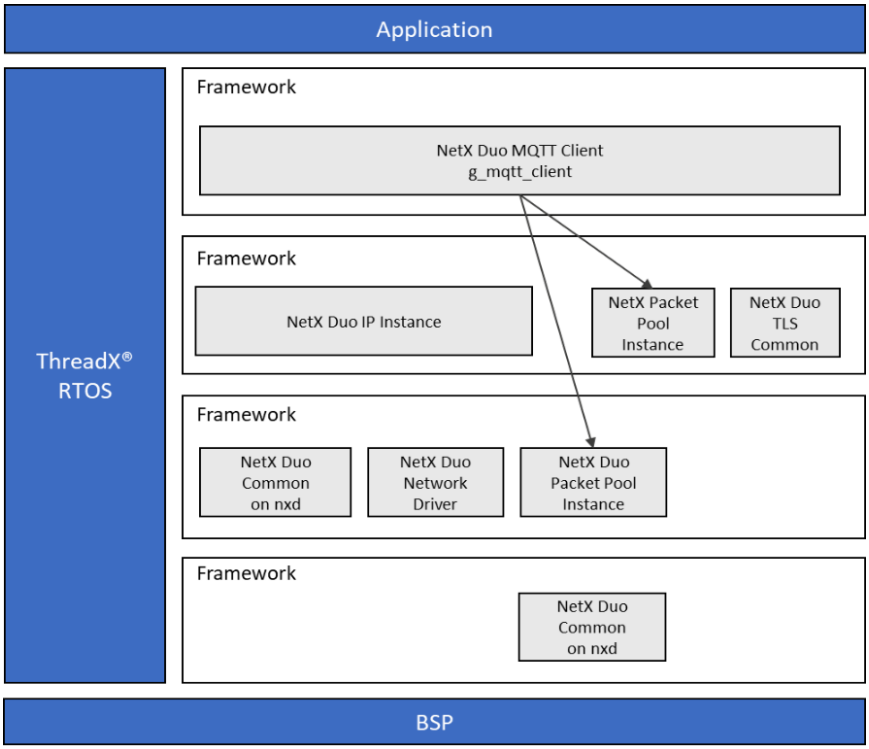


図 393:NetX Duo MQTT クライアントモジュールのブロック図

注 : 上の図で、NetX Duo ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.32.2 NetX Duo MQTT クライアントモジュールの API の概要

NetX Duo MQTT クライアントサポートモジュールでは、MQTT クライアントの作成、ブローカへの接続、TLS セキュリティの設定、MQTT メッセージの受信のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX Duo MQTT クライアントモジュールの API の要約

Function Name	Example API Call and Description
nxd_mqtt_client_create	nxd_mqtt_client_create(&mqtt_client_secure, "my_client", CLIENT_ID_STRING, strlen(CLIENT_ID_STRING), ip_ptr, pool_ptr, (VOID*)mqtt_client_stack, sizeof(mqtt_client_stack), mqtt_thread_priority, (UCHAR*)client_memory, sizeof(client_memory)); Create an MQTT client with the specified Client ID, stack memory and stack size, and message block memory.
nxd_mqtt_client_connect	nxd_mqtt_client_connect(&mqtt_client, &server_ip, NXD_MQTT_PORT, MQTT_KEEP_ALIVE_TIMER, 0, NX_WAIT_FOREVER) Non-secure connect to the MQTT broker specifying broker IP address and port, keep alive timer, and disabling the clean session option
nxd_mqtt_client_secure_connect	nxd_mqtt_client_secure_connect(&mqtt_client_secure, &server_ip, NXD_MQTT_TLS_PORT, tls_setup_amazon, 600, 1, NX_WAIT_FOREVER) Connect to broker with TLS security using the tls_setup_amazon, which is a user-defined function, to set up TLS and set TLS parameters. The clean session option is enabled. This is only available if the NetX Duo library is built with NX_SECURE_ENABLE set, and if the MQTT client property NX Secure is set.

Function Name	Example API Call and Description
nxd_mqtt_client_login_set	nxd_mqtt_client_login_set(mqtt_client_ptr, "Username", strlen("Username"), "Password", strlen("Password"); Set the optional MQTT username and password. This must be called before the nxd_mqtt_client_connect or nxd_mqtt_client_secure_connect call if the broker requires username and password.
nxd_mqtt_client_message_get	nxd_mqtt_client_message_get(&mqtt_client_secure, &topic, &topic_length, &message, &message_length, &packet_ptr); Retreive a published MQTT message for the specified topic.
nxd_mqtt_client_receive_notify_set	nxd_mqtt_client_receive_notify_set(&mqtt_client_secure, my_notify_func); Specify the function the MQTT Client thread task calls when an MQTT message is received.
nxd_mqtt_client_subscribe	nxd_mqtt_client_subscribe(&mqtt_client_secure, TEST_SUBSCRIBE_TOPIC_NAME, strlen(TEST_SUBSCRIBE_TOPIC_NAME), 0); Send a subscriber message to the broker for the specified topic for QoS (quality of service) level 0.
nxd_mqtt_client_unsubscribe	nxd_mqtt_client_unsubscribe(NXD_MQTT_CLIENT *mqtt_client_pr, CHAR *topic_name, UINT topic_name_length); Send an unsubscriber message to the broker for the specified topic.

Function Name	Example API Call and Description
nxd_mqtt_client_publish	nxd_mqtt_client_publish(&mqtt_client_secure, TEST_SUBSCRIBE_TOPIC_NAME, strlen(TEST_SUBSCRIBE_TOPIC_NAME), message_buffer, strlen(message_buffer), 0, 1, NX_WAIT_FOREVER); Send a message to the broker for the specified topic previously subscribed to for QoS (quality of service) level 1, and the retain message option disabled.
nxd_mqtt_client_disconnect	nxd_mqtt_client_disconnect(&mqtt_client); Disconnect from the MQTT broker.
nxd_mqtt_client_disconnect_notify_set	nxd_mqtt_client_disconnect_notify_set(mqtt_client_ptr, my_disconnect_notify); Specify the user defined function for the MQTT Client thread task to call if the broker initiates disconnecting from the client.
nxd_mqtt_client_delete	nxd_mqtt_client_delete(mqtt_client_ptr); Delete the MQTT instance, clear transmit and message queue messages
nxd_mqtt_client_will_message_set	nxd_mqtt_client_will_message_set(mqtt_client_ptr, will_topic, will_topic_length, “will_message”, strlen(“will_message”), 0, 1); Set the optional MQTT Client will message without the retain will message option, for QOS 1. If a will message is needed, this must be called before connecting to the broker.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

4.3.32.3 NetX Duo MQTT クライアントモジュールの動作の概要

MQTT（Message Queue Telemetry Transport）は、NetX Duo TCP/IP スタックとパブリッシャ / サブスクライバモデルに基づいたプロトコルです。クライアントは、MQTT サーバー（ブローカ）を介して他のクライアントに情報をパ

ブリッシュできます。クライアントは、トピックに関心がある場合、ブローカを介してトピックをサブスクライブできます。ブローカは、パブリッシュされたメッセージを、トピックをサブスクライブしているクライアントに提供する必要があります。このパブリッシャ / サブスクライバモデルでは、複数のクライアントが同じトピックでデータをパブリッシュできます。クライアントは、同じトピックをサブスクライブしている場合、パブリッシュされたメッセージを受け取ります。

次の図は、MQTT クライアントのパブリッシュ / サブスクライブモデルの概要です。

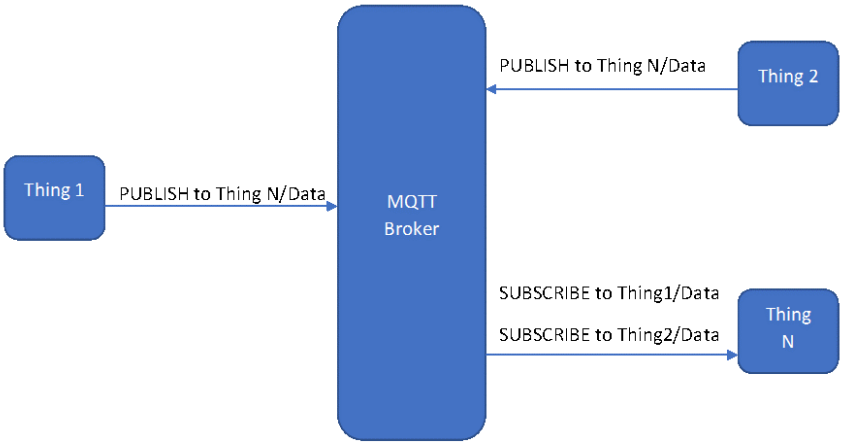


図 394:NetX Duo MQTT クライアントモジュールの MQTT クライアントのパブリッシュ / サブスクライブモデル

NetX Duo MQTT クライアントモジュールのパブリッシュ / サブスクライブモデル

NetX Duo MQTT クライアントモジュールは、ノーマルモードまたはセキュアモードで使用できます。

NetX Duo MQTT クライアントモジュールのノーマルモードの動作の説明

ノーマルモードでは、MQTT クライアントとブローカの間の通信はセキュアではありません。

NetX Duo MQTT クライアントモジュールのセキュアモードの動作の説明

セキュアモードでは、TLS プロトコルによって MQTT クライアントとブローカの間の通信がセキュアになっています。スレッドペインでは、TLS プロトコルは [Add NetX Duo TLS common [Optional]] ブロックによって表されます。

クライアントはメッセージをパブリッシュするときに、ユースケースに応じて、3 つの QoS レベルのうち 1 つを選択できます。

QoS 0：メッセージが最高 1 回送信されます。QoS 0 で送信されたメッセージは、失われることがあります。

QoS 1：メッセージが 1 回以上送信されます。QoS 1 で送信されたメッセージは複数回送信されることがあります。

QoS 2:メッセージが必ず 1 回のみ送信されます。QoS 2 で送信されたメッセージは、重複なしでの送信が保証されます。

注 : この MQTT クライアントの実装では、QoS レベル 2 のメッセージはサポートされません。

QoS 1 と QoS 2 は送信が保証されているため、ブローカは各クライアントに送信された QoS 1 と QoS 2 のメッセージの状態を追跡します。これは、QoS1 または QoS 2 メッセージを期待するクライアントには特に重要なことです。クライアントとブローカの間の接続は解除されることがあります（クライアントのリブート時や、通信リンクが一時的に失われた場合など）。ブローカは、クライアントがブローカと再接続された後にメッセージを送信できるように、QoS 1 と QoS 2 のメッセージを保存する必要があります。

しかし、クライアントでは、再接続の後でもブローカからの古いメッセージを受信しないことを選択できます。そのためには、クライアントは nxd_mqtt_client_connect API で clean_session フラグを NX_TRUE（1）に設定して通信

を開始します。この場合、ブローカは MQTT CONNECT メッセージを受信した時点で、送信または確認がされていない QoS 1 または QoS 2 のメッセージを含む、このクライアントに関連するセッション情報を破棄します。

clean_session フラグが NX_FALSE. の場合、サーバーは QoS 1 と QoS 2 のメッセージを再送信します。clean_session が NX_TRUE. に設定されている場合、MQTT クライアントも、アクノリッジされていないメッセージを再送信します。このアクノリッジは、同様に頻繁に発生する TCP ソケットレイヤーの ACK とは異なります。MQTT クライアントは、メッセージを受信したときは MQTT アクノリッジメッセージをブローカに送信し、メッセージをパブリッシュしたときは戻ってくるアクノリッジメッセージを受け取ります。

受信 MQTT メッセージは MQTT クライアントインスタンスの受信キューに保存されます。アプリケーションは、トピックとトピックメッセージの両方を返す nxd_mqtt_client_message_get API をコールすることで、これらのメッセージを取得します。アプリケーションは、それぞれに十分な量のバッファを確実に提供する必要があります。キュー内の最も古いメッセージが最初にコール側に返されます。nxd_mqtt_client_message_get は非ブロックです。MQTT クライアントの受信キューが空の場合は、即座に NXD_MQTT_NO_MESSAGE（0x1000A）ステータスを返します。これはエラーとしては処理されませんが、受信キューが空だったとして処理されます。

受信キューによる受信メッセージのポーリングを防ぐために、アプリケーションでは nxd_mqtt_client_recieve_notify_set API をコールして、受信メッセージのコールバック関数を MQTT クライアントに登録できます。コールバック関数は、以下のように定義されます。

```
VOID (*receive_notify_callback)(NXD_MQTT_CLIENT *client_ptr, UINT message_count)
;
```

コールバック関数が設定されている場合、MQTT クライアントは、ブローカからメッセージを受信するとコールバック関数を呼び出します。このコールバック関数はクライアント制御ブロックへのポインタとメッセージカウント値を渡します。メッセージカウント値は受信キュー内の MQTT メッセージの数を示します。このコールバック関数は MQTT クライアントスレッドコンテキスト内で実行されることに注意してください。したがって、このコールバック関数では、MQTT クライアントスレッドをブロックする可能性のある手順はいずれも実行しないでください。コールバック関数は、アプリケーションスレッドをトリガして、メッセージを受信するための nxd_mqtt_client_message_get API をコールする必要があります。このことは、モジュールガイドのプロジェクトで示されています。

MQTT クライアントサービスの接続を解除するには、アプリケーションでサービスの nxd_mqtt_client_disconnect および nxd_mqtt_client_delete, API をそれぞれ使用する必要があります。nxd_mqtt_client_disconnect をコールするとブローカへの TCP 接続が解除されます。これにより、既に受信されて受信キューに保存されているメッセージが解放されます。ただし、送信キュー内の QoS レベル 1 のメッセージは開放されません。QoS レベル 1 のメッセージは、clean_session フラグが NX_FALSE. に設定されている場合、接続時に再送信されます。

ブローカはクライアントからの接続の解除を開始できます。MQTT クライアントに切断通知関数を登録すると、アプリケーションに切断要求が通知されます。その方法は、nxd_mqtt_client_disconnect_notify_set API をコールすることです。

MQTT クライアントを削除するには、nxd_mqtt_client_delete API をコールします。これにより、送信キューと受信キュー内のすべてのメッセージブロックが開放されます。アクノリッジされていない QoS レベル 1 のメッセージも削除されます。

セキュアな通信の使用

MQTT クライアントとブローカの間の通信をセキュアにするには、TLS プロトコルを使用する必要があります。スレッドペインでは、TLS プロトコルは [Add NetX Duo TLS common [Optional]] ブロックによって表されます。NetX Duo TLS 共通ブロックを追加すると、TLS のサポートが有効になります。

TLS/NetX Duo Secure を使用した MQTT

MQTT クライアントで TLS を使用する際は、nxd_mqtt_client_secure_connect コール内の TLS セットアップコールバックに、TLS インスタンスの作成、ローカル証明書の定義、リモート証明書の処理のためのメモリの割り当て、タイムスタンプや証明書認証などのオプションのコールバックなど、すべての TLS セットアップを含めることを強くお勧めします。

MQTT クライアントが TCP 経由で正常に接続できたかどうかにかかわらず、また、TLS セッションが正常に開始したかどうかにかかわらず、アプリケーションは再接続を試行する前に nxd_mqtt_client_disconnect をコールして、TLS セッションを適切にクリアおよびリセットする必要があります。

セッションが適切に終了されなかった場合、nxd_mqtt_client_disconnect を同じ理由で引き続きコールする必要があります。

SSP 1.3.x の場合、TLS セットアップコールバックでは、TLS セッションを（再）作成（nxd_secure_tls_session_create）. する前に、NXD_SECURE_TLS_SESSION データブロック上で memset をコールする必要があります。

TLS セットアップ入力での nxd_mqtt_client_secure_connect API の定義は以下のとおりです。

```
UINT nxd_mqtt_client_secure_connect(NXD_MQTT_CLIENT *client_ptr,
                                    NXD_ADDRESS *server_ip,
                                    UINT server_port,
                                    UINT (*tls_setup)(
                                         NXD_MQTT_CLIENT *client_ptr,
                                         NX_SECURE_TLS_SESSION *session_ptr,
                                         NX_SECURE_X509_CERT *,
                                         NX_SECURE_X509_CERT *),
                                   UINT keepalive,
                                   UINT clean_session,
                                   ULONG wait_option)
```

このロジックを tls_setup コールバック関数に追加します（MQTT クライアントインスタンス名が g_mqtt_client0): であると想定しています）。

```
session_ptr = &(g_mqtt_client0.nxd_mqtt_tls_session);
memset(session_ptr, 0, sizeof(NX_SECURE_TLS_SESSION));
status = nxd_secure_tls_session_create(....)
```

memset がコールされていない場合、TLS の nxd_secure_tls_session_create コールは成功しないことがあります。SSP 1.4.0 では memset のコールは必須ではなくなりますが、TLS の作成を含むすべての TLS セットアップをコールバックに含めることが引き続き強く推奨されます。TLS セッションを完全に削除して再作成することは、無駄なことに見えるかもしれません。しかし、TLS が MQTT クライアントに統合されている方法から言えば、再接続の試行を成功させるためには、これが最も確実で信頼性の高い手法となります。

TLS プロトコルの詳細については、『NetX Duo TLS Secure Module Guide』を参照してください。

デバイスあたりの MQTT クライアントの複数インスタンス

SSP 1.4.0 以前では、デバイスは MQTT クライアントの複数インスタンスを安全に実行できませんでした。これらのリリースの MQTT クライアントではグローバル変数を想定していたためです。この問題は、その後のリリースで解決されます。

NetX Duo MQTT クライアントモジュールの動作に関する重要な注意事項と制限事項

スレッドペインウィンドウの [+] 記号 -> [X-Ware] -> [NetX Duo] -> [Protocols] -> [NetX Duo MQTT Client] の順にクリックして、NetX Duo MQTT クライアントコンポーネントを追加します。

NetX Duo MQTT クライアントコンポーネントをプロジェクトに追加すると、セキュアな MQTT に必要な NetX Duo TLS コンポーネントを追加するためのオプションが自動的に追加されます。

MQTT クライアントのプロパティを次の表に示します。

Property	Value
Common	
NX Secure	Enable
Topic Name Max Length	12
Message Max Length	32
Keepalive Timer Rate (s)	1
Ping Timeout Delay (s)	1
Module g_mqtt_client0 NetX Duo MQTT Client	
Name	g_mqtt_client0
Client ID Callback	mqtt_client_id_callback
Client ID Max Length	12
Client Thread Stack Size	4096
Number of Messages to be stored in memory	1
Client thread priority	2
Name of generated initialization function	mqtt_client_init0
Auto Initialization	Enable

図 395:NetX Duo MQTT クライアントモジュールの MQTT クライアントブロックの構成可能なプロパティ

上の図で「Common」とされているプロパティは、プロジェクト内の MQTT クライアントのすべてのインスタンスで共通の、NetX Duo MQTT クライアントの構成可能なオプションです。「Module」のプロパティは、プロジェクト内の MQTT クライアントの各インスタンスに固有です。

共通プロパティ

- NX Secure：TLS のサポートを有効または無効にします。プロパティが [Enabled] に設定されている場合、MQTT クライアントは TLS サポートありでビルドされます。注：このプロパティを有効にするには、必要なソースコードをプロジェクトに提供するために、NetX Duo TLS コンポーネントをプロジェクトに追加する必要があります。追加しなかった場合、プロジェクトはビルドされません。[Disabled] に設定されている場合は、NetX Duo TLS コンポーネントを追加しても効果はありませんが、プロジェクトは引き続きビルドおよび実行されます。
- Topic Name Max Length: アプリケーションがサブスクライブするトピックの最大長です（バイト単位）。デフォルトは 12 バイトです。
- Message Max Length: アプリケーションが送信または受信するメッセージの最大長です（バイト単位）。デフォルトは 32 バイトです。
- Keepalive Timer Rate：このタイマは、最後に MQTT 制御メッセージが送信されてからの時間を追跡し、キープアライブ時間が経過する前に MQTT PINGREQ メッセージを送信するために使用されます。デフォルト値は 1 秒です。
- Ping Timeout Delay: MQTT クライアントが MQTT PINGREQ を送信した後、ブローカからの PINGRESP を待機する時間です。デフォルト値は 1 秒です。

モジュールのプロパティ

- Name：MQTT クライアントインスタンスの名前です。
- Name of generated initialization function：MQTT クライアントインスタンスを作成する初期化関数の名前です。デフォルトは、自動生成関数 mqtt_client_init0. です。
- Auto Initialization：初期化関数のコールを有効化または無効化します。無効にした場合、アプリケーションスレッドエントリ関数はクライアント ID を取得し、MQTT クライアントインスタンスを作成する必要があります。

- Client ID Callback：MQTT クライアントスレッドタスクで一意のクライアント ID を取得するために使用される、ユーザーによって提供されるコールバック関数です。自動初期化が無効である場合、これとクライアント ID 長には効果がありません。
- Client ID Max Length: クライアント ID の最大長（バイト単位）。
- Client Thread Stack Size：MQTT クライアントのスレッドスタックサイズ（バイト単位）です。
- Number of Messages to be stored in Memory：MQTT クライアントは、メモリ領域を使用してメッセージを格納します。MQTT クライアントの動作に必要なメモリは、送受信されるデータの量によって異なります。最小のメモリサイズは単一の MQTT_MESSAGE_BLOCK インスタンスのサイズであり、60 バイトです。デフォルト値は 1 MQTT_MESSAGE_BLOCK（60 バイト）です。ただし、アプリケーションで複数のメッセージを受信可能になる前に複数のメッセージが受信される場合、これは良い選択肢ではありません。TCP ソケットがデータの ACK を受信するまで、または QoS レベルが 1 以上の場合は MQTT クライアントが MQTT サーバーから ACK を受信するまでは、送信されたメッセージをリリースできません。そのため、モジュールガイドのプロジェクトでは 6 つのメッセージブロックを使用しています。この数は 3 つか 4 つにまで減らすこともできます。
- Client Thread Priority：MQTT クライアントのスレッドのプライオリティです。
- Name of Generated Initialization function：nxd_mqtt_client_create API をコールする関数の名前です。自動初期化が無効である場合、これには効果がありません。その場合、Synergy によりこの関数が自動的に作成されます。
- Auto Initialization：これは、[Name of generated initialization function] オプションで指定された関数がコールされるかどうかを決定します。有効に設定すると、その関数が呼び出されます。無効に設定した場合、アプリケーションは NetX Duo MQTT クライアントサービスを使用する前に、nxd_mqtt_client_create API をコールする必要があります。

一意のクライアント ID の設定

前述のように、MQTT クライアントインスタンスは、nxd_mqtt_client_create() API を使用して作成します。MQTT クライアントアプリケーションによって、ISDE での MQTT クライアント作成が可能になっている場合、クライアント ID コールバックを定義する必要があります。このコールバックは、定義済みのクライアント ID 文字列を使用して ISDE が内部で nxd_mqtt_client_create をコールするよりも前にコールされます。MQTT クライアントコンポーネントによって、MQTT クライアントプロパティのリスト内にある [Client ID Callback and Client ID Max Length] を設定できるようになります（上記の「モジュールのプロパティ」を参照してください）。

クライアント ID は一意である必要があります。MQTT ブローカはクライアント ID をパラメータの 1 つとして使用してクライアントを識別します。

クライアント ID コールバックのプロトタイプは次のとおりです。

```
void mqtt_client_id_callback(char * p_client_id, uint32_t * p_client_id_length);
```

p_client_id は取得するクライアント ID へのポインタなので、このコールバック関数によって設定される出力パラメータとなります。p_client_id_length はクライアント ID 長へのポインタなので、入力 / 出力パラメータとなります。このメカニズムにより、コンパイル時ではなく実行時にクライアント ID を決定できるようになります。

e2 studio の [Properties] ペインで [Client ID Callback] を空のままにすると、コンパイラエラーが発生します。NULL は許容されるエントリです。アプリケーションで MQTT クライアントを直接作成した方がよい場合は、このコールバックを NULL に設定し、[*Auto Initialization*] を [Disabled] に設定します。アプリケーションで nxd_mqtt_client_create API がコールされたら、クライアント ID 文字列を直接指定し、その文字列の長さを入力パラメータとして指定します。

```
/* Create MQTT client instance. */
nxd_mqtt_client_create(&mqtt_client, "my_client", CLIENT_ID_STRING,
  strlen(CLIENT_ID_STRING), ip_ptr, pool_ptr, (VOID*)mqtt_client_stack,
  sizeof(mqtt_client_stack), MQTT_THREAD_PRIORTY,
  (UCHAR*)client_memory, sizeof(client_memory));
```

次に示すのはクライアント ID コールバック関数のサンプル参照実装で、MAC アドレスをクライアント ID にコピーしています。

```
void mqtt_client_id_callback(char *p_client_id, uint32_t *p_client_id_length)
{

    uint32_t id_length;
    UCHAR mac_id[6]  = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06};

    if (*p_client_id_length < sizeof(mac_id))
    {
        id_length = *p_client_id_length;
    }
    else
    {
        id_length = sizeof(mac_id);
    }

     /* Copy MAC address to CLient ID and update client ID length */
     memcpy(p_client_id, mac_id, id_length);

    return;
}
```

注 :MQTT セッションでは、クライアント ID 文字列を長さゼロにすることができます。MQTT クライアントがゼロバイトのクライアント ID を提供する場合、そのクライアントでは、MQTT プロトコルに従い、nxd_mqtt_client_connect API での clean_session の入力を NX_TRUE（1）に設定しなければなりません。クライアントが、clean_session が NX_FALSE（0）に設定されたゼロバイトのクライアント ID を提供すると、サーバーは、CONNACK のリターンコード 0x02（識別子拒否）で CONNECT パケットに応答し、ネットワーク接続を閉じます。

- NetX Duo MQTT クライアントは、QoS レベル 2 のメッセージの送受信をサポートしません。
- NetX Duo MQTT クライアントでは、チェーンされたパケットはサポートされません。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.32.4 アプリケーションへの NetX Duo MQTT クライアントモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX Duo MQTT クライアントモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX Duo MQTT クライアントモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX Duo MQTT クライアントモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_mqtt_client0 NetX Duo MQTT Client	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo MQTT Client

次の図に示すように、NetX Duo MQTT クライアントモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

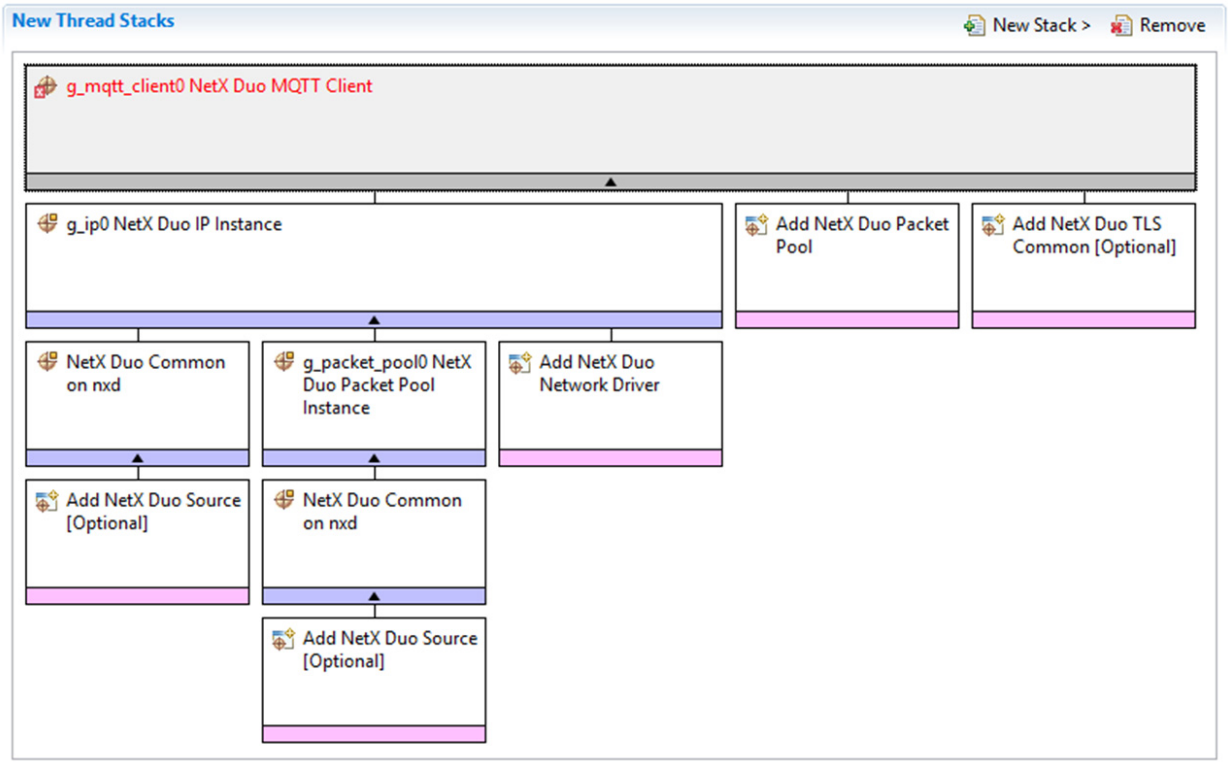


図 396:NetX Duo MQTT クライアントモジュールのスタック

上記のスタックで、NetX ネットワークドライバー (NetX Duo スタックの場合は NetX Duo ネットワークドライバー) はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上

に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.32.5 NetX Duo MQTT クライアントモジュールの構成

ユーザーは必要な動作に合わせて NetX Duo MQTT クライアントモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX Duo MQTT クライアントモジュールの構成設定

ISDE Property	Value	Description
NX Secure	Enable, Disable Default: Enable	This enables/disables TLS support. If this property is set to Enabled, the MQTT Client is built with TLS support. Note: enabling the property requires adding the NetX Duo TLS component to the project to supply the necessary source code to the project, or the project will not build. If set to Disabled, adding the NetX Duo TLS component has no effect though the project will still build and run.

ISDE Property	Value	Description
Topic Name Max Length	12	The maximum topic length (in bytes) the application is going to subscribe to. The default is 12 bytes.
Message Max Length	32	The maximum message length (in bytes) the application is going to send or receive. The default is 32 bytes.
Keepalive Timer Rate(s)	1	This timer is used to keep track of the time since last MQTT control message was sent, and sends out an MQTT PINGREQ message before the keep-alive time expires. The default value is 1 second.
Ping Timeout Delay(s)	1	The time MQTT client waits for PINGRESP from the broker for after it sends out MQTT PINGREQ. The default value is 1 second.
Name	g_mqtt_client0	Name of the MQTT client instance.
Client ID Callback	mqtt_client_id_callback	Callback function provided by user for the MQTT Client thread task to obtain a unique client ID. If Auto Initialization is disabled, this and the Client ID length have no effect.
Client ID Max Length	12	Maximum Length in bytes of the client ID.
Client Thread Stack Size	4096	MQTT Client thread stack size in bytes.

ISDE Property	Value	Description
Number of Messages to be stored in memory	1	MQTT client uses memory area to store messages. The memory needed for MQTT client operation depends on the amount of data being sent or received. The minimal memory size is the size of a single MQTT_MESSAGE_BLOCK instance which is 60 bytes. The default value is 1 MQTT_MESSAGE_BLOCK or 60 bytes. However, this it not a good choice if there will be multiple messages received before the application can receive them. Transmitted messages cannot be released until thee TCP socket receives an ACK for the data, or if the QoS level is 1 or higher and the MQTT Client has received an ACK from the MQTT server. So the module guide project uses 6 message blocks. That number can probably be reduced to 3 or 4.
Client thread priority	2	MQTT Client thread priority.
Name of generated initialization function	mqtt_client_init0	Name of the function that will call the nxd_mqtt_client_create API. If Auto Initialization is disabled, this has no effect. If it is, Synergy will create this function automatically

ISDE Property	Value	Description
Auto Initialization	Enable, Disable Default: Enable	This determines if the function specified in the Name of Generated Initialization function option is called. If set to Enable, it will invoke this function. Otherwise if set to Disable, the application must call the nxd_mqtt_client_create API before using any NetX Duo MQTT Client services.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

スタックモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、NX Secure を無効化するには、長めのクライアント ID 文字列が必要です。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX Duo MQTT クライアントのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	0,0,0,0	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection

ISDE Property	Value	Description
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo TLS 共通の構成設定

ISDE Property	Value	Description
Crypto Engine	Hardware	Crypto engine selection
Self Signed Certificates	Enable, Disable Default: Disable	Self signed certificates selection
PSK Cipher Suite	Enable, Disable Default: Disable	PSK cipher suite selection
X509 Strict Name Compare	Enable, Disable Default: Disable	X509 strict name compare selection
X509 Extended Distinguished Names	Enable, Disable Default: Disable	X509 extended distinguished names selection
Maximum RSA Modulus size (bits)	1024, 2048, 3072, 4096 Default: 4096	Maximum RSA modulus size (bits) selection
TLS v 1.0	Enable, Disable Default: Disable	TLS v 1.0 selection
TLS v 1.1	Enable, Disable Default: Disable	TLS v 1.1 selection
Server Mode	Enable, Disable Default: Enable	Server mode selection
Client Mode	Enable, Disable Default: Enable	Client mode selection

ISDE Property	Value	Description
Name of generated initialization function	nx_secure_common_init	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo ソフトウェア暗号化の構成設定

ISDE Property	Value	Description
Name	g_crypto_generic	Module name

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX 暗号化ハードウェアアクセラレータの構成設定

ISDE Property	Value	Description
Name	g_sf_el_nx_crypto	Module name

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

SCE 共通ドライバーの構成設定

ISDE Property	Value	Description
Name	g_sce_0	Module name

ISDE Property	Value	Description
Endian Flag	CRYPTO_WORD_ENDIAN_BIG, CRYPTO_WORD_ENDIAN_LITTLE Default: CRYPT_WORD_ENDIAN_BIG	Endian flag selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo MQTT クライアントモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX Duo MQTT クライアントモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I^2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1

Property	Value	Description
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.32.6 アプリケーションでの NetX Duo MQTT クライアントモジュールの使用

一般的なアプリケーションで NetX Duo MQTT クライアントモジュールを使用する際の手順は次のとおりです。

1) NX_IP_LINK_ENABLED オプションを指定して nx_ip_status_check（システムに複数のネットワークインタフェースがある場合は nx_ip_interface_status_check）をコールし、ネットワークリンクが有効化されるのを待機します。
2) tx_event_flags_create API を使用して、イベントフラググループを作成します。
3) nxd_mqtt_client_connect API を使用して、MQTT サーバー（ブローカ）に接続します。
4) nxd_mqtt_client_receive_notify_set API を使用して、受信通知コールバックを設定します。受信コールバックは、基礎となる NetX Duo ソケットサービスからこの接続のパケットを受信している旨の通知を受信すると、フラグを設定します。
5) nxd_mqtt_client_subscribe API を使用して、MQTT サーバーのトピックを購読します。
6) nxd_mqtt_client_publish API を使用して、トピックへメッセージをパブリッシュします。
7) tx_event_flags_get API をコールすることにより、メッセージの受信を待機します。

8) nxd_mqtt_client_message_get API を使用して、メッセージを受信します。ソケットからのパケットの受信と違い、MQTT クライアントがパケットの解放を考慮する必要はありません。パケットおよびメッセージブロックの割り当てと解放は、MQTT クライアントのスレッドタスクが処理します。
9) トピックを指定して、nxd_mqtt_client_unsubscribe API をコールすることにより、トピックを購読解除します。
10) nxd_mqtt_client_disconnect API をコールすることにより、トピックとの接続を解除します。

次の図は、一般的な手順を示した通常の動作フロー図です。

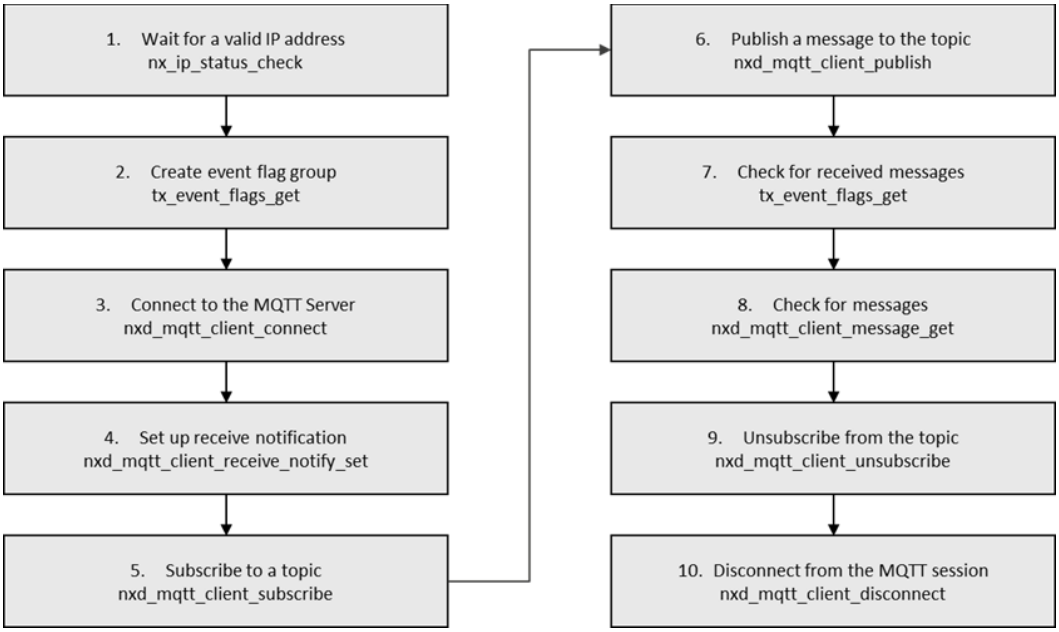


図 397:一般的な NetX Duo MQTT クライアントモジュールアプリケーションのフロー図

4.3.33 NetX Duo NAT

IP ネットワーク アドレス変換（NAT）は、複数のデバイスがインターネットへアクセスする必要があるものの、インターネットサービスプロバイダ（ISP）によって割り当てられている IPv4 インターネットアドレスが 1 つのみという局面において、IPv4 アドレスの数の制約の問題を解決できる技術です。NAT 対応のルータをパブリックネットワークとプライベートネットワークの間に設置して、内部のプライベート IPv4 アドレスと割り当てられたパブリック IPv4 アドレスの間の変換を行うことができます。これにより、プライベートネットワーク上の複数のデバイスが、同じパブリック IPv4 アドレスを共有できます。

4.3.33.1 NetX Duo NAT モジュールの特長

- NetX NAT は以下の RFC をサポートしています。
  - RFC 2663：IP ネットワークアドレス変換（NAT）の用語および懸念事項
  - RFC 3022：従来の IP ネットワークアドレス変換（従来の NAT）

- RFC 4787：ユニキャストユーザーデータグラムプロトコル（UDP）に関するネットワークアドレス変換（NAT）の動作上の要件

- NetX NAT は、以下のハイレベル API を提供します。
  - NAT サーバーの作成と削除
  - NetX Duo での NAT の無効化
  - NAT エントリテーブルがいっぱいの場合にアプリケーションに通知する、NAT のコールバックの設定
  - NAT テーブルでの静的インバウンドエントリの作成

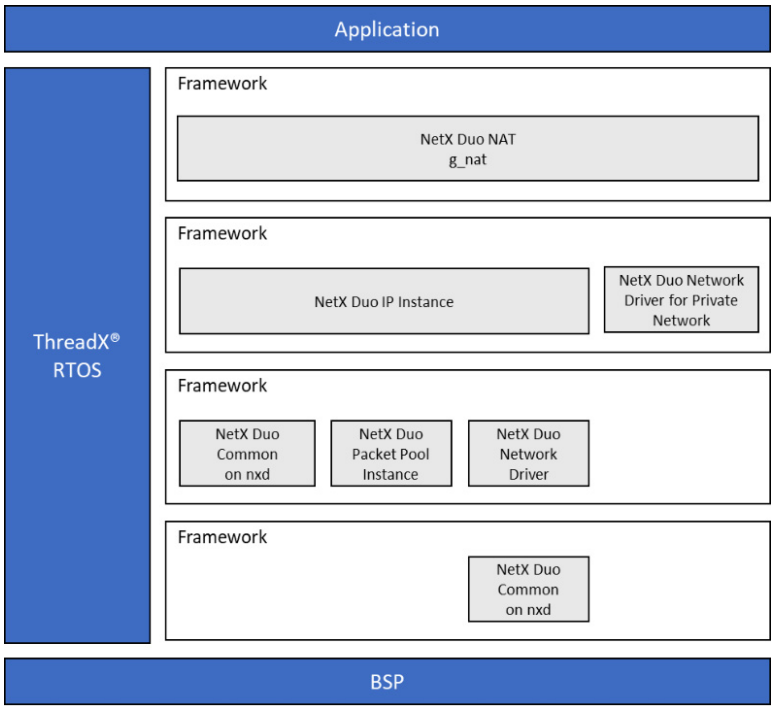


図 398:NetX Duo NAT モジュールのブロック図

注：上の図で、NetX Duo ネットワークドライバーモジュールには実装オプションが複数あります。その他の詳細については、セクション 4 のモジュールスタックの図の直後に記載されている説明を参照してください。

4.3.33.2 NetX Duo NAT モジュールの API の概要

NetX Duo NAT モジュールでは、作成、削除、および有効化操作のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX Duo NAT モジュール API の要約

Function Name	Example API Call and Description
nx_nat_create	nx_nat_create(nat_ptr, ip_ptr, global_interface_index, nat_cache, NX_NAT_ENTRY_CACHE_SIZE); Create a NAT Instance in which the network interface is the global interface (not the local/private network) with the specified network index.
nx_nat_delete	nx_nat_delete (nat_ptr); Delete a NAT instance.
nx_nat_enable	nx_nat_enable (nat_ptr); Enable the NAT server.
nx_nat_disable	nx_nat_disable (nat_ptr); Disable the NAT server.
nx_nat_cache_notify_set	nx_nat_cache_notify_set(nat_ptr, cache_full_notify_cb); Set the NAT cache full notify function to a user-defined notify function.
nx_nat_inbound_entry_create	nx_nat_inbound_entry_create(nat_ptr, entry_ptr, IP_ADDRESS(192,168,2,2), 5001, 5001, NX_PROTOCOL_TCP); Create an inbound translation table entry. This is typically used by application servers to allow clients to initiate a connection externally.

Function Name	Example API Call and Description
nx_nat_inbound_entry_delete	nx_nat_inbound_entry_delete(nat_ptr, delete_entry_ptr); Delete an inbound translation table entry.

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

ステータス戻り値

Name	Description
NX_SUCCESS	Successful NAT function
NX_PTR_ERROR*	Invalid input pointer parameter
NX_CALLER_ERROR*	Invalid caller (e.g. must be a thread) of a service
NX_NAT_PARAM_ERROR*	Invalid non pointer input
NX_NAT_CACHE_ERROR*	Cache memory not 4 byte aligned, or is too small
NX_NAT_PORT_UNAVAILABLE	Invalid external port for creating static entry
NX_NAT_ENTRY_NOT_FOUND	Entry to delete is not found in Cache table
NX_NAT_ENTRY_TYPE_ERROR*	Invalid entry (not static) to delete

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

注 : これらのエラーコードは、エラーチェックが有効化されている場合にのみ返されます。
NetX と NetX Duo でのエラーチェックサービスの詳細については、それぞれ『NetX User Guide for the Renesas Synergy ™ Platform』または『NetX Duo User's Guide for the Renesas Synergy ™ Platform』を参照してください。

4.3.33.3 NetX Duo NAT モジュールの動作の概要

NAT 対応のルータには、一般に 2 つのネットワークインタフェースがあります。1 つはパブリックインターネットに接続され、もう 1 つはプライベートネットワークに接続されます。このようなセットアップの一般的なルータは、宛先 IP アドレスに応じてプライベートネットワークとパブリックネットワーク間の IP データグラムのルーティングを行います。NAT 対応のルータは、アドレス変換を行ってからパブリックインタフェースとプライベートインタフェース間での IPv4 データグラムのルーティングを行います。変換は、内部ソースアドレスおよびソースポート番号のほ

か、外部宛先アドレスおよび宛先ポート番号に応じて、TCP セッションまたは UDP セッションごとに確立されます。ICMP のエコー要求および応答データグラムの場合は、ポート番号の代わりにインターネット制御通知プロトコル（ICMP）クエリ ID が使用されます。

通常、NAT の境界を越えた接続は、アウトバウンドパケットを外部ホストに送信するプライベートネットワーク上のホストによって開始されます。この場合、ホストには通常、動的（一時）IP アドレスが割り当てられます。当該のプライベートネットワークに外部ネットワークからのクライアント要求を受け入れる「サーバー」（HTTP または FTP など）がある場合は、接続を逆の方向で開始することもできます。NAT アプリケーションは、通常これらのローカルホストに静的（永続的な）IP アドレスポートを割り当てます。

以下の 3 つの図と関連する手順は、NAT ルータ経由でパケットを送信する場合の一連のイベントを説明しています。

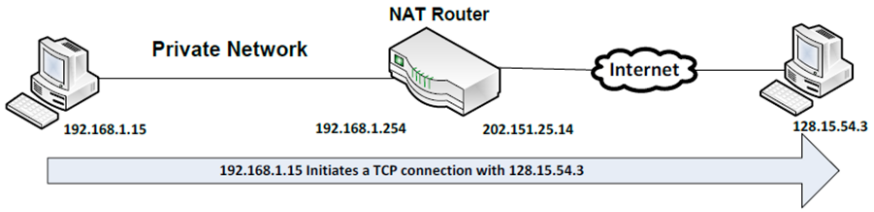


図 399:NetX Duo NAT モジュールの接続の開始

1) クライアントが TCP SYN メッセージを Web サーバーに送信します。プライベートネットワーク上のクライアントのアドレスは 192.168.1.15、ポート番号は 6732 です。宛先アドレスは 128.15.54.3、ポート番号は 80 です。
2) クライアントからのパケットが、NAT ルータによってプライベートネットワークインタフェースで受信されます。このパケットには、次のようなアウトバウンドトラフィック規則が適用されます。送信元（クライアント）のアドレスは NAT ルータのパブリック IP アドレス 202.151.25.14 に、送信元（クライアントの）のソースポート番号はパブリックインタフェースの外部向け送信用の TCP ポート番号 2015 に、それぞれ変換されます。
3) 次に、パケットがインターネット経由で送信され、最終的に宛先ホスト 128.15.54.3 に到着します。以下の図に示すように、受信側では送信元が 202.151.25.14、ポート番号 2015 と表示されているパケットが、実際にその IP アドレスおよびポートからリレーされていることに注意してください。

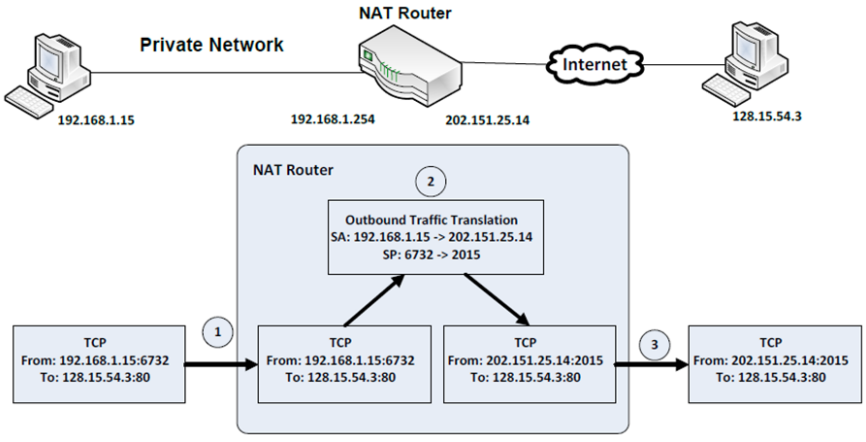


図 400:NetX Duo NAT モジュールのネットワークアドレス変換

4) ホスト 128.15.54.3 は、NAT ルータのインターネットアドレスをその宛先として指定した応答パケットを返します。
5) パケットが NAT ルータに到着します。これはインバウンドパケットであるため、次のようなインバウンド変換規則が適用されます。宛先アドレスは元の送信者（クライアント）の IP アドレス 192.168.1.15 に、宛先ポート番号は 6732 に、それぞれ戻され（変換され）ます。
6) 次に、パケットが、内部ネットワークに接続されたインタフェースを使用してクライアントに転送されます。

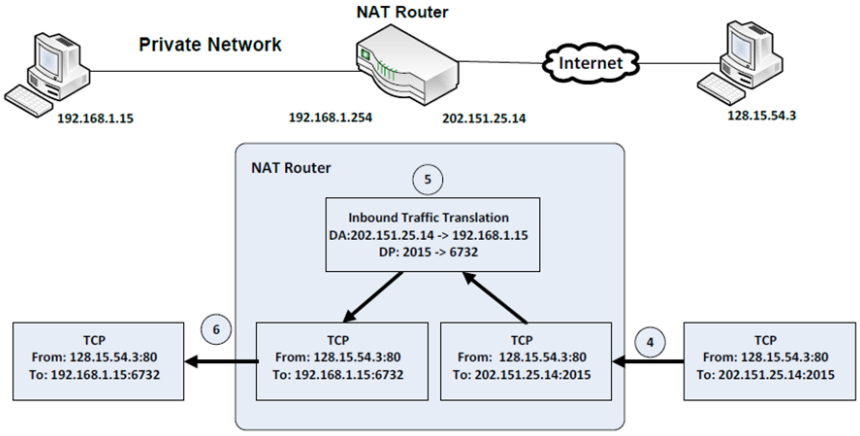


図 401:NetX Duo NAT モジュールパケットのリターンパス

NAT ルータ経由でパケットが送信された場合、送信元のインターネットネットワークアドレスおよびポート番号はパブリックインターネット上にある他のホストには公開されません。

ローカルネットワークと外部ネットワークの間のアクティブなすべての接続についてネットワークアドレスの変換の追跡を続けるために、NetX Duo NAT 対応のルータでは、各プライベートホストの接続に関する情報（送信元および宛先の IP アドレスおよびポート番号など）を持つ変換テーブルが維持されています。

NetX Duo NAT は、IPv4 ルータで使用することを想定しています。NAT が機能するためには、NetX Duo が受信パケットを内部の NetX Duo NAT ハンドラに転送するように構成されている必要があります。このハンドラによって、パケットがグローバルネットワーク（インバウンド）とプライベートネットワーク（アウトバウンド）のどちらから受信したものであるかが判定されます。

- インバウンドパケットの場合、ハンドラはこのパケットをプライベート（ローカル）ネットワーク上のホストへ転送（消費）できるかどうかを判定します。この判定を行うために、ハンドラは NAT 変換テーブルでパケットの宛先アドレスおよびポートに基づき一致するエントリを探します。一致するエントリが見つかると、ハンドラは宛先アドレスを一致するプライベートホストのIPアドレスおよびポートに変換して、そのパケットをホストへ送信します。エントリが見つからない場合、ハンドラは NetX Duo プロセスにパケットを通常どおり（パケットが NAT デバイス自体を対象としているかのように）処理するよう指示します。
- アウトバウンドパケットの場合、NAT ハンドラは宛先 IP アドレスを確認して、ハンドラがパケットをグローバルネットワークに転送できるか、NetX Duo が通常どおりパケットを処理すべきかを判定します。パケットにブロードキャストまたはループバック宛先アドレス、あるいは NAT デバイスのグローバルネットワークアドレスに一致しない IP アドレスがある場合、NAT ハンドラは NetX Duo にパケットを処理するように指示します。それ以外の場合、ハンドラは変換テーブルで送信元の IP およびアドレスの一致するエントリを探します。一致するエントリが見つかると、ハンドラは IP アドレスおよびポートをローカルの IP アドレスおよびポートに変換し、そのパケットをローカルホストに転送（消費）します。以前のエントリが見つからない場合、NAT は NAT 変換テーブルにエントリを作成し、送信元の IP アドレスをグローバルインタフェース用に変換し、そのパケットを外部ホストに転送します。

NetX Duo NAT モジュールの動作に関する重要な注意事項と制限事項

- NAT を有効化するには、NetX Duo のソースコンポーネントを [Configurator] ペインに追加し、NetX Duo ソースの [*NAT*] プロパティを設定します。ビルド済み NetX Duo ソースライブラリでは NAT は有効化されていません。
- NetX Duo ソース内でも、Maximum Physical Interfaces プロパティが、1 つのプライベートネットワークと 1 つのグローバルネットワークを想定して 2 に設定されている必要があります。自動生成されたコードは、グローバルネットワーク IP アドレスを使用して IP インスタンスを作成し、セカンダリインタフェースをプライベートネットワークインタフェースとしてアタッチします。
- ネットワークドライバーインスタンスは 2 つ存在する必要があります。このモジュール用のプロジェクトでは、両方のインタフェースに NetX ポート ETHER フレームワーク（sf_el_nx）を使用していますが、1 つのネットワークインターフェスを Wi-Fi やその他のネットワークメディアに配置することもできます。2 つの sf_el_nx ドライバーインスタンスを使用する場合は、これらのインスタンスの名前が同じではないこと、1 つがチャネル 0 を参照し、もう 1 つがチャネル 1 を参照していることを確認してください。デュアルポート設定のドライバーの MAC アドレスは 1 つの NetX ポート ETHER 構成内にあるため、これらを変更する必要はありませんが、チャネル 0 とチャネル 1 が同じになっていないことについては確認が必要です。
- コンフィギュレータで NAT インスタンスの [Auto initialization] プロパティが有効になっている場合（デフォルトでは有効）、[Name of the generated initialization function] プロパティで指定されている関数はセカンダリインタフェースとアタッチされ、NAT インスタンスを作成する必要があります。[Private IPv4 Address] プロパティは、NAT デバイス（サーバー）のローカル IP アドレスです。グローバルネットワークインタフェースインデックスは、グローバルネットワークが使用するネットワークインタフェースを指定します。デフォルトでは、このインデックスはゼロ（IP インスタンスのプライマリインタフェース）に設定されています。セカンダリインタフェースはローカルネットワーク（インタフェースインデックス 1）に設定されています。
- [*Auto Initialization*] が無効化されている場合、NAT アプリケーションとセカンダリインタフェースをアタッチし、NAT サービスを利用する前に NAT インスタンスを作成する必要があります。セカンダリインタフェースをアタッチしたら、アプリケーションは内部の NetX Duo の処理によって、nx_ip_interface_status_check API を使用してそのインタフェースのリンクが有効化されるのを待機する必要があります（この方法については、モジュールガイドのプロジェクト例を参照してください）。
- 実行時に、NetX Duo NAT フレームワークは、NAT 変換エントリを保管するための 4 バイトに調整されたテーブルまたはキャッシュも作成します。キャッシュのサイズは NAT インスタンスの [*Cache Size*] プロパティによって設定されます。デフォルト値は 1024 バイトです（NAT 変換レコードは 28 バイトです）。NetX Duo NAT 変換テーブルの最小サイズは 3 つのエントリです。この値は [*Minimum count for translation entry*] プロパティで設定されます。デフォルトは 3 ですが、多数のローカルホストがあり、トラフィックの多いネットワークではさらに大きい数に設定する必要があります。
- デフォルトでは、エントリは NAT サーバーによって、受信するインバウンドおよびアウトバウンドパケットが動的エントリの場合に作成されます。これらにはタイムアウト値が割り当てられます（[*Timeout for translation_entry*] プロパティ）。デフォルト値の 240 秒は、RFC 2663 によって推奨されるタイムアウトです。エントリのタイムアウト時間に達すると、そのエントリは削除対象としてマークされます。ただし、NetX Duo NAT サーバーの実装にはタイマがありません。このため、テーブルに期限切れのエントリがあるかどうかを確認し、期限切れのエントリがあれば削除する処理は、新規エントリがテーブルに追加されるときに行われます。期限切れのエントリがなく、テーブルがいっぱいの場合、NetX Duo NAT サーバーは cache full コールバックによってアプリケーションに通知します。アプリケーションでは、*nx_nat_cache_notify_set* サービスを使用してこのコールバックを設定できます。
- アプリケーションで nx_nat_inbound_entry_create サービスを使用することにより、期限切れになることのない静的なエントリを作成できます。これらのエントリはインバウンドパケットに対してのみ作成できるもので、「インバウンド規則」と呼ばれることもあります。静的エントリはサーバーアプリケーションでの利用を想定しており、クライアントがサーバーとの接続セッションを開始できるようにします。最初に、NAT はインバウンド規則で要求されているグローバルポートおよびローカルポー

トが利用できるかどうかを検証します。これらのエントリを削除するために、アプリケーションは nx_nat_inbound_entry_delete サービスをコールします。

- NetX Duo NAT は、さまざまな TCP、UDP、および ICMP 変換ポートを構成して、一意のローカルアドレス（外部ホストに接続するローカルホスト用のポートエントリ）を作成します。
  - TCP 変換ポートに使用できる TCP ポートは、最小値（[*Minimum assigned port number for outbound TCP packets*] プロパティ）と最大値（[*Maximum assigned port number for outbound TCP packets*] プロパティ）の間になります。
  - 同様に、UDP エントリの場合はポートが [*Minimum assigned port number for outbound UDP packets*] プロパティと [*Maximum assigned port number for outbound UDP packets*] プロパティの間になります。
  - ICMP パケットにはポートはありません。代わりに ICMP ID をエントリ変換ポートとして使用できます（[*Minimum ICMP query identifier*] と [*Maximum ICMP query identifier*] プロパティ）。
- NAT プロトコルを使用して NetX Duo でパケットの転送を開始するには、アプリケーションで nx_nat_enable サービスをコールします。NetX の転送をサスペンドするには、アプリケーションで nx_nat_disable サービスをコールします。
- NAT が有効化されると、NAT スレッドエントリアプリケーションは、グローバルネットワークとローカルネットワークの間におけるパケットのやり取りをNATの内部処理に任せます。一般にローカルネットワーク上のサーバーに対して行われるのと同様に、グローバルネットワーク上のホストがローカルホストに到達できるようにするために、インバウンド規則を随時作成することができます。
- インターネットグループ管理プロトコル（IGMP）はサポートされません。NetX Duo NAT がサポートするのは TCP、UDP、および ICMP のみです。
- NAT プロトコルは IPv6 パケット送信には適用されません。
- NetX Duo NAT には DNS サービスと DHCP サービスのいずれも含まれていませんが、NetX Duo NAT ではこれらのサービスをその NAT の動作と統合することができます。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.33.4 アプリケーションへの NetX Duo NAT モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX Duo NAT モジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX Duo NAT モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX Duo NAT モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_nat0 NetX Duo NAT	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo NAT

次の図に示すように、NetX Duo NAT モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

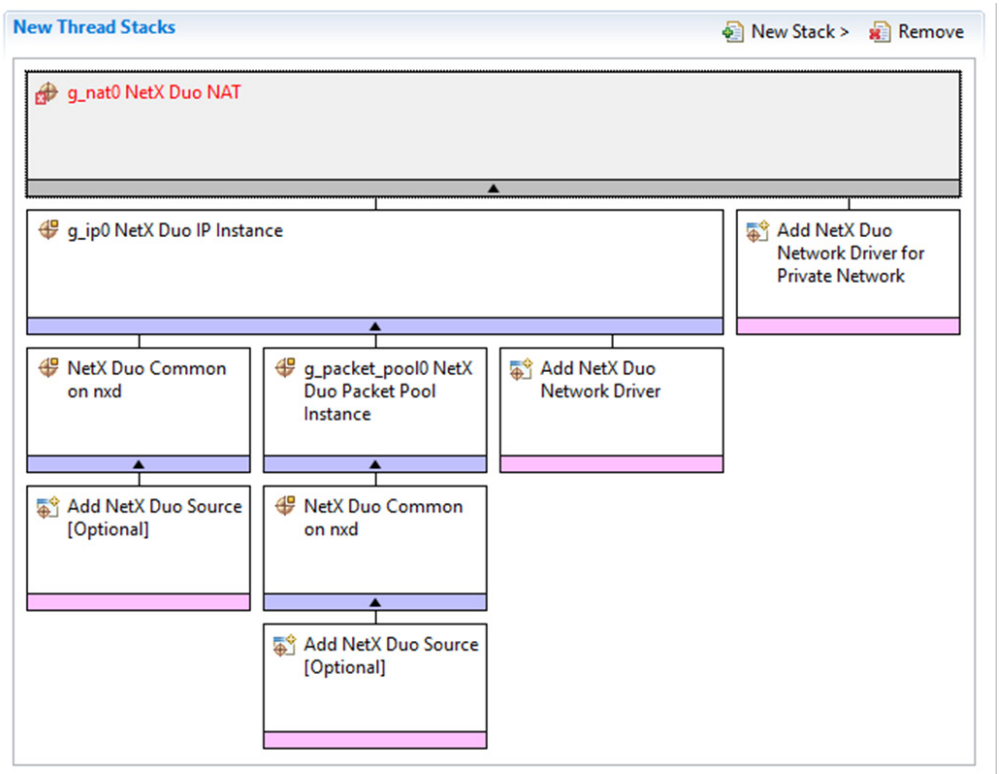


図 402:NetX Duo NAT モジュールのスタック

上記のスタックで、NetX ネットワークドライバー (NetX Duo スタックの場合は NetX Duo ネットワークドライバー) はまだ設定されていません。ネットワークドライバーには複数の選択肢があります。図や次の構成表では、必要以上に複雑になるのを避けるためにすべては示していません。使用可能なオプションは MCU ターゲットによって異なりますが、一般的なオプションは次のとおりです。

- nxd_ppp 上の PPP を使用する NetX Duo ポート
- sf_el_nx 上の NetX ポート ETHER
- sf_cellular_nsal_nx 上のセルラーフレームワークを使用する NetX ポート
- nx_ppp 上の PPP を使用する NetX ポート
- sf_wifi_nsal_nx 上で Wi-Fi フレームワークを使用する NetX ポート

4.3.33.5 NetX Duo NAT モジュールの構成

ユーザーは必要な動作に合わせて NetX Duo NAT モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX Duo NAT モジュールの構成設定

ISDE Property	Value	Description
Minimum count for translation entry	3	Minimum count for translation entry selection
Timeout for translation entry (ticks)	240	Timeout for translation entry selection
Minimum assigned port number for outbound TCP packets	20000	Minimum assigned port number for outbound TCP packets selection
Maximum assigned port number for outbound TCP packets	30000	Maximum assigned port number for outbound TCP packets selection
Minimum assigned port number for outbound UDP packets	20000	Minimum assigned port number for outbound UDP packets selection
Maximum assigned port number for outbound UDP packets	30000	Maximum assigned port number for outbound UDP packets selection

ISDE Property	Value	Description
Minimum ICMP query identifier	20000	Minimum ICMP query identifier selection
Maximum ICMP query identifier	30000	Maximum ICMP query identifier selection
Name	g_nat0	Module name
Cache size (bytes)	1024	Cache size selection
Private IPv4 Address (use commas for separation)	192,168,0,2	Private IPv4 Address selection
Private IPv4 Netmask (use commas for separation)	255, 255, 255, 0	Private IPv4 Netmask selection
Global network interface index	0	Global network interface index selection
Name of generated initialization function	nat_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

スタックモジュールの場合は、デフォルト以外の設定が望ましいこともあります。たとえば、異なる IP アドレスおよびサブネットマスクの選択が役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX Duo NAT のローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX Duo IP インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	0,0,0,0	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
IPv6 Link Local Address (use commas for separation, All zeros means use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	Enable, Disable Default: Enable	UDP selection
ICMP	Enable, Disable Default: Enable	ICMP selection

ISDE Property	Value	Description
IGMP	Enable, Disable Default: Enable	IGMP selection
IP fragmentation	Enable, Disable Default: Disable	IP fragmentation selection
Name of generated initialization function	ip_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo パケットプールインスタンスの構成設定

ISDE Property	Value	Description
Name	g_packet_pool0	Module name

ISDE Property	Value	Description
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo NAT モジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX Duo NAT モジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I^2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.33.6 アプリケーションでの NetX Duo NAT モジュールの使用

次の例は、実際に使用できる有効な IP、ARP、ICMP、TCP、UDP が確立済みで、リンクが実行されているシステムを前提にしています。

一般的なアプリケーションで NetX Duo NAT モジュールを使用する際の手順は次のとおりです。

1) プライマリ（「グローバル」）インタフェースの nx_ip_status_check API を使用して、IP インスタンスがドライバーを初期化し、有効な IP アドレスを取得するのを待機します。
2) nx_ip_interface_status_check API を使用して、IP のセカンダリ（「ローカル」）インタフェースが現時点で有効な IP アドレスを取得するのも待機します。

3) nx_nat_cache_notify_set API をコールすることにより、NAT 変換テーブルがいっぱいの場合に cache full コールバックに通知されるように設定します（オプション）。
4) nx_nat_enable API をコールすることにより、NAT プロトコルに従った NetX 内の IP レイヤーによるパケット転送を開始します。残りの処理は、IP スレッドタスクおよび NAT サービスによって行われます。
5) nx_nat_inbound_entry_create[Optional]. をコールすることにより、必要に応じて静的エントリを追加します（たとえば、クライアント要求を待機するサーバーアプリケーション）（オプション）。これは、NAT サービスを有効化する前後に行うことができます。
6) nx_nat_disable API をコールして、NAT をサスペンドします。
7) nx_nat_delete API をコールして、NAT を削除します。

次の図は、一般的な手順を示した通常の動作フロー図です。

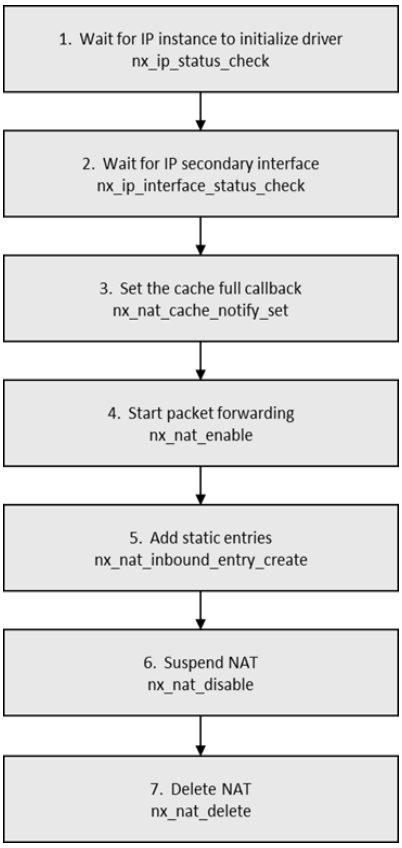


図 403:一般的な NetX Duo NAT モジュールアプリケーションのフロー図

4.3.34 NetX Duo TLS セッション

NetX Duo TLS セッションモジュールは、トランスポート層セキュリティ（TLS）プロトコルベースのクライアントにハイレベルな API を提供します。このモジュールでは、Synergy Crypto Engine（SCE）によって提供される、ハードウェアによって高速化された暗号化および復号を実行するサービスが使用されます。

NetX Duo TLS セッションモジュールは、セキュアソケットレイヤー（SSL）プロトコルおよびそれに代わるトランスポート層セキュリティ（TLS）プロトコル（RFC 2246（バージョン 1.0）および RFC 5246（バージョン 1.2）を参照）を実装している ThreadX「NetX Duo Secure」を基盤としています。NetX Duo Secure には、基本的な X.509（RFC 5280）用のルーチンも含まれています。NetX Duo Secure は、ThreadX RTOS を使用するアプリケーション用です。

TLS/SSL プロトコルは、2 つのアプリケーション間の通信にプライバシと信頼性を提供します。次のような基本特性を備えています。

- 暗号化 : 通信するアプリケーション間で交換されるメッセージは暗号化され、接続のプライベート性が保証されます。データの暗号化には、AES（Advanced Encryption Standard）などの対称暗号メカニズムが使用されます。
- 認証 : 証明書を使用してピアの ID を確認するメカニズムです。
- 整合性 : メッセージの改ざんや偽造を検出して接続が信頼できることを保証するメカニズムです。メッセージの整合性を保証するには、セキュアハッシュアルゴリズム（SHA）などのメッセージ認証コード（MAC）が使用されます。

Renesas Web サイトで、MQTT クライアントの一部としての TLS の利用をデモンストレーションするアプリケーションプロジェクトを利用できます。「Synergy Enterprise Cloud Toolbox」アプリケーションプロジェクトおよびアプリケーションノートは、Renesas の Web サイトで「R20AN0485」を検索すると見つかります。

4.3.34.1 NetX Duo TLS セッションモジュールの特長

- RFC 2246- TLS プロトコルバージョン 1.0
- RFC 5246- トランスポート層セキュリティ（TLS）プロトコルバージョン 1.2
- RFC 5280 X.509 PKI 証明書（v3）
- RFC 3268- トランスポート層セキュリティ（TLS）向け Advanced Encryption Standard（AES）暗号スイート
- 公開キー暗号化標準（PKCS）#1: RSA 暗号仕様バージョン 2.1（RFC 3447）
- HMAC: メッセージ認証用キー付きハッシュ（RFC 2104）
- US セキュアハッシュアルゴリズム（SHA および SHA ベースの HMAC と HKDF）（RFC 6234）
- TLS 向け事前共有キー暗号スイート（RFC 4279）
- 以下の TLS 拡張をサポート：
  - Secure Renegotiation Indication：この拡張は、再ネゴシエーションハンドシェイクの間に発生する可能性がある、中間者攻撃に対する脆弱性を最小化します。
  - Server Name Indication：この拡張は、TLS クライアントが TLS サーバーに対して特定の DNS 名を指定できるようにし、サーバーが正しい資格情報を選択できるようにします（当該のサーバーに、複数の ID 証明書とネットワークエントリポイントがあることを前提とします）。
  - Signature Algorithms：この拡張は、TLS クライアントが TLS サーバーに受け入れ可能な署名とハッシュアルゴリズムのリストを提供できるようにします。
- 以下の X.509 拡張をサポート：
  - Key Usage：ビットフィールドにおける証明書の公開鍵の許容される使用方法について指定します。

– Extended Key Usage：OID を使用する証明書の公開鍵の許容される使用方法について追加で指定します。

- Subject Alternative Name：同じ証明書で表される別の DNS 名を指定します。

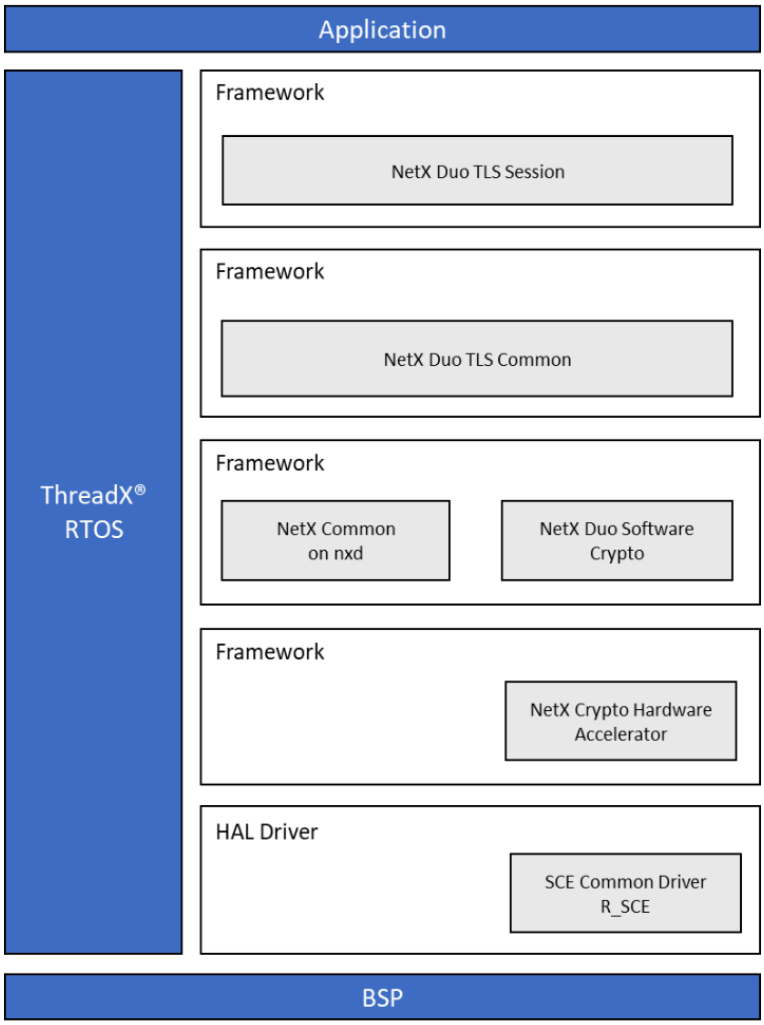


図 404:NetX Duo TLS セッションモジュールのブロック図

4.3.34.2 NetX Duo TLS セッションモジュールの API の概要

NetX Duo TLS サポートモジュールでは、TLS セキュリティセッションを作成し、設定するための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

NetX Duo TLS セッションモジュール API の要約

Function Name	Example API Call and Description
nx_secure_tls_local_certificate_add	nx_secure_tls_local_certificate_add(tls_session, certificate); Adds an initialized certificate to a TLS session for use as a local identification certificate - the TLS Server certificate for TLS servers, and the Client certificate for TLS clients.
nx_secure_tls_local_certificate_remove	nx_secure_tls_local_certificate_remove(tls_session, common_name, common_name_length); Removes a certificate instance from the local certificates list, keyed on the Common Name field.
nx_secure_tls_metadata_size_calculate	nx_secure_tls_metadata_size_calculate(cipher_table, metadata_size); Determines the size of the buffer needed by TLS for encryption metadata for a given ciphersuite table
nx_secure_tls_packet_allocate	nx_secure_tls_packet_allocate(tls_session, pool_ptr, packet_ptr, wait_option); Allocates a packet for a TLS application such that it allows additional room for the TLS header
nx_secure_tls_remote_ certificate_allocate	nx_secure_tls_remote_certificate_allocate(tls_session, certificate, raw_certificate_buffer, buffer_size); Adds an uninitialized certificate instance to a TLS session for the purpose of allocating space for certificates provided by a remote host during a TLS session

Function Name	Example API Call and Description
nx_secure_tls_session_certificate_callback_set	nx_secure_tls_session_certificate_callback_set(tls_session, session); Sets up a function pointer that TLS will invoke when a certificate is received from a remote host, allowing the application to perform validation checks such as certificate revocation and certificate policy enforcement
nx_secure_tls_session_create	nx_secure_tls_session_create(session_ptr, cipher_table, metadata_area, metadata_size); Initializes a TLS session control block for later use in establishing a secure TLS session over a TCP socket or other lower-level networking protocol
nx_secure_tls_session_client_verify_disable	nx_secure_tls_session_client_verify_disable(tls_session); Disables Client Certificate Verification for a particular TLS Session which previously had it enabled.
nx_secure_tls_session_client_verify_enable	nx_secure_tls_session_client_verify_enable (tls_session); Enables Client Certificate Verification for TLS Server instances. If enabled, the TLS Server will request and verify a remote TLS Client Certificate using all available crypto signature routines.
nx_secure_tls_session_delete	nx_secure_tls_session_delete(tls_session); Deletes a TLS session object, returning any resources to the system

Function Name	Example API Call and Description
nx_secure_tls_session_end	nx_secure_tls_session_end(tls_session, wait_option); Ends an active TLS session by sending the TLS CloseNotify alert to the remote host, then waiting for the response CloseNotify before returning.
nx_secure_tls_session_packet_buffer_set	nx_secure_tls_session_packet_buffer_set(session_ptr, buffer_ptr, buffer_size); Sets the buffer TLS uses to reassemble incoming messages which may span multiple TCP packets.
nx_secure_tls_session_protocol_oversion_override	nx_secure_tls_session_protocol_version_override(tls_session, protocol_version); Overrides the TLS protocol version to use for the TLS session. This allows for a different version of TLS to be utilized even if a newer version is enabled.
nx_secure_tls_session_receive	nx_secure_tls_session_receive(tls_session, packet_ptr_ptr, wait_option); Receives data from an active TLS session, handling all decryption and verification before returning the data to the caller in the supplied NX_PACKET structure
nx_secure_tls_session_reset	nx_secure_tls_session_reset(session_ptr); Resets a TLS session object, clearing out all data for initialization or re-use.
nx_secure_tls_session_send	nx_secure_tls_session_send(tls_session, packet_ptr, wait_option); Sends data using an active TLS session, handling all encryption and hashing before sending data over the established TCP socket connection

Function Name	Example API Call and Description
nx_secure_tls_session_start	nx_secure_tls_session_start(tls_session, tcp_socket, wait_option); Starts a TLS session given a TCP socket. The TCP connection must be established before calling this function or the TLS handshake will fail.
nx_secure_tls_session_time_function_set	nx_secure_tls_session_time_function_set(tls_session, time_func_ptr); Sets up a function pointer that TLS will invoke when it needs to get the current time, which is used in various TLS handshake messages and for verification of certificates.
nx_secure_tls_trusted_ certificate_add	nx_secure_tls_trusted_certificate_add(tls_session, certificate); Adds an initialized certificate to a TLS session for use as a trusted Root Certificate
nx_secure_tls_trusted_certificate_remove	nx_secure_tls_trusted_certificate_remove(tls_session, common_name, common_name_length); Removes a certificate instance from the trusted certificates store, keyed on the Common Name field.
nx_secure_tls_remote_certificate_free_all	nx_secure_tls_remote_certificate_free_all(NX_SECURE_TLS_SESSION *session_ptr); Release certificates previously registered with the TLS session.

Function Name	Example API Call and Description
**nx_secure_tls_psk_add	nx_secure_tls_psk_add(tls_session, pre_shared_key, psk_length, psk_identity, identity_length, hint, hint_length); Adds a pre-shared key (PSK) to a TLS session for use with a PSK ciphersuite. The second parameter is the PSK identity used during the TLS handshake to select the proper key.
**nx_secure_tls_psk_find	nx_secure_tls_psk_find(tls_session, psk_data, psk_length,psk_identity, identity_length); Finds a pre-shared key (PSK) in a TLS session for use with a PSK ciphersuite. The PSK is found using an "identity hint" that should match a field in the PSK structure in the TLS session.
**nx_secure_tls_client_psk_set	nx_secure_tls_client_psk_set(tls_session, pre_shared_key, psk_length, psk_identity, identity_length, hint, hint_length); Sets the pre-shared key (PSK) for a TLS Client in a TLS session control block for use with a remote server that is sing a PSK ciphersuite.
nx_secure_x509_certificate_initialize	nx_secure_x509_certificate_initialize(NX_SECURE_X509_CERT *certificate_ptr,const UCHAR *certificate_data,USHORT certificate_data_length,UCHAR *raw_data_buffer,USHORT buffer_size,const UCHAR *private_key_data,USHORT private_key_data_length,UINT private_key_type); Initialize X.509 Certificate for NetX Secure TLS

Function Name	Example API Call and Description
nx_secure_x509_common_name_dns_check	nx_secure_x509_common_name_dns_check(&certificate, dns_tld, dns_tld_length) Check DNS name against X.509 Certificate
nx_secure_x509_crl_revocation_ check	nx_secure_x509_crl_revocation_check(&crl_data, crl_length, &cert_store, &certificate) Check X.509 Certificate against a supplied Certificate Revocation List

注 : 関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細については、「参考文献」セクションの関連する『Express Logic User's Manual』を参照してください。

*注 :**TLS 共通コンポーネントで [PSK Cipher Suite of the TLS Common component] プロパティが有効化されている必要があります。*

4.3.34.3 NetX Duo TLS セッションモジュールの動作の概要

TLS は TCP を使用し、HTTP や MQTT などのアプリケーションレイヤープロトコルを対象にセキュアな通信を実現します。TLS は「ベア」TCP ソケットアプリケーションにおいて、別の TCP ピアに対するセキュアセッションで TCP パケットの送受信を行うためにも使用できます。このプロジェクトのモジュールガイドでは、シンプルな TLS のアプリケーションを使用して、TCP ピアとの間でデータをやり取りする TLS クライアントの TCP ソケットおよび TLS サーバーの TCP ソケットについて説明します。

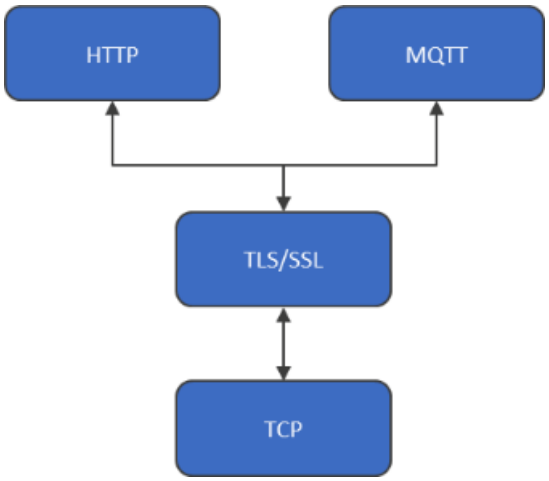


図 405:NetX Duo TLS セッションモジュール TLS/SSL 階層化

TLS にはウェルノウンポート番号はありません。代わりに、上位レイヤープロトコルのセキュア版の指定ポート番号を使用します。たとえば、セキュア HTTP の場合はポート番号 443、MQTT の場合はポート番号 8883 になります。

TLS/SSL を使用してセキュアな接続が確立されると、クライアント（常に接続を開始します）とサーバーの間でたとえば HTTPS などを使用してメッセージが交換されます。次の図で示すように、最初のメッセージセットによってハンドシェイクプロトコルが実行された後、クライアントとサーバーは双方向にデータをセキュアに送受信できます。

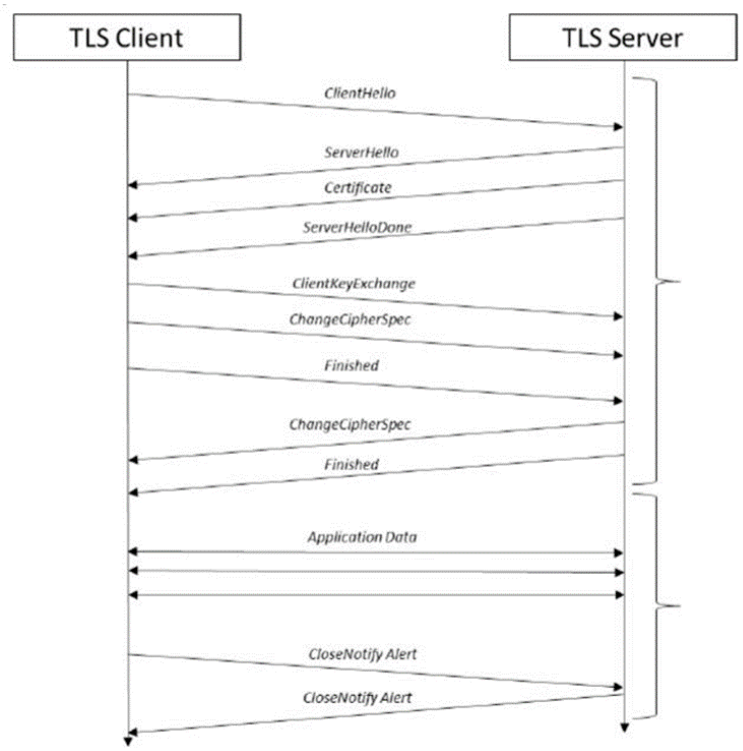


図 406:NetX Duo TLS セッションモジュール TLS プロトコルのシーケンス

NetX Duo TLS セッションモジュールの動作に関する重要な注意事項と制限事項

- TLS セッションを作成する前に、メタデータバッファを割り当てる必要があります。メタデータのサイズは、[Properties] ペインを使用して設定できます。また、ユーザーは NetX Duo Secure の nx_secure_metadata_size_calculate API を使用して、メタデータバッファの必要なサイズを計算できます。メタデータのサイズは TLS セッションコンポーネントの [Meta data size] で指定するか、nx_secure_tls_session_create コールで指定します。デフォルト値は 4KB ですが、多くのサーバーに対応するために、メモリスペースが利用できるのであれば 8KB をお勧めします。
- パケット再アセンブリバッファを TLS セッションに関連付けるには、nx_secure_tls_session_packet_buffer_set API を使用します。再アセンブリバッファは、複数の TCP パケットにまたがる可能性のある受信 TLS レコードを格納するために使用されます。受信 TLS レコードが提供されているバッファより大きい場合、TLS セッションはエラーで終了します。妥当なパケットバッファのサイズは 6 ～ 8KB です。
- また、TLS クライアントセッションを開始する前に、アプリケーションでは nx_secure_tls_remote_certificate_allocate コールでサーバー証明書データを処理するためのメモリを割り当てておく必要があります。大半の証明書に対して妥当なサイズは 2KB です。TLS クライアントは、ほとんどのサーバーから 2 ～ 3 の証明書を受け入れる必要があります。
- 受信する証明書のすべてに対して、NetX Duo Secure TLS は基本的な X.509 パス検証を実行します。
- また、検証プロセスの各ステージにおいて、各証明書の有効期限の日付が、アプリケーションのタイムスタンプ関数によって提供される日時と比較してチェックされます。オプションで、TLS が現在の日時を取得する必要があるときに呼び出すアプリケーションタイムスタンプ関数に対する関数ポインタを設定するには、nx_secure_tls_session_time_function_set API を使用します。現在の日時は、一部の TLS ハンドシェイクメッ

セージおよび証明書の検証に使用されます。TLS セッションにタイムスタンプ関数が登録されると、ServerHello または ClientHello の生成と、リモート証明書の検証にタイムスタンプが使用されます。

- 同じ TLS サーバーまたは別の TLS サーバーへの再接続を試行する前に、TLS クライアントで TLS セッションをクリアしておく必要があります。これを最も簡単に行う方法は、nx_secure_tls_session_end. をコールすることです。別の接続を試行する前に、TLS セッションを削除して、再作成することをお勧めします。SSP 1.3.x を使用するアプリケーションの場合、nx_secure_tls_session_create コールの前に TLS セッションで memset コールを実行して、completely:memset(tls_session_ptr, 0, sizeof(NXD_SECURE_TLS_SESSION)); セッションを完全にクリアする必要があります。
- ユーザーは「Netx Duo TLS Common」モジュールを使用して、最大の RSA モジュールサイズをビット単位で構成できます。最大の RSA モジュールサイズが 4096 ビットの場合、暗号化 / 復号など、RSA を対象とした暗号動作はソフトウェアで実施されます。ユーザーが最大の RSA モジュールサイズを 1024 または 2048 ビットとして選択した場合、暗号化動作は Synergy Crypto Engine（SCE）ハードウェアアクセラレータを使用して実行されます。
- 内蔵デバイスの特性のため、システムによっては、最大 TLS レコードサイズである 16KB をサポートするのに十分なメモリを用意できない場合があります。NetX Duo TLS Secure は、十分なリソースのあるデバイスでは 16KB のレコードを処理できます。
- NetX Duo Secure は、証明書の基本的な検証のみを実行します。NetX Duo TLS Secure は、証明書に対して基本的な X.509 チェーン検証を実行して、証明書が有効であり、信頼できる証明機関によって署名されていることを確認します。また、リモートホストの最上位のドメイン名と比較するための証明書共通名をアプリケーションに提供できます。ただし、証明書の拡張機能および他のデータの検証は、アプリケーションの実装者が行う必要があります。詳細については、Synergy Gallery にある『Express Logic NetX Duo TLS Secure User Guide』を参照してください。
- ソフトウェアベースの暗号化はプロセッサへの負荷が高く、利用できません。したがって、NetX Duo TLS Secure では、最適なパフォーマンスを実現するためにハードウェアベースの暗号化を採用しています。
- このモジュールの使用に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.34.4 アプリケーションへの NetX Duo TLS セッションモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに NetX Duo TLS セッションモジュールのいずれか、または両方を組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

NetX Duo TLS セッションモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

NetX Duo TLS セッションモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_tls_session0 NetX Duo TLS Session	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo TLS Session

次の図に示すように、NetX Duo TLS セッションモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

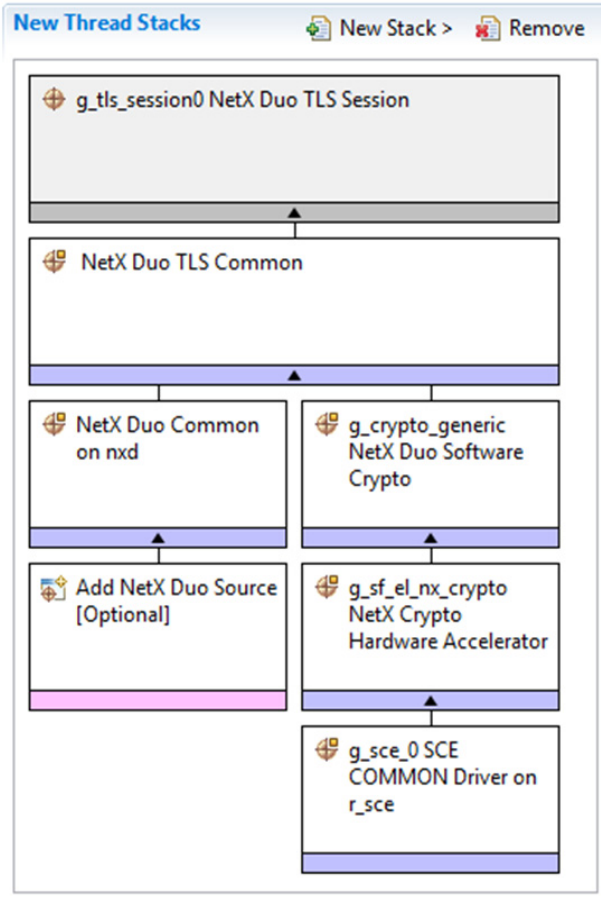


図 407:NetX Duo TLS セッションモジュールのスタック

4.3.34.5 NetX Duo TLS セッションモジュール

ユーザーは必要な動作に合わせて NetX Duo TLS セッションモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます (ブロックが赤で強調表示される)。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

NetX Duo TLS セッションモジュールの構成設定

ISDE Property	Value	Description
Name	g_tls_session0	Module name
Meta data size	4000	Meta data size selection
*Name of Timestamp function	tls_timestamp_callback0	Name of timestamp function
**Name of Certificate Verification function	certificate_verification_callback0	Name of certificate verification function
Name of generated initialization function	tls_dtls_session_init0	Name of generated initialization function
Auto initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注 : この Time Stamp 関数については別途取り上げていますが、このモジュールガイドでは複雑さを回避するために割愛されています。この関数は一般に、各証明書の有効期限日をアプリケーションによって指定されている時期と比較して確認するために使用されます。timestamp コールバックを使用しないと、相互運用性に関する問題が発生し、TLS アプリケーションセッションの生産リリースのセキュリティが低下する可能性があります。*

*注 :** 証明書検証関数では、検証対象の証明書をアプリケーションに提供するため、追加の検証手順が実行される可能性があります。このモジュールガイドプロジェクトでは複雑になるのを避けるために NULL に設定されていますが、実際の TLS セッションでは検証ツールを指定します。*

スタックモジュールに対するデフォルト値以外の設定を無効化できる場合もあります。たとえば、イーサネットポートに対して異なるアドレスを選択するときに役立つ場合があります。ローレベルスタックモジュールで構成可能なプロパティについては、参照のためにすべてこの後のセクションで記載しています。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

NetX Duo TLS セッションのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

NetX Duo 共通インスタンスの構成設定

ISDE Property	Value	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo TLS 共通の構成設定

ISDE Property	Value	Description
Crypto Engine	Hardware	Crypto engine selection
Self Signed Certificates	Enable, Disable Default: Disable	Self signed certificates selection
PSK Cipher Suite	Enable, Disable Default: Disable	PSK cipher suite selection
X509 Strict Name Compare	Enable, Disable Default: Disable	X509 strict name compare selection
X509 Extended Distinguished Names	Enable, Disable Default: Disable	X509 extended distinguished names selection
Maximum RSA Modulus size (bits)	1024, 2048, 3072, 4096 Default: 4096	Maximum RSA modulus size (bits) selection

ISDE Property	Value	Description
TLS v 1.0	Enable, Disable Default: Disable	TLS v 1.0 selection
TLS v 1.1	Enable, Disable Default: Disable	TLS v 1.1 selection
Server Mode	Enable, Disable Default: Enable	Server mode selection
Client Mode	Enable, Disable Default: Enable	Client mode selection
Name of generated initialization function	nx_secure_common_init	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo ソフトウェア暗号化の構成設定

ISDE Property	Value	Description
Name	g_crypto_generic	Module name

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX 暗号化ハードウェアアクセラレータの構成設定

ISDE Property	Value	Description
Name	g_sf_el_nx_crypto	Module name

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

SCE 共通ドライバーの構成設定

ISDE Property	Value	Description
Name	g_sce_0	Module name
Endian Flag	CRYPTO_WORD_ENDIAN_BIG, CRYPTO_WORD_ENDIAN_LITTLE Default: CRYPT_WORD_ENDIAN_BIG	Endian flag selection

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

NetX Duo TLS セッションモジュールのクロック構成

ETHERC ペリフェラルモジュールは PCLKA をそのクロックソースとして使用します。PCLKA 周波数を設定するには、ビルドの前に SSP コンフィギュレータの [clock] タブを使用するか、実行時に CGC インタフェースを使用します。

NetX Duo TLS セッションモジュールのピン構成

ETHERC ペリフェラルモジュールは、MCU デバイスのピンを使用して外部デバイスと通信します。I/O ピンは、必要に応じて外部デバイスで選択して構成する必要があります。次の表は [SSP configuration] ウィンドウ内でのピンの選択方法を示し、その次の表は I^2C ピンの選択例を示しています。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

ETHERC モジュールのピン選択

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

注 : 選択シーケンスでは、ETHERC1 がドライバーに必要なハードウェアターゲットであることを想定しています。

ETHERC1 のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, RMII Default: Disabled	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only Default: _A only	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.35 Express Logic 社の USBX の概要

SSP には、Express Logic 社の USBX USB スタック（ux）が組み込まれています。このドキュメントでは、Express Logic 社の SSP 用 USBX インタフェースの概要と要約を取り上げます。SSP の具体的な USBX モジュールには、それぞれ独自の概要セクションがあります。特定のモジュールの設計に関する詳細については、『SSP ユーザーズマニュアル』の該当するモジュールの概要セクションを参照してください。

4.3.35.1 Express Logic 社の USBX モジュールの機能

USBX は、既存の 2 つの USB 仕様である 1.1 および 2.0 に対応します。拡張性に優れた設計となっており、デバイスが 1 つのみ接続された単純な USB トポロジだけではなく、複数のデバイスとカスケードハブが接続された複雑なトポロジにも対応します。USBX は、ホスト側とデバイス側の両方をサポートします。

4.3.35.2 Express Logic 社の USBX でサポートされる USB クラス

以下に示す USBX クラスモジュールは、Synergy 構成ツールで完全にサポートされています。これらの USBX クラスモジュールを使用するには、Synergy 構成ツール（configuration.xml）の [Threads] タブに移動して、以下のいずれかの USB クラスモジュールを選択します。

- ux_device_class_cdc_acm
- ux_device_class_hid
- ux_device_class_storage
- ux_host_class_cdc_acm
- ux_host_class_hid
- ux_host_class_storage
- ux_host_class_video
- ux_host_class_hub

注 :USBX クラスの仕様の詳細については、『USBX Device Class User Guide』、『USBX Host Class User Guide』または『USBX Host Stack UVC User Guide』を参照してください。上記に列記した各モジュールには、『SSP ユーザーズマニュアル』に関連するモジュールの概要セクションがあります。また、Renesas Synergy の Web サイトで、各モジュールのモジュールガイドを入手することもできます。

下記に列記された USBX クラスモジュールは、Synergy パーツに対する試験的なモジュールです。

現在、これらのモジュールは、Synergy 構成ツールではサポートされていません。これらの USBX クラスモジュールを試験するには、Synergy 構成ツール（configuration.xml）の [Components] タブに移動して、いずれかの USB クラスモジュールを選択します。

- ux_device_class_cdc_ecm
- ux_device_class_rndis
- ux_host_class_audio
- ux_host_class_gser
- ux_host_class_printer
- ux_host_class_prolific
- ux_host_class_swar
- ux_network_driver

注 : 上記の USB クラスはいずれも試験的なものであり、Synergy パーツに対する試験はまだ行われていません。そのため、製品開発への使用は推奨されません。

上記のコンポーネントを追加すると、ビルド済みのアプリケーションコードライブラリも追加されます。上記の各コンポーネントには、保護ソースファイルを含む、末尾が「'_src'」の類似のコンポーネントがあります。ビルド済みのライブラリモジュールのほかに、「'_src'」コンポーネントを追加することが可能です。

USBX モジュールを使用したアプリケーションの作成

USBX モジュールを使用したアプリケーションの作成方法に関する詳細については、『SSP ユーザーズマニュアル』の関連するモジュールの概要セクションを参照してください。また、関連する各モジュールガイドには、一般的なアプリケーションでのモジュールの使用方法を説明した、実際に動作するアプリケーションプロジェクトも用意されています。モジュールガイドは、Renesas Synergy Web サイトから入手できます。

非推奨モジュールを使用するアプリケーションの作成

現在のバージョンの SSP（1.2.0 以降）では、Synergy 構成ツールで USBX スタックの構成がサポートされています。また、以前のバージョンの SSP で定義されたいくつかの USBX 関連コンポーネントが [DEPRECATED] コンポーネントとして取り扱われています。これらのモジュールは、下位互換性を維持するために、現在のバージョンの SSP で保持されています。新しいバージョンの SSP を使用しており、既存のアプリケーションコードと既存の SSP バージョンとの互換性を維持したい場合には、[DEPRECATED] コンポーネントを使用できます。

e^2 studio でプロジェクトの作成と設定を行い、ドライバーを追加します。

1) プロジェクトを作成します (Creating a Project を参照)。
2) プロジェクトを設定します (Configuring a Project を参照)。
3) 必要に応じて、以下のコンポーネントを追加します。追加の方法は、既存のバージョンの SSP を使用していたときと同じです。

スレッドへのドライバーの追加およびドライバーの構成

Resource	ISDE Tab	Stacks Selection Sequence
Express Logic USBX Device Class CDC-ACM(option)	Threads	Framework > USB > [DEPRECATED] USBX Device Class CDC-ACM on ux_device_class_cdc_acm
Express Logic USBX (option)	Threads	Framework > USB > [DEPRECATED] USBX on ux

注 : 新規の開発では、[DEPRECATED] とマークされているコンポーネントは使用しないでください。これらのコンポーネントを使用するのは、以前の SSP リリースで開発した既存のユーザーアプリケーションのみにしてください。

USB クラススタック構成の概要

次の図は、USBX デバイスクラススタックのインタフェース図を示しています。このスタックの構成は、最上部が USBX デバイスクラスのコンポーネント（ux_device_class_xxx）のいずれか 1 つ、中間部が USBX（ux）、デバイスクラススタックの最下部に USBX ポートドライバー（sf_el_ux デバイスコントローラドライバー（DCD））となってい

ます。SSP 転送モジュール（r_dmac）は、Synergy USB ペリフェラル（USBHS または USBFS）のハードウェア FIFO とメモリ間のデータ転送をサポートしています（推奨オプション）。USB デバイススタック構成をサポートするために、USBX デバイス構成および USBX インタフェース構成という名前のコンポーネントが用意されています。これらの 2 つのコンポーネントは、SSP の実際のソフトウェアモジュールではなく、コード生成を処理する仮想モジュールです。

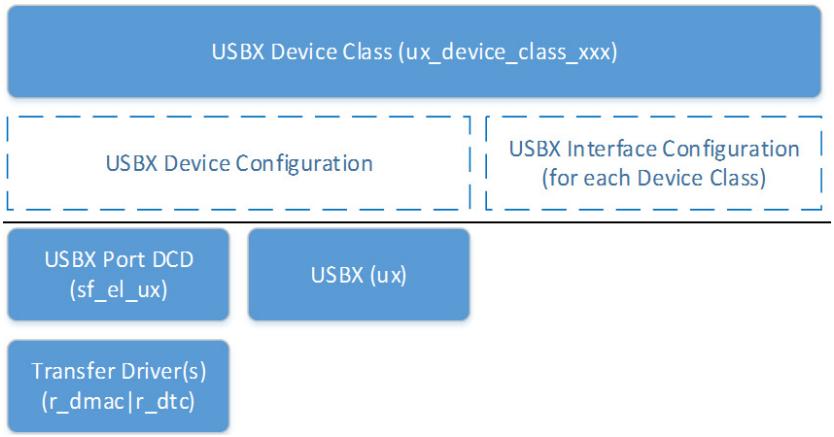


図 408:USBX デバイスクラススタック構成

注 : デバイス側ドライバーでは現在、r_dtc はサポートされていません（r_dmac のみがサポート対象です）。

次の図は、USBX ホストクラススタックのインタフェース図を示しています。このスタックの構成は、最上部が USBX ホストクラスのコンポーネント（ux_host_class_xxx）のいずれか 1 つ、中間部が USBX（ux）、ホストクラススタックの最下部に USBX ポートドライバーコンポーネント（sf_el_ux ホストコントローラドライバー（HCD））となっています。SSP 転送モジュール（r_dmac または r_dtc）は、Synergy USB ペリフェラル（USBHS または USBFS）のハードウェア FIFO とメモリ間のデータ転送をサポートしています（推奨オプション）。USB ホストスタック構成をサポートするために、仮想コンポーネント（USBX ホスト構成）というコード生成を処理する仮想コンポーネントがあります。

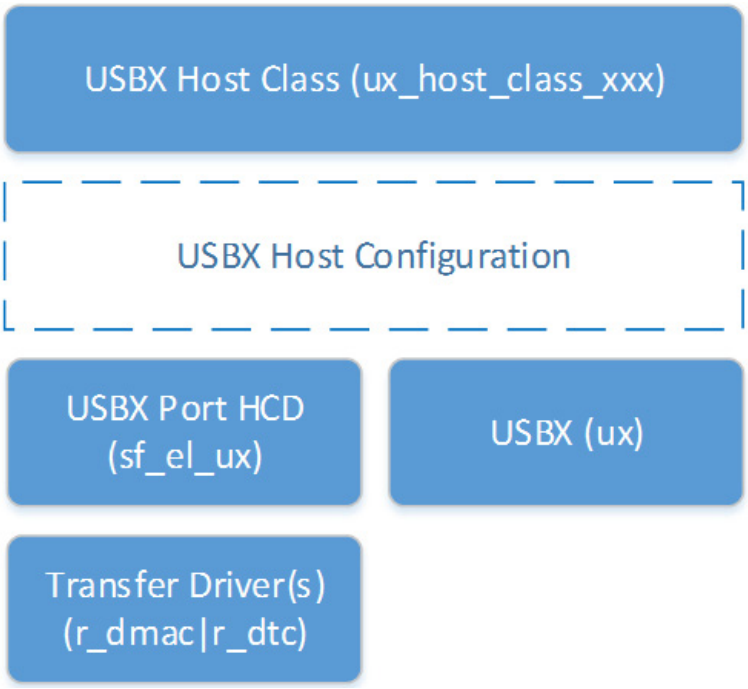


図 409:USBX ホストクラススタック構成

USB デバイス記述子の構成

USBX デバイス構成コンポーネントには、次の表に示すように、USB デバイス記述子と USB 構成記述子を自動的に生成するための構成項目があります。USB　デバイス記述子と　USB　構成記述子の仕様の詳細については、http://www.usb.org を参照し、USB 2.0 の仕様をダウンロードしてください。

USBX デバイス構成

Configuration	Settings	Description
Vendor ID	16-bit arbitrary number (Default: 0x045B)	Specify Vendor ID assigned by USB-IF. This configuration is a part of the USB Device Descriptor (**idVendor**).
Product ID	16-bit arbitrary number (Default: 0x0000)	Specify Product ID assigned by manufacturer. This configuration is a part of the Device Descriptor (**idProduct**).

Configuration	Settings	Description
Device Release Number	16-bit arbitrary number (Default: 0x0000)	Specify Device Release Number in binary-coded decimal. This configuration is a part of the USB Device Descriptor (**bcdDevice**).
Index of Manufacturer String Descriptor	Arbitrary number from 0 to 255 (Default: 0x00)	Specify the Index of Manufacturer String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (**iManufacturer**). Set zero if String Descriptor is not used. See section USBX-String-Framework-Configuration for more information.
Index of Product String Descriptor	Arbitrary number from 0 to 255 (Default: 0x00)	Specify the Index of Product String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (**iProduct**). Set zero if String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Index of Serial Number String Descriptor	Arbitrary number from 0 to 255 (Default: 0x00)	Specify the Index of Serial Number String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (**iSerialNumber**). Set zero if the String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Class Code	Device(0x00), Communications(CDC) (0x02, Default), HID (0x03), Mass Storage (0x08), Miscellaneous (0xEF), Vendor Specific (0xFF)	Select the USB Device Class Code. This configuration is a part of the USB Configuration Descriptor (**bDeviceClass**).

Configuration	Settings	Description
Index of String Descriptor describing this configuration	Arbitrary number from 0 to 255 (Default: 0x00)	Specify the Index of String Descriptor describing this configuration. This configuration is a part of the USB Configuration Descriptor (**iConfiguration**). Set zero if String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Size of USB Descriptor in bytes for this configuration	0x00 (Default) or value to be set to wTotalLength (calculated by user)	Specify the size of USB Descriptor in bytes. Modify the value for Vendor-specific Class, otherwise you can set zero to calculate the size automatically in the auto-generated code from Synergy Configuration tool. This configuration is a part of the USB Configuration Descriptor (**wTotalLength**).
Number of Interfaces	0x00 (Default) or value to be set to bNumInterfaces (calculated by user).	Specify the Number of interfaces supported by this configuration. Modify the value for Vendor-specific Class, otherwise you can set zero to calculate the value automatically in the auto-generated code from Synergy Configuration tool. This configuration is a part of the USB Configuration Descriptor (**bNumInterfaces**).
Self-Powered	Enable (Default) or Disable	Enable this configuration if your USB Device is a self-powered device. This configuration is a part of the USB Configuration Descriptor (**bmAttributes** bit6)

Configuration	Settings	Description
Remote Wakeup	Enable or Disable (Default)	Enable this configuration if your USB Device supports remote wakeup. This configuration is a part of the USB Configuration Descriptor (**bmAttributes** bit5)
Maximum Power Consumption in 2mA units(0-250)	50 (Default), Integer value from 0 to 250.	Set the maximum power consumption of your device to indicate the amount of bus power required. This configuration is 2mA units, thus, the maximum 500 mA can be specified. This configuration is a part of the USB Configuration Descriptor (**bMaxPower**).
Supported Language Code	16-bit number assigned by Manufacturer (Default: 0x0409)	Specify the Language ID Code. For example, 0x0409 English - United States. This configuration is used for Language ID Framework code generation. See section "USBX Language Framework Configuration" for more information.
Name of USBX String Framework	Arbitrary C language symbol (Default : NULL)	Specify the name of user defined USBX String Framework. This must be a valid C symbol. Set NULL if the String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Total index number in USB String Descriptor in USBX String Framework	Arbitrary integer number (Default : 0)	Specify the total number of index for String Descriptor. See section "USBX String Framework Configuration" for more information.

Configuration	Settings	Description
Name of USB Language Descriptor	Arbitrary C language symbol (Default : NULL)	Specify the name of user defined USBX Language Framework. This must be a valid C symbol. If '0' is set to the property "Total Number of Language Support", this configuration is ignored. See section "USBX Language Framework Configuration" for more information.
Total Number of Language Support	Arbitrary integer number (Default : 0)	Specify the total number of languages to support. See section "USBX String Framework Configuration" for more information. If '0' is set here, US English (0x0409) is applied as the default language.

USBX 文字列フレームワークは、人間が読むことのできる文字列で USB デバイス情報を USB ホストデバイスに供給するバイトストリームデータです。ユーザーは、必要に応じて、アプリケーションコードでバイトストリームデータを定義する必要があります。USBX 文字列フレームワークの詳細については、『USBX Device Class User Guide』の「*デバイスフレームワークの文字列の定義*」を参照してください。以下は、3 つのインデックス文字列記述子で構成される USBX 文字列フレームワークの例です。

```
const UCHAR g_usb_string_framework[] =

{

/* Index #1 (this example shows manufacturer information) */
(uint8_t) (0x0409),     /* Byte0 Language Code, US English */
(uint8_t) (0x0409 >> 8),    /* Byte1 Language Code */
0x01,               /* Byte2 Index */
7,              /* Byte3 Length */
'R', 'E', 'N', 'E', 'S', 'A', 'S',

/* Index #2 (this example shows product information) */
(uint8_t) (0x0409),     /* Byte0 Language Code, US English */
(uint8_t) (0x0409 >> 8),    /* Byte1 Language Code */
0x02,               /* Byte2 Index */
10,                 /* Byte3 Length */
'C', 'O', 'M', ' ', 'D', 'E', 'V', 'I', 'C', 'E',

/* Index #3 (this example shows Device Serial Number information) */
(uint8_t) (0x0409),     /* Byte0 Language Code, US English */
(uint8_t) (0x0409 >> 8),    /* Byte1 Language Code */
0x03,               /* Byte2 Index */
4,              /* Byte3 Length */
```

```
'0', '1', '0', '0'
};
```

USBX デバイス構成コンポーネントのプロパティは、次の表のように構成できます。各構成の詳細については、この表を参照してください。

USBX 文字列フレームワーク構成の例

Configurations for String Descriptor on Synergy Configuration tool	Setting Example
Name of USB String Framework	g_usb_string_framework
Total index number of USB String Descriptor in USBX String Framework	3 (3 indexes)
Index of Manufacturer String Descriptor	1 (Index #1)
Index of Product String Descriptor	2 (Index #2)
Index of Serial Number String Descriptor	3 (Index #3)
Index of String Descriptor describing this configuration	0 (no string information)
Index of String Descriptor Describing Communications Class interface	0 (no string information)

USBX 文字列フレームワークは、複数の言語をサポートするバイトストリームデータです。必要に応じて、アプリケーションコードでバイトストリームデータを定義する必要があります。詳細については、『USBX Device Class User Guide』の「各文字列に対してデバイスがサポートする言語の定義」* を参照してください。以下は USBX 言語フレームワークの例で、2 言語をサポートしています。

```
const UCHAR g_usb_language_framework[] =
{
    /* US English */
    (uint8_t) (0x0409),
    (uint8_t) (0x0409 >> 8),
    /* Japanese */
    (uint8_t) (0x0411),
    (uint8_t) (0x0411 >> 8),
};
```

USBX デバイス構成コンポーネントのプロパティは、次の表のように構成できます。詳細については、「USBX デバイスの構成」を参照してください。

USBX 言語フレームワーク構成の例

Configurations for String Descriptor on Synergy Configuration tool	Setting Example
Name of USB Language Descriptor	g_usb_language_framework
Total Number of Language Support	2 (2 languages)

デバイスオンリーのサイズ最適化

Express Logic 社の USBX モジュールは、Synergy S1 パーツのコードサイズを小さくするためにデバイスオンリーモードでビルドされます。Synergy 構成ツールの [BSP] タブで S1 ボードを選択した場合、自動的にこの構成（UX_SYSTEM_DEVICE_ONLY）が適用されます。次の構成は、デバイスオンリーモードに固定されることに注意してください。

- UX_THREAD_STACK_SIZE=512
- UX_SLAVE_REQUEST_DATA_MAX_LENGTH=512

4.3.35.3 USBX の自動生成コードの手順

注:このセクションのコード例は、変更される場合があります。内容および今後の変更に関する保証に注意してください。

USBX デバイススタック自動生成コードの手順

次のコード例は、USBX デバイスクラスの擬似コードです。1 つの（または複数の）USBX デバイスクラス component(s) を Synergy 構成ツールに追加すると、関数コールシーケンスが Synergy 構成ツールから自動生成されます。自動生成されたコードは、common_data.c ファイルに対して定義され、以下の機能を提供します。

- USB デバイス記述子を生成します。
- メモリプール内の USBX ソフトウェアコンテキストを初期化します。
- Synergy 構成ツールで追加した USBX デバイススタックを初期化します。
- Synergy USB controller(s) をデバイスモードで初期化します。USBX デバイススタックは、複数のデバイスクラスと 1 つのデバイスコントローラとの間のトポロジとして「多対 1」をサポートします。たとえば、USB 複合デバイスで構成される 2 つの USBX デバイスクラスは同一の USB コントローラを使用できます。

注 :USBX デバイススタックは、複数の USB コントローラの同時使用をサポートしていません。Synergy パーツに複数の USB コントローラがある（S7G2 パーツなど）場合でも、一度に使用されるのは、いずれか 1 つの USB コントローラのみとなります。

```
// Interrupt vector registering for USBHS or USBFS controller.
SSP_VECTOR_DEFINE_UNIT();

void g_common_init(void)
{
    // Initialize USBX Memory.
    ux_system_initialize ();

    // Initialize USBX Device stack.
    ux_device_stack_initialize ();
```

```
    // Register the Device CDC-ACM Class if the class is used.
    ux_device_stack_class_register();

    // Register the Device HID Class if the class is used.
    ux_device_stack_class_register();

    // Register the Device Mass Storage Class if the class is used.
    ux_device_stack_class_register();

    // Initialize the USB Device Controller. This function calls either of
    // _ux_dcd_synergy_initalize() or
    // _ux_dcd_synergy_initalize_transfer_support()
    ux_dcd_initialize ();
}
```

USBX ホストスタック自動生成コードの手順

次のコード例は、USBX ホストクラスの擬似コードです。1 つの（または複数の）USBX ホストクラス component(s) を Synergy 構成ツールに追加すると、関数コールシーケンスが Synergy 構成ツールから自動生成されます。自動生成されたコードは、common_data.c ファイルに対して定義され、以下の機能を提供します。

- メモリプール内の USBX ソフトウェアコンテキストを初期化します。
- Synergy 構成ツールで追加した USBX ホストスタックを初期化します。ホストクラススタックのマルチインスタンスをビルドできます。
- Synergy 構成ツールで USBX ホスト HID クラスを追加した場合に、USBX HID クライアントを初期化します。
- Synergy USB controller(s) をホストモードで初期化します。USBX ホストスタックは、デバイスクラスとホストコントローラとの間のトポロジとして " 多対 1" と " 多対多 " をサポートします。たとえば、2 つの USBX ホストクラスが同一の USB コントローラを使用できます。また、Synergy パーツに複数の USB コントローラがある（S7G2 パーツなど）場合には、2 つの USBX ホストクラスがそれぞれ別の USB コントローラを使用することもできます。
- ユーザーのアプリケーションのために USBX ホストクラスコンテナを取得します。

```
// Interrupt vector registering for USBHS or USBFS controller.
SSP_VECTOR_DEFINE_UNIT();

void g_common_init(void)
{
    // Initialize FileX (ONLY for Mass Storage Class)
    fx_system_initialize ();

    // Initialize USBX Memory.
    ux_system_initialize ();

    // Initialize the USBX Host stack.
    ux_host_stack_initialize ();

    // Register the HUB Class if the class is used.
    ux_host_stack_class_register();
```

```
    // Register the Host CDC-ACM Class if the class is used.
    ux_host_stack_class_register();

    // Register the Host HID Class if the class is used.
    ux_host_stack_class_register();

    // Register the Host HID Clents if the HID clients are used.
    ux_host_class_hid_clients_register ();

    // Register the Host Mass Storage Class if the class is used.
    ux_host_stack_class_register();

    // Register and initialize the USB Host Controller.
    ux_host_stack_hcd_register ();

    // Get the USBX Host Class Container for registered classes.
    ux_host_stack_class_get ();
}
```

4.3.35.4 USBX のアプリケーションコードの例

各 USBX モジュール用のモジュールガイドで、アプリケーションプロジェクトを利用できます。一般的なユースケースを対象とした作業コードを示すアプリケーションプロジェクトについては、モジュールのガイドを参照してください。

4.3.35.5 USBX の特殊リンカセクション

USBX デバイススタック構成は、リンカスクリプトファイルで以下の特殊なメモリセクションを使用します。リンカスクリプトのメモリセクションの順序は、USB デバイス記述子のバイトストリームで構成する必要があります。このバイトストリームは、_ux_device_stack_initialize() 関数に渡されるため、リンカスクリプトの定義を変更することはできません。

USBX デバイス記述子のメモリセクション

Memory section	USB Descriptor to be defined in the section
.usb_device_desc_fs*	The USB Device Descriptor for FS mode
.usb_config_desc_fs*	The USB Configuration Descriptor for FS mode
.usb_interface_desc_fs*	The USB Interface Descriptor for FS mode
.usb_device_desc_hs*	The USB Device Descriptor for HS mode
.usb_config_desc_hs*	The USB Configuration Descriptor for HS mode

Memory section	USB Descriptor to be defined in the section
.usb_interface_desc_hs*	The USB Interface Descriptor for HS mode

注 :HS モードのメモリセクションは、USBHS コントローラがある Synergy パーツ専用です。

4.3.35.6 USBX のメモリ要件

USBX デバイススタックおよび USBX ホストスタックは、制御ブロックの RAM を消費します。Synergy 構成ツールは、自動生成コードでメモリを USBX メモリプールに静的に割り当てます。メモリの消費は、各クラスによって異なります。特定のモジュールの USBX メモリ要件については、『SSP ユーザーズマニュアル』で該当するモジュールの概要セクションを参照してください。

4.3.35.7 USBX の制限事項

- USB デバイスベンダー固有のクラスに対するサポートは利用できません。
- このモジュールでは、USB コントローラの割り込みが有効になっている必要があります。詳細については、Logic USBX Synergy ポートフレームワークモジュールの制限事項を参照してください。
- USBX のクラス（ux_device_class_cdc_ecm,、ux_device_class_rndis,、ux_host_class_audio、ux_host_class_gser,、ux_host_class_printer、ux_host_class_prolific、ux_host_class_swar）および USBX のネットワークドライバー（ux_network_driver）は実験的なモジュールであり、このバージョンの SSP の Synergy パーツに対するテストは行われていません。そのため、これらのモジュールを製品開発に使用することはお勧めしません。

4.3.36 USBX ソースコンポーネントモジュール

4.3.36.1 USBX ソースコンポーネントモジュールの概要

このドキュメントの目的は、e^2 studio の USBX ソースコンポーネントに関する簡単なリファレンスを提供することです。プロパティについて、各プロパティで提供されるフッターコメントよりもかなり詳細に説明しています。デフォルト値の変更がどのような場合に必要であるかについて、コンテキスト固有の使用方法が記載されています。このドキュメントを使えば、Express Logic 社の『USBX User Guide』を参照することなく、より容易に USBX ソースコンポーネントを使用でき、開発者は USBX の機能をさらに素早く習得することができます。

4.3.36.2 どのような場合に USBX ソースコンポーネントを含めるか

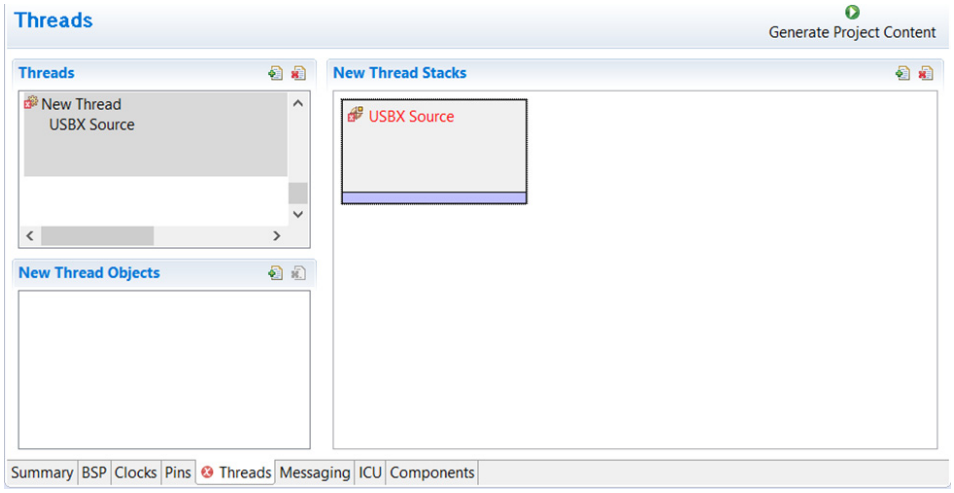
Synergy コンフィギュレータ環境を使用する開発者は、USBX ソースコンポーネントを追加することで、USBX ライブラリのカスタマイズ、デフォルト設定からの値の変更、特定の機能の有効化または無効化を行うことができるようになります。USBX ソースコンポーネントを追加しない場合は、ビルド済みの USBX ライブラリを使用する必要があります。非常にシンプルなものを除くほとんどのプロジェクトでは、通常、開発者が USBX 環境をカスタマイズする必要があります。ThreadX ソースコンポーネントは、よりハイレベルなソースコンポーネント（FileX、NetX、NetX Duo、GUIX、USBX など）を追加するときに自動的に追加されることに留意してください。

USBX ソースコンポーネントを追加しない場合、e^2 studio コンフィギュレータは、USBX のデフォルト設定を使用して事前にビルドされたライブラリを使用します。

4.3.36.3 USBX ソースコンポーネントの追加

e2 studio コンフィギュレータで、[Threads] リストから任意のスレッドを選択し、[New Stack] ボタンを押してメニューから [X-Ware] > [USBX] > [Common] > [USBX Source]

を選択することで、USBX ソースコンポーネントを追加します。



4.3.36.4 USBX ソースコンポーネントプロパティの変更

USBX プロパティの設定を変更した後、開発者は [Generate Project Content] ボタンをクリックして、e2studio のプロジェクトコンフィギュレータを更新し、USBX ライブラリを再ビルド（つまり、プロジェクトを再ビルド）する必要があります。プロジェクトを再ビルドせず単にプロパティを変更（またはプリプロセッサリストで #define を適用）しても、e2 studio は以前にビルドされたライブラリを使用するため、いずれの変更にも影響しません。

デフォルト設定は経験に基づいており、ほとんどの一般的なユースケースに該当する設定です。

4.3.36.5 USBX ソースコンポーネントの概要

USBX ソースコンポーネントのプロパティについて、Synergy コンフィギュレータの [Properties] ウィンドウに表示される順序で説明します。

Ticks per second for USBX system – デフォルト値は非表示、ThreadX の値が使用される – デフォルトでは、USBX は ThreadX で定義された値を使用します。ThreadX のティックタイマメカニズムを大幅に変更していない限り、この値を変更してはなりません。

Maximum Classes – デフォルト値は非表示、8 が使用される – 定義されると、この値が USBX でロードできるクラスの最大数になります。この値が表すのはクラスコンテナであり、クラスのインスタンス数ではありません。たとえば、特定の USBX の実装がハブクラス、プリンタクラス、ストレージクラスを必要とする場合は、これらに所属するデバイスの数にかかわらず、UX_MAX_CLASSES の値を 3 に設定します。

Maximum Slave Classes – デフォルト値は非表示、3 が使用される – 定義されると、この値が USBX でロードできるデバイススタック内のクラスの最大数になります。

Maximum Slave Interfaces - デフォルト値は非表示、16 が使用される – 定義されると、この値がデバイスフレームワーク内にあるインタフェースの最大数になります。

Maximum Host Class Containers - デフォルト値は非表示、最大値は設定なし。

Maximum Device Class Containers - デフォルト値は非表示、最大値は設定なし。

Maximum Host Controllers - デフォルト値は非表示、最大値は設定なし - この値は、システムで使用できる各種のホストコントローラの数を表します。USB 1.1 をサポートする場合、たいていの場合はこの値を 1 にします。USB 2.0 をサポートする場合は、この値を 1 よりも大きくすることができます。この値は同時に実行できるホストコントローラの数を表します。たとえば、2 つの OHCI インスタンスを実行するか、EHCI コントローラと OHCI コントローラをそれぞれ 1 つずつ実行する場合、UX_MAX_HCD は 2 に設定する必要があります。

Maximum Devices – デフォルト値は非表示、8 が使用される – この値は、USB にアタッチできるデバイスの最大数を表します。通常、1 つの USB にアタッチできるデバイスの理論上の最大数は 127 になります。この値はメモリを温存するために小さくすることができます。この値はデバイスの合計数を表しており、システム内の USB バスの数とは無関係であることに留意してください。

Maximum EDs – デフォルト値は非表示、80 が使用される – この値は、コントローラプール内の ED の最大数を表します。この数は、1 つのコントローラのみに割り当てられます。コントローラの複数のインスタンスが存在する場合、この値は各コントローラに対して使用されます。

Maximum TDs – デフォルト値は非表示、128 が使用される – この値は、コントローラプール内の通常の TD の最大数を表します。この数は、1 つのコントローラのみに割り当てられます。コントローラの複数のインスタンスが存在する場合、この値は各コントローラに対して使用されます。

Maximum Isochronous TDs - デフォルト値は非表示、128 が使用される – この値は、コントローラプール内のアイソクロナス TD の最大数を表します。この数は、1 つのコントローラのみに割り当てられます。コントローラの複数のインスタンスが存在する場合、この値は各コントローラに対して使用されます。

Stack size for USBX threads - デフォルト値は非表示。ホストおよび混在したコントローラでは 1024 バイト、デバイスのみのコントローラでは 512 が使用される – この値は USBX スレッド内のスタックサイズ（バイト単位）です。一般に、この値は使用されるプロセッサおよびホストコントローラに応じて 1024 または 2048 バイトにします。

USBX Enumeration Thread Priority - デフォルト値は非表示。20 が使用される – これはバストポロジを監視する USBX 列挙スレッドの ThreadX におけるプライオリティの値です。

USBX Standard Thread Priority - デフォルト値は非表示。20 が使用される – これは標準 USBX スレッドの ThreadX におけるプライオリティの値です。

USBX HID Keyboard Class Priority - デフォルト値は非表示。20 が使用される – これは USBX HID キーボードクラスの ThreadX におけるプライオリティの値です。

USBX HCD Thread Priority - デフォルト値は非表示。2 が使用される – これはホストコントローラスレッドの ThreadX におけるプライオリティの値です。

No use of time slice - デフォルト値は無効 – 有効化されている場合、ThreadX のターゲットポートでタイムスライシングが使用されません。

Maximum Slave Logical Units - デフォルト値は非表示。2 が使用される – この値は、デバイスストレージクラスドライバー内で示される SCSI 論理ユニットの現在の数を表します。

Maximum Host Logical Units - デフォルト値は非表示。16 が使用される – この値は、ホストストレージクラスドライバー内で示される SCSI 論理ユニットの最大数を表します。

Slave Request Control Maximum Length - デフォルト値は非表示。256 が使用される – この値は、デバイススタックの制御エンドポイントで受信する最大バイト数を表します。デフォルトは 256 バイトですが、メモリ制約環境ではこれよりも小さくすることができます。

Slave Request Data Maximum Length - デフォルト値は非表示。デバイスのみのコントローラでは 512 が、それ以外の場合は 4096 が使用される – この値は、デバイススタックのバルクエンドポイントで受信する最大バイト数を表します。デフォルトは 4096 バイトですが、メモリ制約環境ではこれよりも小さくすることができます。

Enforce Safe Alignment - デフォルト値は無効 – 有効化されると、メモリ割り当てスキームにより配列が強制されます。デフォルトの配列値は UX_SAFE_ALIGN.

です。

Show linkage warning - デフォルト値は有効 – デフォルトでは、リンキングの警告を表示します。

4.3.37 USBX Synergy ポートフレームワーク

Express Logic 社の USBX Synergy ポートフレームワークモジュール（sf_el_ux）は、SSP に統合され、Express Logic 社のUSBXで使用されます。USBXの詳細(APIリファレンスなど)については、『USBX User Guide』を参照してください。

4.3.37.1 USBX Synergy ポートフレームワークモジュールの特長

Express Logic 社の USBX Synergy ポートフレームワークモジュールは以下の機能に対応します。

- Express Logic USBX を SSP サポート USBX API に実装します
- USBHS 周辺機能用のポートデバイスコントローラドライバー（DCD）をサポートします
- USBFS 周辺機能用のポートデバイスコントローラドライバー（DCD）をサポートします
- USBHS 周辺機能用のポートホストコントローラドライバー（HCD）をサポートします
- USBFS 周辺機能用のポートホストコントローラドライバー（HCD）をサポートします
- 転送モジュールの動作をサポートします（オプション）

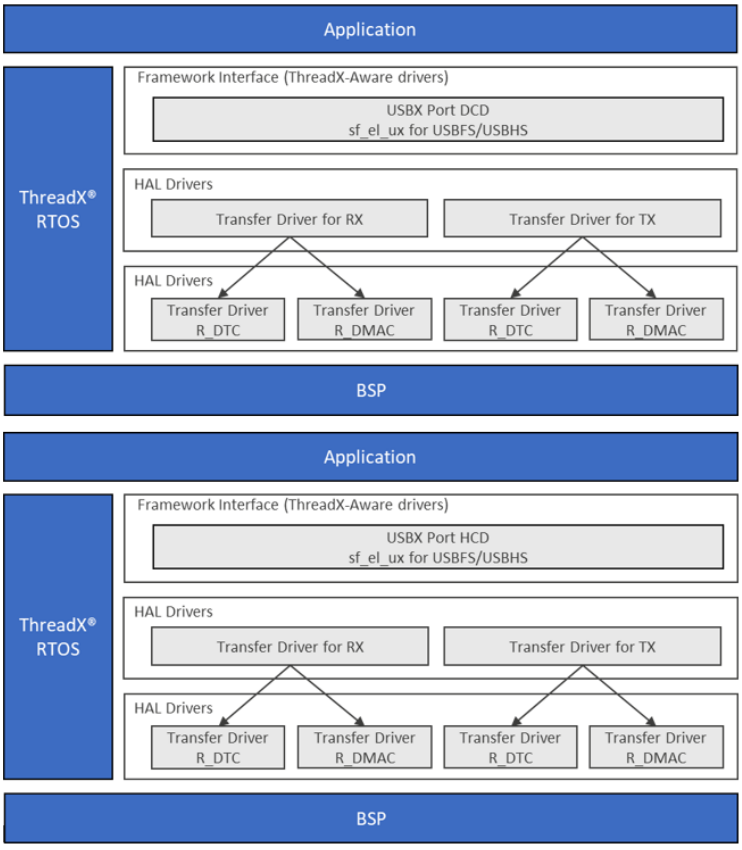


図 410:USBX Synergy ポートフレームワークモジュールのブロック図

4.3.37.2 USBX Synergy ポートフレームワークモジュール API の概要

Express Logic USBX Synergy ポートフレームワークモジュール自体は API 呼び出しを持たず、Express Logic USBX API 呼び出しの API 呼び出しを実装しています。これらの API に関するドキュメントについては、『Express Logic USBX User Manual』で利用できます。

4.3.37.3 USBX Synergy ポートフレームワークモジュールの動作の概要

Express Logic 社の USBX Synergy ポートフレームワークモジュールは、Synergy ハードウェアの USBX スタックを使用するために必要な Synergy USB ハードウェアポート機能を提供します。このモジュールを使用したアプリケーションコードは、USBX API コールの使用を想定したものとなっています。

USBX Synergy ポートフレームワークモジュールの動作に関する重要な注意事項と制限事項

Express Logic USBX Synergy ポートフレームワークモジュールには、SSP における Express Logic USBX API のサポートが含まれます。使用可能な API の詳細な説明については、『Express Logic USBX User Manual』を参照してください。

Express Logic 社の USBX Synergy ポートフレームワークモジュールは、USBHS および USBFS ペリフェラルのポートデバイスコントローラドライバー（DCD）と、USBHS および USBFS ペリフェラルのポートホストコントローラドライバー（HCD）をサポートします。

ユーザーは、USBX Synergy ポートフレームワークモジュールの転送モジュールを使用して、ブロック転送モードでデータを転送することにより、USB のデータスループットを向上させることができます。転送モジュールを有効にするには、Synergy 構成ツールで 2 つの転送コンポーネントインスタンスを USBX クラススタックに追加し、プロパティで割り込みを有効にするだけです。Synergy 構成ツールは、ドライバーセットアップコードを自動生成して、common_data.c. での DMAC 転送または DTC 転送を有効にします。

注 : このモジュールは、USB コントローラの割り込みを使用します。Synergy 構成ツールで適切な割り込み優先順位レベルを設定してください。設定しないとモジュールが機能しません。このモジュールが使用されると、転送モジュール（DMAC または DTC として実装）の割り込みが使用されます。Synergy 構成ツールで適切なプライオリティレベルを設定してください。転送モジュールのプライオリティレベルは USB コントローラよりも高くします。そのようにしないとモジュールが機能しません。

- Synergy USB コントローラ（USBHS および USBFS）には、アイソクロナス転送タイプ（PIPE1 および PIPE2）に使用できるパイプが一定数備わっています。これは、UVC HOST として構成される Synergy ボードに接続できる UVC デバイス（2 つのデバイス）の数を制限するものです。
- デバイス側のドライバー（sf_el_ux DCD ドライバー）は転送インタフェースとしての DTC をサポートしません。
- アイソクロナス転送は、USB ホストに限りサポートされます。USB デバイスについては、この転送タイプをサポートしていません。
- USBFS コントローラは、一般的な UVC デバイスをサポートしない可能性があります。USBFS コントローラ上のアイソクロナスパイプの最大パケットサイズは256バイトに制限されています。これはUVCの使用に影響します。
- Synergy USB コントローラ（USBHS および USBFS）は高帯域幅のアイソクロナス転送をサポートしていません。このコントローラは高速で実行されている場合、マイクロフレームごとに 1 トランザクションをサポートします。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.37.4 アプリケーションへの USBX Synergy ポートフレームワークモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに USBX Synergy ポートフレームワークモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の始めの方にあるいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

USBX Synergyポートフレームワークモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

USBX Synergy ポートフレームワークモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_el_ux_hcd_hs_0 USBX Port HCD on sf_el_ux for USBHS	Threads	New Stack> X-Ware> USBX> Host > Synergy Port> USBX Port HCD on sf_el_ux for USBHS
g_sf_el_ux_hcd_fs_0 USBX Port HCD on sf_el_ux for USBFS	Threads	New Stack> X-Ware> USBX> Host > Synergy Port> USBX Port HCD on sf_el_ux for USBFS
g_sf_el_ux_dcd_hs_0 USBX Port HCD on sf_el_ux for USBHS	Threads	New Stack> X-Ware> USBX> Device > Synergy Port> USBX Port HCD on sf_el_ux for USBHS
g_sf_el_ux_dcd_fs_0 USBX Port HCD on sf_el_ux for USBFS	Threads	New Stack> X-Ware> USBX> Device > Synergy Port> USBX Port HCD on sf_el_ux for USBFS

次の図に示すように、USBX Synergy ポートフレームワークモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

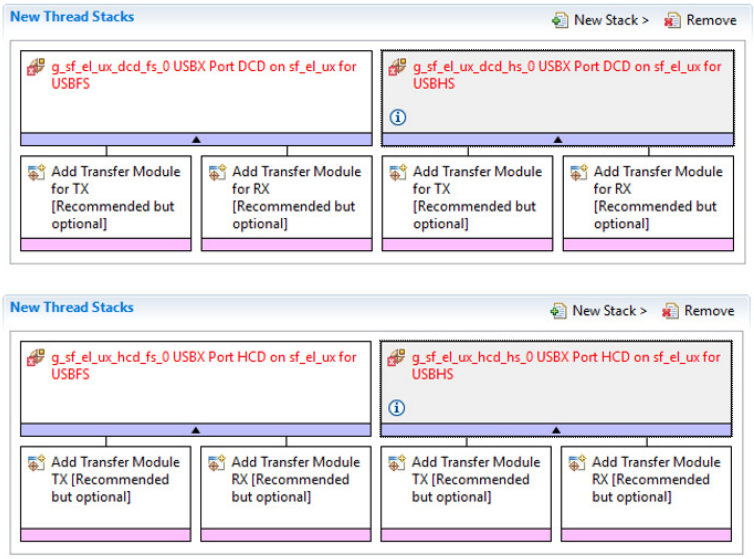


図 411:**USBX Synergy ポートフレームワークモジュールのスタック**

4.3.37.5 USBX Synergy ポートフレームワークモジュールの構成

ユーザーは必要な動作に合わせて、USBX Synergy ポートフレームワークモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

USBFS の sf_el_ux の USBX ポート DCD に対する構成設定

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for full-speed USB.

ISDE Property	Value	Description
LDO Regulator (Only for S3 and S1 part MCUs)	Enable, Disable Default: Disable	Select the LDO regulator will be enabled.
Name	g_sf_el_ux_dcd_fs_0	Module name.
USB Controller Selection	USBFS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBHS の sf_el_ux 上の USBX ポート DCD の構成設定

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for high speed USB.
Name	g_sf_el_ux_dcd_hs_0	Module name.
USB Controller Selection	USBHS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBFS の sf_el_ux 上の USBX ポート HCD の構成設定

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for full-speed USB.

ISDE Property	Value	Description
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	Select the VBUSEN pin signal logic.
LDO Regulator (Only for S3 and S1 part MCUs)	Enable, Disable Default: Disable	Select the LDO regulator will be enabled.
Name	g_sf_el_ux_hcd_fs_0	Module name.
USB Controller Selection	USBFS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBHS の sf_el_ux 上の USBX ポート HCD の構成設定

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for high speed USB.
FIFO size for Bulk/Isochronous Pipes	512, 1024, 1536, 2048 bytes Default: 512 bytes	Select the FIFO size for bulk and isochronous transfers.
Number of Isochronous Pipes Reserved	0,1,2 Default: 0	Select the number of isochronous pipes to reserve.
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	Select the VBUSEN pin signal logic.

ISDE Property	Value	Description
Enable High Speed	Enable, Disable Default: Enable	Select if high speed should be enabled.
Name	g_sf_el_ux_hcd_hs_0	Module name.
USB Controller Selection	USBHS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。注 : ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

USBX Synergy ポートフレームワークのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

r_dmac 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Name	g_transfer0	Module name.
Channel	0	Specify the hardware channel.
Mode	Block	Select the transfer mode.
Transfer Size	1 Byte	Select the transfer size.
Destination Address Mode	Fixed	Select the address mode for the destination.
Source Address Mode	Incremented	Select the address mode for the source.

ISDE Property	Value	Description
Repeat Area (Unused in Normal Mode	Source	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.
Number of Transfers	0	Specify the number of transfers.
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source	Device: Event USBFS FIFO 0 Host: Software Activation	Select the DMAC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback	NULL	A user callback that is called at the end of the transfer.
Transfer End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Software Start	Enabled, Disabled Default: Disabled	Include code for software start in the build.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Section to place the DTC vector table.
Name	g_transfer0	Module name.
Mode	Block	Specify the hardware channel.
Transfer Size	1 Byte	Select the transfer mode.
Destination Address Mode	Fixed	Select the transfer size.
Source Address Mode	Incremented	Select the address mode for the destination.
Repeat Area (Unused in Normal Mode	Source	Select the address mode for the source.
Interrupt Frequency	After all transfers have completed	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.
Number of Transfers	0	Specify the number of transfers.

ISDE Property	Value	Description
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source (Must enable IRQ)	Device: Event USBFS FIFO 0 Host: Software Activation 1	Select the DTC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback (Only valid with Software start)	NULL	A user callback that is called at the end of the transfer.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dmac 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Name	g_transfer0	Module name.
Channel	0	Specify the hardware channel.
Mode	Block	Select the transfer mode.
Transfer Size	1 Byte	Select the transfer size.

ISDE Property	Value	Description
Destination Address Mode	Incremented	Select the address mode for the destination.
Source Address Mode	Fixed	Select the address mode for the source.
Repeat Area (Unused in Normal Mode	Destination	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.
Number of Transfers	0	Specify the number of transfers.
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source	Device: Event USBFS FIFO 1 Host: Software Activation	Select the DMAC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback	NULL	A user callback that is called at the end of the transfer.
Transfer End Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

r_dtc 上の転送ドライバーの構成設定

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	If selected code for parameter checking is included in the build.
Software Start	Enabled, Disabled Default: Disabled	Include code for software start in the build.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Section to place the DTC vector table.
Name	g_transfer0	Module name.
Mode	Block	Specify the hardware channel.
Transfer Size	1 Byte	Select the transfer mode.
Destination Address Mode	Incremented	Select the transfer size.
Source Address Mode	Fixed	Select the address mode for the destination.
Repeat Area (Unused in Normal Mode	Destination	Select the address mode for the source.
Interrupt Frequency	After all transfers have completed	Select the repeat area. Either the source or destination address resets to its initial value after completing Number of Transfers in Repeat or Block mode.
Destination Pointer	NULL	Specify the transfer destination pointer.
Source Pointer	NULL	Specify the transfer source pointer.
Number of Transfers	0	Specify the number of transfers.

ISDE Property	Value	Description
Number of Blocks (Valid only in Block Mode)	0	Specify the number of blocks to transfer in Repeat or Block mode.
Activation Source (Must enable IRQ)	Device: Event USBFS FIFO 1 Host: Software Activation 2	Select the DTC transfer start event.
Auto Enable	False	Auto enable the transfer in open().
Callback (Only valid with Software start)	NULL	A user callback that is called at the end of the transfer.
ELC Software Event Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the transfer end interrupt priority.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX Synergy ポートフレームワークモジュールのクロック構成

USB ペリフェラルモジュールは UCLK をクロックソースとして使用します。UCLK は 48 MHz の動作に合わせて構成する必要があります。SSP 構成ウィンドウで、[Clocks] タブを選択してクロックソースの設定を確認します。

USBX Synergy ポートフレームワークモジュールのピン構成

USB ペリフェラルモジュールは、MCU ピンを使用して外部デバイスと通信します。I/O ピンを選択し、外部デバイスの要件に合うように構成します。次の表では [SSP Configuration] ウィンドウでのピンの選択方法を説明し、それ以降の表では USB ピンを例に使って選択プロセスを示します。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

USBFS および USBHS のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
USBFS	Pins	Select Peripherals > Connectivity: USBFS> USBFS0

Resource	ISDE Tab	Pin selection Sequence
USBHS	Pins	Select Peripherals > Connectivity: USBHS> USBHS0

注 : 選択シーケンスでは、USBFS0 または USBHS0 がドライバーに必要なハードウェアターゲットであることを想定しています。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select device as the Operation Mode
USBDP	USBDP	USBDP pin
USBDM	USBDM	USBDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	None	VBUSEN pin
VBUS	None, P407 Default: P407	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID Pin
VCCUSB	VCCUSB	VCCUSB pin
VSSUSB	VSSUSB	VSSUSB pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select Device as the Operation Mode
USBHSDP	USBHSDP	USBHSDP pin
USBHSDM	USBHSDM	USBHSDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	PB00	VBUSEN pin
VBUS	PB01	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID pin
USBHSRREF	USBHSRREF	USBHSRREF pin
AVCCUSBHS	AVCCUSBHS	AVCCUSBHS pin
AVSSUSBHS	AVSSUSBHS	AVSSUSBHS pin
PVSSUSBHS	PVSSUSBHS	PVSSUSBHS pin
VCCUSBHS	VCCUSBHS	VCCUSBHS pin
VSS1USBHS	VSS1USBHS	VSS1USBHS pin
VSS2USBHS	VSS2USBHS	VSS2USBHS pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.37.6 アプリケーションでの USBX Synergy ポートフレームワークモジュールの使用

Express Logic Synergy ポート USBX フレームワークモジュールをスレッドに追加すると、よりハイレベルなモジュールで Express Logic API を使用できるようになります。通常は、Express Logic Synergy ポート USBX フレームワーク

モジュールをアプリケーションコードの中で直接使用する必要はありません。このモジュールをスタンドアロンで使用し、関連する API に直接アクセスする必要がある場合は、『Express Logic USBX User Manual』を参照してください。

4.3.38 USBX デバイスクラス CDC-ACM

USBX ™デバイスクラス CDC-ACM モジュールは、USBX デバイスクラス CDC-ACM モジュールアプリケーション用のハイレベル API を提供し、Synergy MCU 上の USB およびデータ転送ペリフェラルを使用します。ユーザー定義のコールバックを作成し、USB CDC-ACM クラスがアクティブ化または非アクティブ化されたことを判定できます。

4.3.38.1 USBX デバイスクラス CDC-ACM モジュールの特長

USB デバイスクラス CDC-ACM モジュールにより、USB ホストシステムは、シリアルデバイスなどのデバイスと通信することが可能になります。このクラスは、USB 標準に基づいており、CDC 標準のサブセットです。USBX デバイスクラス CDC-ACM モジュールには、以下の主要な特長があります。

- USB フルスピード（USBFS）または USB ハイスピード（USBHS）のサポート
- 性能向上を目的とした受信および送信データ転送ドライバー
- リードとライトのためのハイレベル API

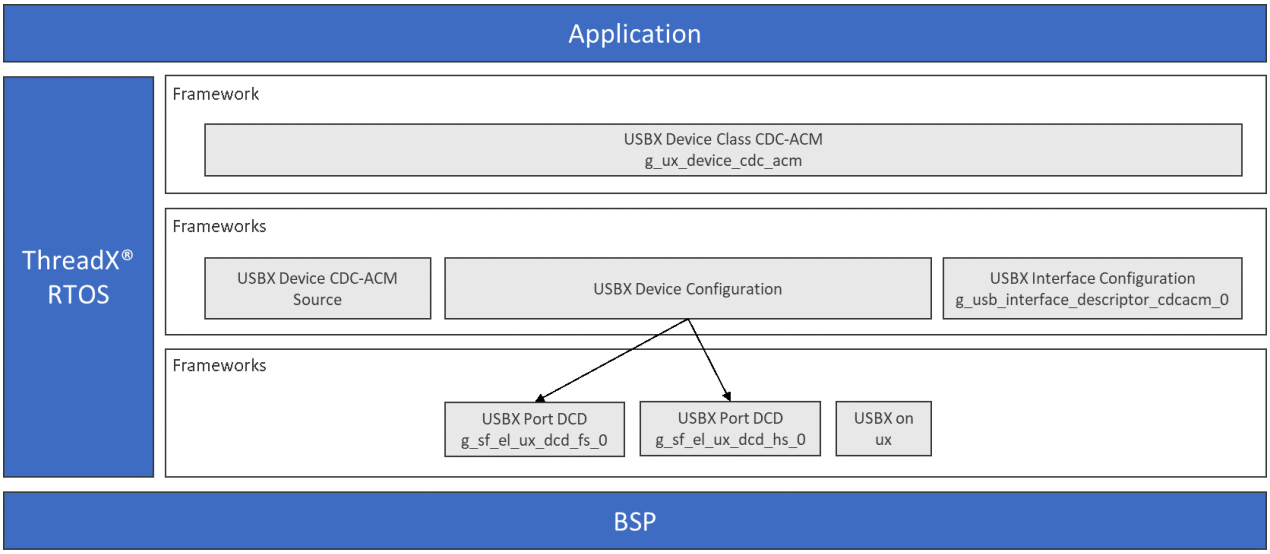


図 412:**USBX デバイスクラス CDC-ACM モジュールのブロック図**

4.3.38.2 USBX デバイスクラス CDC-ACM モジュールの API の概要

USBX デバイスクラス CDC-ACM モジュールでは、USB ペリフェラルを介したリードおよびライトのための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

USBX デバイスクラス CDC-ACM モジュールの API の要約

Function Name	Example API Call and Description
ux_device_class_cdc_acm_read	status = ux_device_class_cdc_acm_read(cdc, buffer, UX_DEMO_BUFFER_SIZE, &actual_length); This function is called when an application needs to read from the OUT data pipe (OUT from the host, IN from the device).
ux_device_class_cdc_acm_write	status = ux_device_class_cdc_acm_write(cdc, buffer, UX_DEMO_BUFFER_SIZE, &actual_length); This function is called when an application needs to write to the IN data pipe (IN from the host, OUT from the device).

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
UX_SUCCESS	This operation was successful.
UX_CONFIGURATION_HANDLE_UNKNOWN	Device is no longer in the configured state.
UX_TRANSFER_NO_ANSWER	No answer from device. The device was probably disconnected while the transfer was pending.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.3.38.3 USBX デバイスクラス CDC-ACM モジュールの動作の概要

このモジュールガイドに関連するアプリケーションプロジェクトで示されているように、CDC-ACM クラスの初期化には、特定のパラメータがいくつか必要です。

CDC-ACM は USB-IF 標準に基づいており、Mac OS® および Linux OS® では自動的に認識されます。Windows® プラットフォームでは、.inf ファイルが必要です。Express Logic® は、CDC-ACM クラス用のテンプレートを提供しています。このテンプレートは、本資料作成時、次から入手できます。

Synergy - Media Gallery: Download USBX Windows Host Files

https://renesasrulz.com/synergy/m/mediagallery/2707

このファイルは、デバイスが使用する PID/VID に合わせて変更する必要があります。PID/VID は、メーカーおよび製品が USB-IF に登録された時点で、最終ユーザーに固有の値になります。.inf ファイルの変更するフィールドについては、このモジュールガイドに関連するアプリケーションプロジェクトで説明されています。

USBX CDC-ACM デバイスのデバイスフレームワークでは、デバイス記述子に PID/VID が格納されています（デバイス記述子は、前の段落で言及されているアプリケーションプロジェクトで宣言されています）。

USB ホストシステムは、USBX CDC-ACM デバイスを検出するとシリアルクラスをマウントするため、デバイスを任意のシリアルターミナルプログラムで使用できるようになります（ホストのオペレーティングシステムを参照してください）。

USBX デバイスクラス CDC-ACM モジュールの動作に関する重要な注意事項と制限事項

USBX デバイススタックまたは USBX ホストスタックは、制御ブロックの RAM を消費します。Synergy 構成ツールは、次の表に示すように、自動生成コードでメモリを USBX メモリプールに静的に割り当てます。[USBX on ux Configuration] セクションの Synergy 構成ツールで ux 上の USBX コンポーネントの [USBX Pool Memory Size] プロパティに、適切なメモリサイズ（バイト単位）を設定する必要があります。クラスを複数使用する場合は、このプロパティに合計メモリサイズを設定してください。

USBX メモリプールのメモリ（RAM）要件

USBX Class	S1 Parts	Other Parts
USBX Device CDC-ACM (ux_device_class_cdc_acm)	6.1KB	18KB

注 : 上の表に示す情報は、デフォルトの USBX 構成でコンパイルした場合のものです。

*注 : 上の表に示す USBX クラスのメモリサイズは、ビルド済みライブラリのものであり、ビルドには以下の構成が適用されています。*UX_THREAD_STACK_SIZE:S1 パーツには 512（バイト）、その他のパーツには 2048（バイト）。

アプリケーションは、USBX CDC-ACM instance_activate 関数コールバックに登録されたコールバック関数を使用して、インスタンスを保存する必要があります。リードとライトは、保存されたインスタンスを使用して実行されます。

- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.38.4 アプリケーションへの USBX デバイスクラス CDC-ACM モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに USBX デバイスクラス CDC-ACM モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユー

ザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

USBX デバイスクラス CDC-ACM モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

USBX デバイスクラス CDC-ACM モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ux_device_class_cdc_acm0 USBX Device Class CDC-ACM	Threads	New Stack> X-Ware> USBX> Device> Classes > CDC-ACM > USBX Device Class CDC-ACM

次の図に示すように、USBX デバイスクラス CDC-ACM モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

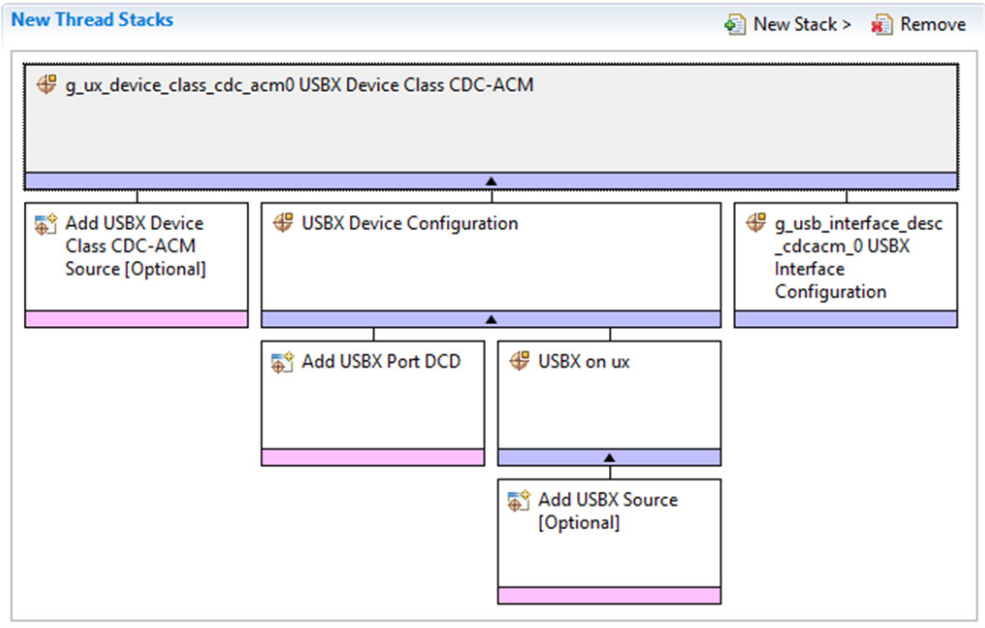


図 413:USBX デバイスクラス CDC-ACM モジュールのスタック

4.3.38.5 USBX デバイスクラス CDC-ACM モジュールの構成

ユーザーは必要な動作に合わせて USBX デバイスクラス CDC-ACM モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

USBX デバイスクラス CDC-ACM モジュールの構成設定

ISDE Property	Value	Description
Name	g_ux_device_class_cdc_acm0	Specify the name of the USBX Device CDC-ACM Class module instance. It must be a valid C symbol.
USBX CDC-ACM instance_activate Function Callback	ux_cdc_device0_instance_activate	Specify the name of the instance_activate user callback function for the USBX Device CDC-ACM Class module. Name must be a valid C symbol. See the USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device CDC-ACM Class" for more information about the instance_activate callback function.

ISDE Property	Value	Description
USBX CDC-ACM instance_deactivate Function Callback	ux-cdc_device0_instance_deactivate	Specify the name of the instance_deactivate user callback function for the USBX Device CDC-ACM Class module. Name must be a valid C symbol. Refer to the USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device CDC-ACM Class" for more information about the instance_activate callback function.
Name of generated initialization function	ux_device_class_cdc_acm_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

USBX デバイスクラス CDC-ACM のローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

USBX デバイスクラス CDC-ACM ソースの構成設定

ISDE Property	Value	Description
Show linkage warning	Enabled, Disabled Default: Enabled	Notification message for users will be shown if “Enabled" option is selected. This is just to warn users possible linkage errors by multiple symbol definitions. Select “Disabled” stops the notification message.

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX デバイス構成インスタンスの構成設定

ISDE Property	Value	Description
Vendor ID	0x045B	Specify Vendor ID assigned by USB-IF. This configuration is a part of the USB Device Descriptor (idVendor).
Product ID	0x0000	Specify Product ID assigned by manufacturer. This configuration is a part of the Device Descriptor (idProduct).
Device Release Number	0x0000	Specify Device Release Number in binary-coded decimal. This configuration is a part of the USB Device Descriptor (bcdDevice).

ISDE Property	Value	Description
Index of Manufacturing String Descriptor	0x00	Specify the Index of Manufacturer String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iManufacturer). Set zero if String Descriptor is not used. See section USBX-String-Framework-Configuration for more information.
Index of Product String Descriptor	0x00	Specify the Index of Product String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iProduct). Set zero if String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Index of Serial Number String Descriptor	0x00	Specify the Index of Serial Number String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iSerialNumber). Set zero if the String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Class Code	Communications(CDC), HID, Mass Storage, Miscellaneous, Vendor specific Default: Communications(CDC)	Select the USB Device Class Code. This configuration is a part of the USB Configuration Descriptor (bDeviceClass).

ISDE Property	Value	Description
Index of String Descriptor describing this configuration	0x00	Specify the Index of String Descriptor describing this configuration. This configuration is a part of the USB Configuration Descriptor (iConfiguration). Set zero if String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Size of USB Descriptor in bytes for this configuration (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	Specify the size of USB Descriptor in bytes. Modify the value for Vendor-specific Class, otherwise you can set zero to calculate the size automatically in the auto-generated code from Synergy Configuration tool. This configuration is a part of the USB Configuration Descriptor (wTotalLength).
Number of Interfaces (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	Specify the Number of interfaces supported by this configuration. Modify the value for Vendor-specific Class, otherwise you can set zero to calculate the value automatically in the auto-generated code from Synergy Configuration tool. This configuration is a part of the USB Configuration Descriptor (bNumInterfaces).
Self-Powered	Enable, Disable Default: Enable	Enable this configuration if your USB Device is a self-powered device. This configuration is a part of the USB Configuration Descriptor (bmAttributes bit6).

ISDE Property	Value	Description
Remote Wakeup	Enable, Disable Default: Disable	Enable this configuration if your USB Device supports remote wakeup. This configuration is a part of the USB Configuration Descriptor (bmAttributes bit5).
Maximum Power Consumption (in 2mA units)	50	Set the maximum power consumption of your device to indicate the amount of bus power required. This configuration is 2mA units, thus, the maximum 500 mA can be specified. This configuration is a part of the USB Configuration Descriptor (bMaxPower).
Supported Language Code	0x0409	Specify the Language ID Code. For example, 0x0409 English - United States. This configuration is used for Language ID Framework code generation. See section "USBX Language Framework Configuration" for more information.
Name of USBX String Framework	NULL	Specify the name of user defined USBX String Framework. This must be a valid C symbol. Set NULL if the String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Total index number of USB String Descriptors in USB String Framework	0	Specify the total number of index for String Descriptor. See section "USBX String Framework Configuration" for more information.

ISDE Property	Value	Description
Name of USBX Language Framework	NULL	Specify the name of user defined USBX Language Framework. This must be a valid C symbol. If '0' is set to the property "Total Number of Language Support", this configuration is ignored. See section "USBX Language Framework Configuration" for more information.
Number of Languages to support (US English is applied if zero is set)	0	Specify the total number of languages to support. See section "USBX String Framework Configuration" for more information. If '0' is set here, US English (0x0409) is applied as the default language.
Name of generated initialization function	ux_device_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX インタフェース構成 CDC-ACM インスタンスの構成設定

ISDE Property	Value	Description
Name	g_usb_interface_desc_cdcacm_0	Specify the name of USBX Interface Descriptor for CDC-ACM. It must be a valid C symbol.

ISDE Property	Value	Description
Interface Number of Communications Class interface	0x00	Specify the index number of Communications Class interface. This configuration is a part of the USB Interface Descriptor (bInterface). The number must not be duplicated with the Interface Number of Data Class interface. Also must not be duplicated with any Interface Numbers if your USB device consists of a USB composite device.
Interrupt Transfer endpoint to use for Communications Class	Endpoint 1-9 Default: Endpoint 3	Specify the Endpoint Number of Interrupt Endpoint. It must not be duplicated with ones for the other Endpoints.
Polling period for Interrupt Endpoint (in mS/125us units for FS/HS)	0x0F	Specify the Interval for polling Endpoint transfers. This configuration is valid for Interrupt Endpoint and ignored for Bulk Endpoints. Value is in frame counts (1ms units for FS mode and 125us units for HS mode).
Interface Number of Data Class interface	0x01	Specify the index number of Data Class interface. This configuration is a part of the USB Interface Descriptor (bInterface). The number must not be duplicated with the Interface Number of Communications Class interface. Also must not be duplicated with any Interface Numbers if your USB device consists of a USB composite device.
Bulk In Transfer endpoint to use for Data Class	Endpoint 1-9 Default: Endpoint 1	Specify the Endpoint Number of Bulk In Endpoint. It must not be duplicated with ones for the other Endpoints.

ISDE Property	Value	Description
Bulk Out Transfer endpoint to use for Data Class	Endpoint 1-9 Default: Endpoint 2	Specify the Endpoint Number of Bulk Out Endpoint. It must not be duplicated with ones for the other Endpoints.
Index of String Descriptor Describing Communications Class interface (Interface Descriptor: Interface)	0x00	Specify the index number of String Descriptor Describing Communications Class interface. This configuration is a part of the USB Interface Descriptor (iInterface). Set '0' if do not have String information for the interface.
Index of String Descriptor Describing Data Class interface (Interface Descriptor: Interface)	0x00	Specify the index number of String Descriptor Describing Data Class interface. This configuration is a part of the USB Interface Descriptor (iInterface). Set '0' if do not have String information for the interface.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBFS の sf_el_ux 上の USBX ポート DCD の構成設定

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for full-speed USB.
LDO Regulator (Only for S3 and S1 part MCUs)	Enable, Disable Default: Disable	Select the LDO regulator will be enabled.
Name	g_sf_el_ux_dcd_fs_0	Module name.
USB Controller Selection	USBFS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBHS 上の USBX ポート DCD の構成設定

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for high speed USB.
Name	g_sf_el_ux_dcd_hs_0	Module name.
USB Controller Selection	USBHS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ux 上の USBX インスタンスの構成設定

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	Name must be a valid C symbol.
USBX Pool Memory Size	18432	See section "Express Logic USBX Memory Requirements" for the required memory size for each classes.
User Callback for Host Event Notification (Only valid for USB Host)	NULL	Name must be a valid C symbol. The name of User defined USBX Host event notification can be given to this property.
Name of generated initialization function	ux_common_init0	Name of generated initialization function selection.

ISDE Property	Value	Description
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX デバイスクラス CDC-ACM モジュールのクロック構成

USB ペリフェラルモジュールは UCLK をクロックソースとして使用します。UCLK は 48 MHz の動作に合わせて構成する必要があります。SSP 構成ウィンドウで、[Clocks] タブを選択してクロックソースの設定を確認します。

USBX デバイスクラス CDC-ACM モジュールのピン構成

USB ペリフェラルモジュールは、MCU ピンを使用して外部デバイスと通信します。I/O ピンを選択し、外部デバイスの要件に合うように構成します。次の表では [SSP Configuration] ウィンドウでのピンの選択方法を示し、それ以降の表では USB ピンを例に使って選択プロセスを示します。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

USBFS および USBHS のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
USBFS	Pins	Select Peripherals > Connectivity: USBFS> USBFS0
USBHS	Pins	Select Peripherals > Connectivity: USBHS> USBHS0

注 : 選択シーケンスでは、USBFS0 または USBHS0 がドライバーに必要なハードウェアターゲットであることを想定しています。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select device as the Operation Mode

Property	Value	Description
USBDP	USBDP	USBDP pin
USBDM	USBDM	USBDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	None	VBUSEN pin
VBUS	None, P407 Default: P407	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID Pin
VCCUSB	VCCUSB	VCCUSB pin
VSSUSB	VSSUSB	VSSUSB pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select Device as the Operation Mode
USBHSDP	USBHSDP	USBHSDP pin
USBHSDM	USBHSDM	USBHSDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	PB00	VBUSEN pin

Property	Value	Description
VBUS	PB01	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID pin
USBHSRREF	USBHSRREF	USBHSRREF pin
AVCCUSBHS	AVCCUSBHS	AVCCUSBHS pin
AVSSUSBHS	AVSSUSBHS	AVSSUSBHS pin
PVSSUSBHS	PVSSUSBHS	PVSSUSBHS pin
VCCUSBHS	VCCUSBHS	VCCUSBHS pin
VSS1USBHS	VSS1USBHS	VSS1USBHS pin
VSS2USBHS	VSS2USBHS	VSS2USBHS pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.38.6 アプリケーションでの USBX デバイスクラス CDC-ACM モジュールの使用

コンフィギュレータは、USBX デバイスクラス CDC-ACM モジュールの作成と登録に必要な処理を生成しますが、通信はデバイスがホストに接続されてから行う必要があります。

アプリケーションで USBX デバイスクラス CDC-ACM モジュールを使用する際の一般的な手順は次のとおりです。

1) USBX CDC-ACM instance_activate 関数コールバックに登録されているコールバック関数が呼び出されるまで待ちます。
2) コールバック関数で、インスタンスを保存します。
3) 受信データのリードには、ux_device_class_cdc_acm_read API を使用します。
4) データの送信には、ux_device_class_cdc_acm_write API を使用します。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

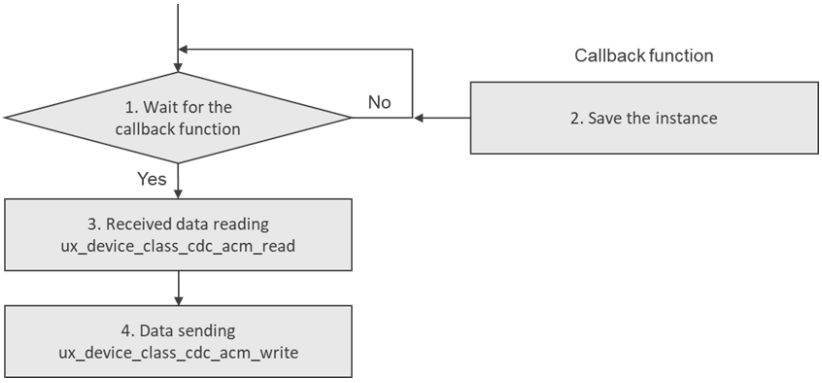


図 414:一般的な USBX デバイスクラス CDC-ACM モジュールアプリケーションのフロー図

4.3.39 USBX デバイスクラス HID

USBX™ デバイスクラスヒューマンインタフェースデザイン（HID）モジュールは、HID アプリケーション用のハイレベル API を提供し、USBX デバイスクラス HID ソース、USBX ホスト構成、USBX ソース、USBX ポート HCD を構成します。USBX デバイスクラス HID モジュールは、Synergy MCU 上の USB ペリフェラルを使用します。

4.3.39.1 USBX デバイスクラス HID モジュールの特長

USB デバイスクラス HID モジュールにより、USB ホストシステムは、キーボードデバイス、マウスデバイスなどの HID デバイスと通信することが可能になります。このクラスは、USB 標準に基づいており、HID 標準のサブセットです。USBX デバイスクラス HID モジュールには、以下の主要な特長があります。

- USB ハイスピード（USBHS）または USB フルスピード（USBFS）をサポートします。
- 性能向上を目的に、受信および送信データ転送ドライバーを使用します。
- リードとライトのためのハイレベル API を提供します。

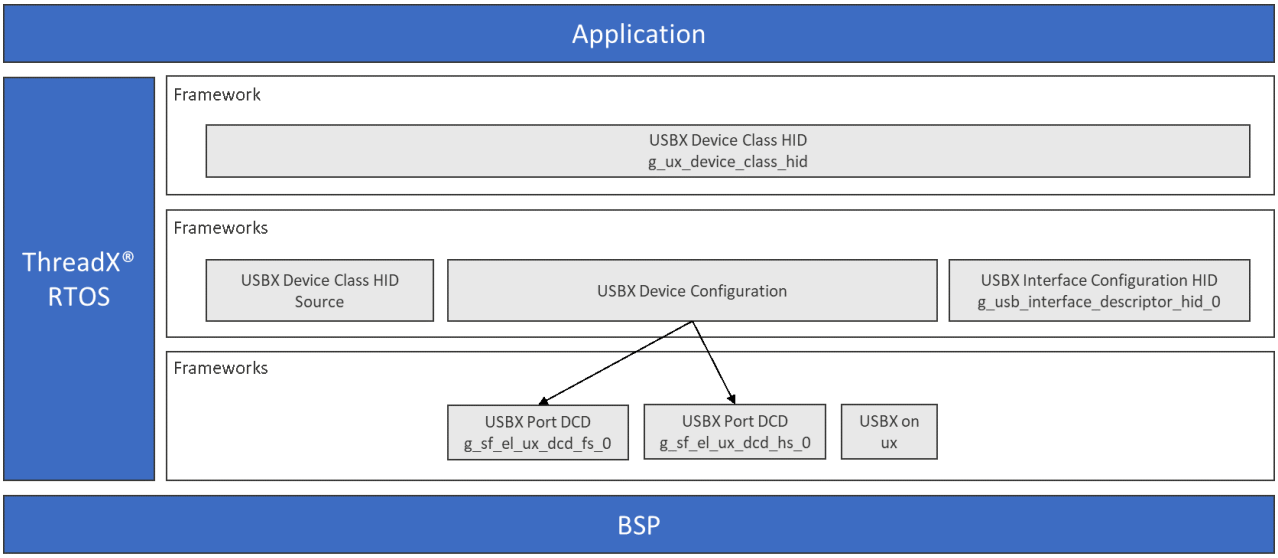


図 415:USBX デバイスクラス HID モジュールのブロック図

4.3.39.2 USBX デバイスクラス HID モジュールの API の概要

USBX デバイスクラス HID モジュールでは、HID イベントとレポートを送受信するための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

USBX デバイスクラス HID モジュールの API の要約

Function Name	Example API Call and Description
ux_device_class_hid_event_set	ux_device_class_hid_event_set (hid, &hid_event); This function is called when an application needs to send a HID event to the host.
ux_device_class_hid_event_get	ux_device_class_hid_event_get (hid, & hid_event); This function is called when an application needs to receive a HID event from the host.

Function Name	Example API Call and Description
ux_device_class_hid_report_set	ux_device_class_hid_report_set (hid, descriptor_type, request_index, host_length); This function is called when an application needs to send a HID report to the host.
ux_device_class_hid_report_get	ux_device_class_hid_report_get (hid, descriptor_type, request_index, host_length); This function is called when an application needs to receive a HID report to the host.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
UX_SUCCESS	The data transfer was completed.
UX_TRANSFER_TIMEOUT	Transfer timeout, reading/writing not completed.
UX_MEMORY_INSUFFICIENT	Not enough memory.
UX_HOST_CLASS_UNKNOWN	Wrong class instance.
UX_FUNCTION_NOT_SUPPORTED	Unknown IOCTL function.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.3.39.3 USBX デバイスクラス HID モジュールの動作の概要

USBX リソースの初期化

USBX には、独自のメモリマネージャがあります。メモリは、USBX のホスト側またはデバイス側が初期化される前に割り当てる必要があります。USBX メモリマネージャは、メモリのキャッシュが可能なシステムに対応できます。

USB ホストコントローラの定義

USBX をホストモードで動作させるには、USB ホストコントローラを 1 つ以上定義する必要があります。この定義は、アプリケーションの初期化ファイルに含める必要があります。SSP は、USB ホストコントローラドライバーがスレッドタスクに追加された場合に、USB ホストコントローラを定義します。

デバイスクラスの定義

USBX には、1 つ以上のデバイスクラスを定義する必要があります。USB クラスは、USB スタックが USB デバイスを構成した後に、USB デバイスを駆動するために必要です。USB クラスは、デバイスに固有です。USB デバイス記述子に含まれるインタフェースの数に応じて、USB を駆動するために 1 つ以上のクラスが必要になることがあります。

USB クラスのバインド

デバイスが構成されると、トポロジマネージャは、クラスマネージャがデバイスインタフェース記述子を調べ、デバイスの検出を続行することを許可します。1 つのデバイスに複数のインタフェース記述子が存在する場合があります。

インタフェースは、デバイスの機能を表します。たとえば、USB スピーカには、オーディオストリーミング用、オーディオ制御用、各種スピーカボタンの管理用の 3 つのインタフェースが存在します。

クラスマネージャには、デバイスインタフェースを 1 つ以上のクラスに結合する 2 つのメカニズムがあります。1 つはインタフェース記述子に含まれる PID と VID（製品 ID とベンダ ID）の組み合わせを使用する方法、もう 1 つはクラス、サブクラス、プロトコルの組み合わせを使用する方法です。

PID と VID の組み合わせは、一般的なクラスでは駆動できないインタフェースに有効です。クラス、サブクラス、プロトコルの組み合わせは、プリンタ、ハブ、ストレージ、オーディオ、ヒューマンインタフェースデザイン（HID）など、USB-IF が認定したクラスに属するインタフェースに使用されます。

クラスマネージャには、USBX の初期化で登録されたクラスのリストが格納されます。クラスマネージャは、デバイスのインタフェースを管理するクラスが決まるまでクラスを 1 つずつ呼び出します。1 つのクラスが管理できるインタフェースは、1 つだけです。USB オーディオスピーカの場合は、クラスマネージャは、各インタフェースに対してすべてのクラスを呼び出します。

クラスがインタフェースを受け入れると、そのクラスの新しいインスタンスが作成され、クラスマネージャがインタフェースのデフォルトの代替設定を検索します。1 つのデバイスの各インタフェースには、1 つ以上の代替設定が存在する場合があります。クラスで変更されない限り、デフォルトで代替設定 0 が使用されます。

デフォルトの代替設定の場合は、クラスマネージャが代替設定に含まれるすべてのエンドポイントをマウントします。各エンドポイントのマウントが成功すると、インタフェースの初期化を完了するクラスにクラスマネージャが戻ってジョブを完了します。

USBX デバイスクラス HID モジュールの動作に関する重要な注意事項と制限事項

USBX デバイススタックまたは USBX ホストスタックは、制御ブロックの RAM を消費します。Synergy 構成ツールは、次の表に示すように、自動生成コードでメモリを USBX メモリプールに静的に割り当てます。[USBX on ux Configuration] セクションの Synergy 構成ツールで ux 上の USBX コンポーネントの [USBX Pool Memory Size] プロパティに、適切なメモリサイズ（バイト単位）を設定する必要があります。クラスを複数使用する場合は、このプロパティに合計メモリサイズを設定してください。

USBX メモリプールのメモリ（RAM）要件

USBX Class	S1 Parts	Other Parts
USBX Device HID (ux_device_class_hid)	6.1KB	12KB

注 : 上の表に示す情報は、デフォルトの USBX 構成でコンパイルした場合のものです。

注 : 上の表に示す USBX クラスのメモリサイズは、ビルド済みライブラリのものであり、ビルドには以下の構成が適用されています。

UX_THREAD_STACK_SIZE:S1 パーツには 512（バイト）、その他のパーツには 2048（バイト）。

- アプリケーションは、グローバル変数である _ux_system_slave; からスレーブデバイスの HID インスタンスを取得し、このインスタンスを使用して送受信を実行します。
- [USBX Interface Configuration HID Driver] の [Protocol code] プロパティを使用して、実際のデバイスの動作を判定します。
- このモジュールでは、USB コントローラの割り込みが有効になっている必要があります。
- このモジュールは、USB コントローラの割り込みを使用します。適切な動作を確保するために、Synergy 構成ツールで適切な割り込み優先順位レベルを設定してください。
- このモジュールが使用されると、転送モジュール（DMAC または DTC として実装）の割り込みが使用されます。Synergy 構成ツールで適切な割り込み優先順位レベルを設定してください。優先順位レベルは USB コントローラよりも高くします。設定しないと機能しません。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.39.4 アプリケーションへの USBX デバイスクラス HID モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに USBX デバイスクラス HID モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

USBX デバイスクラス HID モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

USBX デバイスクラス HID モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ux_device_class_hid USBX Device Class HID	Threads	New Stack> X-Ware> USBX> Device > Classes > HID > USBX Device Class HID

次の図に示すように、USBX デバイスクラス HID モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」と

いうテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

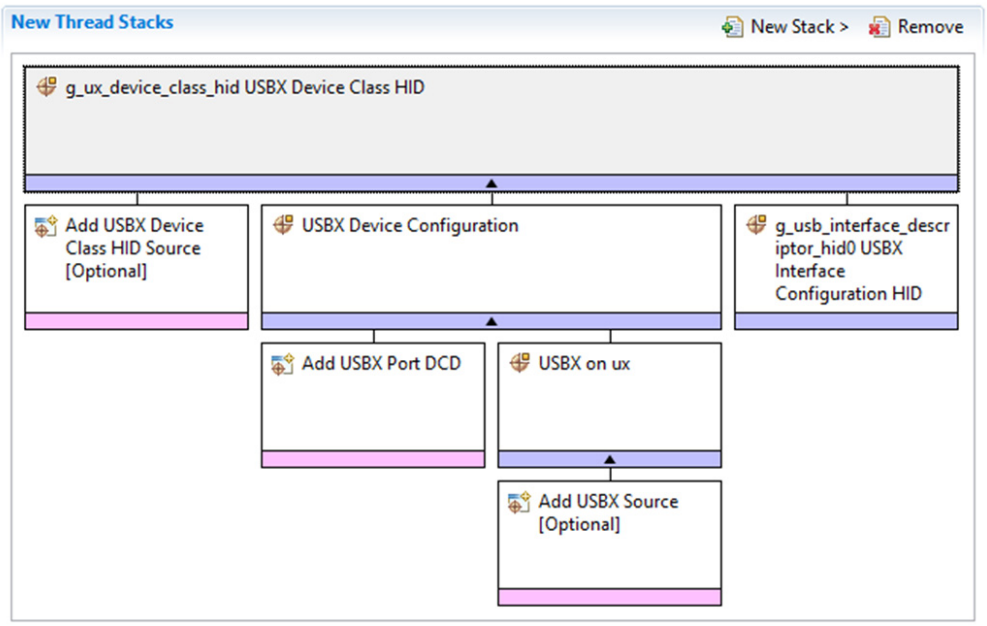


図 416:**USBX デバイスクラス HID モジュールのスタック**

4.3.39.5 USBX デバイスクラス HID モジュールの構成

ユーザーは必要な動作に合わせて USBX デバイスクラス HID モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

USBX デバイスクラス HID モジュールの構成設定

ISDE Property	Value	Description
Name	g_ux_device_class_hid	Specify the name of USBX Interface Descriptor for HID Class. It must be a valid C symbol.
USBX Device HID Entry Function	ux_device_class_hid_entry	Specify the name of a user callback function to get an event from HID Class. Name must be a valid C symbol. Refer to the USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device HID Class" for more information about the user callback function.
USBX Device HID User Callback Function	ux_hid_device_callback	Specify the name of user entry function for the USBX Device HID Class module. Name must be a valid C symbol. See the USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device HID Class" for more information about the user entry function.
USBX Device HID Instance Activate Callback Function	NULL	Specify the name of instance_activate user callback function for the USBX Device HID Class module. Name must be a valid C symbol. See the USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device HID Class" for more information about the instance_activate callback function.

ISDE Property	Value	Description
USBX Device HID Instance Deactivate Callback Function	NULL	Specify the name of instance_deactivate user callback function for the USBX Device HID Class module. Name must be a valid C symbol. Refer to the USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device HID Class" for more information about the instance_activate callback function.
Multiple HID Report Support	Enable, Disable Default: Disable	Set Enable to support multiple HID report. This configuration is used to indicate which data fields are represented in each report structure.
Name of generated initialization function	ux_device_class_hid_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注:ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

USBX デバイスクラス HID のローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

USBX デバイスクラス HID ソースの構成設定

ISDE Property	Value	Description
Show linkage warning	Enabled, Disabled Default: Enabled	Notification message for users will be shown if “Enabled" option is selected. This is just to warn users possible linkage errors by multiple symbol definitions. Select “Disabled” stops the notification message.

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX デバイス構成インスタンスの構成設定

ISDE Property	Value	Description
Vendor ID	0x045B	Specify Vendor ID assigned by USB-IF. This configuration is a part of the USB Device Descriptor (idVendor).
Product ID	0x0000	Specify Product ID assigned by manufacturer. This configuration is a part of the Device Descriptor (idProduct).
Device Release Number	0x0000	Specify Device Release Number in binary-coded decimal. This configuration is a part of the USB Device Descriptor (bcdDevice).

ISDE Property	Value	Description
Index of Manufacturing String Descriptor	0x00	Specify the Index of Manufacturer String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iManufacturer). Set zero if String Descriptor is not used. See section USBX-String-Framework-Configuration for more information.
Index of Product String Descriptor	0x00	Specify the Index of Product String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iProduct). Set zero if String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Index of Serial Number String Descriptor	0x00	Specify the Index of Serial Number String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iSerialNumber). Set zero if the String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Class Code	Communications(CDC), HID, Mass Storage, Miscellaneous, Vendor specific Default: Communications(CDC)	Select the USB Device Class Code. This configuration is a part of the USB Configuration Descriptor (bDeviceClass).

ISDE Property	Value	Description
Index of String Descriptor describing this configuration	0x00	Specify the Index of String Descriptor describing this configuration. This configuration is a part of the USB Configuration Descriptor (iConfiguration). Set zero if String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Size of USB Descriptor in bytes for this configuration (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	Specify the size of USB Descriptor in bytes. Modify the value for Vendor-specific Class, otherwise you can set zero to calculate the size automatically in the auto-generated code from Synergy Configuration tool. This configuration is a part of the USB Configuration Descriptor (wTotalLength).
Number of Interfaces (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	Specify the Number of interfaces supported by this configuration. Modify the value for Vendor-specific Class, otherwise you can set zero to calculate the value automatically in the auto-generated code from Synergy Configuration tool. This configuration is a part of the USB Configuration Descriptor (bNumInterfaces).
Self-Powered	Enable, Disable Default: Enable	Enable this configuration if your USB Device is a self-powered device. This configuration is a part of the USB Configuration Descriptor (bmAttributes bit6).

ISDE Property	Value	Description
Remote Wakeup	Enable, Disable Default: Disable	Enable this configuration if your USB Device supports remote wakeup. This configuration is a part of the USB Configuration Descriptor (bmAttributes bit5).
Maximum Power Consumption (in 2mA units)	50	Set the maximum power consumption of your device to indicate the amount of bus power required. This configuration is 2mA units, thus, the maximum 500 mA can be specified. This configuration is a part of the USB Configuration Descriptor (bMaxPower).
Supported Language Code	0x0409	Specify the Language ID Code. For example, 0x0409 English - United States. This configuration is used for Language ID Framework code generation. See section "USBX Language Framework Configuration" for more information.
Name of USBX String Framework	NULL	Specify the name of user defined USBX String Framework. This must be a valid C symbol. Set NULL if the String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Total index number of USB String Descriptors in USB String Framework	0	Specify the total number of index for String Descriptor. See section "USBX String Framework Configuration" for more information.

ISDE Property	Value	Description
Name of USBX Language Framework	NULL	Specify the name of user defined USBX Language Framework. This must be a valid C symbol. If '0' is set to the property "Total Number of Language Support", this configuration is ignored. See section "USBX Language Framework Configuration" for more information.
Number of Languages to support (US English is applied if zero is set)	0	Specify the total number of languages to support. See section "USBX String Framework Configuration" for more information. If '0' is set here, US English (0x0409) is applied as the default language.
Name of generated initialization function	ux_device_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX インタフェース構成 HID インスタンスの構成設定

ISDE Property	Value	Description
Name	g_usb_interface_descriptor_hid0	Specify the name of USBX Interface Descriptor for CDC-ACM. It must be a valid C symbol.

ISDE Property	Value	Description
Protocol code: None(0) Keyboard(1) Mouse(2) Keyboard+Mouse(3)	0x00	Both the keyboard and mouse interface will be available only if Keyboard+Mouse(3) protocol is selected. Select device class as Device (0x00) in device configuration when protocol selected is (Keyboard+Mouse).
(Keyboard) Interface Number of HID Class interface	0x00	Keyboard Interface will be available for use when protocol code selected is either Keyboard(1) or Keyboard+Mouse(3).
(Keyboard) Endpoint Number to be used for Interrupt-In	Endpoint 1-9 Default: Endpoint 1	Specify the Endpoint Number of Interrupt-In Endpoint. It must not be duplicated with ones for the other Endpoints.
(Keyboard) Maximum packet size in bytes for Interrupt-In	0x8	Specify the maximum packet size this endpoint is capable of sending or receiving when this configuration is selected.
Interval for polling Interrupt-In EP for data transfers (milliseconds)	0x8	Specify the Interval for polling Endpoint transfers. This configuration is valid for Interrupt-In Endpoint. Value is in frame counts (1ms units for FS mode and 125us units for HS mode).
(Keyboard) Interrupt-Out Endpoint (Optional)	Enable, Disable Default: Disable	This configuration is reserved and currently not used.
Endpoint Number for Interrupt-Out (Optional)	Endpoint 1-9 Default: Endpoint 3	This configuration is reserved and currently not used.
(Keyboard) Maximum packet size in bytes for Interrupt-Out EP (Optional)	0x8	This configuration is reserved and currently not used.

ISDE Property	Value	Description
(Keyboard) Interval for polling Interrupt-Out EP for data transfers (milliseconds) (Optional)	0x8	This configuration is reserved and currently not used.
(Mouse) Interface Number of HID Class interface	0x01	Mouse Interface will be available for use when protocol code selected is either Mouse(2) or Keyboard+Mouse(3).
(Mouse) Endpoint Number to be used for Interrupt-In	Endpoint 1-9 Default: Endpoint 2	Specify the Endpoint Number of Interrupt-In Endpoint. It must not be duplicated with ones for the other Endpoints.
(Mouse) Maximum packet size in bytes for Interrupt-In	0x8	Specify the maximum packet size this endpoint is capable of sending or receiving when this configuration is selected.
Interval for polling Interrupt-In EP for data transfers (milliseconds)	0x8	Specify the Interval for polling Endpoint transfers. This configuration is valid for Interrupt-In Endpoint. Value is in frame counts (1ms units for FS mode and 125us units for HS mode).
(Mouse) Interrupt-Out Endpoint (Optional)	Enable, Disable Default: Disable	This configuration is reserved and currently not used.
Endpoint Number for Interrupt-Out (Optional)	Endpoint 1-9 Default: Endpoint 4	This configuration is reserved and currently not used.
(Mouse) Maximum packet size in bytes for Interrupt-Out EP (Optional)	0x8	This configuration is reserved and currently not used.
(Mouse) Interval for polling Interrupt-Out EP for data transfers (milliseconds) (Optional)	0x8	This configuration is reserved and currently not used.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBFS の sf_el_ux 上の USBX ポート DCD の構成設定

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for full-speed USB.
LDO Regulator (Only for S3 and S1 part MCUs)	Enable, Disable Default: Disable	Select the LDO regulator will be enabled.
Name	g_sf_el_ux_dcd_fs_0	Module name.
USB Controller Selection	USBFS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBHS 上の USBX ポート DCD の構成設定

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for high speed USB.
Name	g_sf_el_ux_dcd_hs_0	Module name.
USB Controller Selection	USBHS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ux 上の USBX インスタンスの構成設定

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	Name must be a valid C symbol.
USBX Pool Memory Size	18432	See section "Express Logic USBX Memory Requirements" for the required memory size for each classes.
User Callback for Host Event Notification (Only valid for USB Host)	NULL	Name must be a valid C symbol. The name of User defined USBX Host event notification can be given to this property.
Name of generated initialization function	ux_common_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX デバイスクラス HID モジュールのクロック構成

USB ペリフェラルモジュールは UCLK をクロックソースとして使用します。UCLK は 48 MHz の動作に合わせて構成する必要があります。SSP 構成ウィンドウで、[Clocks] タブを選択してクロックソースの設定を確認します。

USBX デバイスクラス HID モジュールのピン構成

USB ペリフェラルモジュールは、MCU ピンを使用して外部デバイスと通信します。I/O ピンを選択し、外部デバイスの要件に合うように構成します。次の表では [SSP Configuration] ウィンドウでのピンの選択方法を示し、それ以降の表では USB ピンを例に使って選択プロセスを示します。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

USBFS および USBHS のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
USBFS	Pins	Select Peripherals > Connectivity: USBFS> USBFS0
USBHS	Pins	Select Peripherals > Connectivity: USBHS> USBHS0

注 : 選択シーケンスでは、USBFS0 または USBHS0 がドライバーに必要なハードウェアターゲットであることを想定しています。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select device as the Operation Mode
USBDP	USBDP	USBDP pin
USBDM	USBDM	USBDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	None	VBUSEN pin
VBUS	None, P407 Default: P407	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID Pin
VCCUSB	VCCUSB	VCCUSB pin

Property	Value	Description
VSSUSB	VSSUSB	VSSUSB pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select Device as the Operation Mode
USBHSDP	USBHSDP	USBHSDP pin
USBHSDM	USBHSDM	USBHSDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	PB00	VBUSEN pin
VBUS	PB01	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID pin
USBHSRREF	USBHSRREF	USBHSRREF pin
AVCCUSBHS	AVCCUSBHS	AVCCUSBHS pin
AVSSUSBHS	AVSSUSBHS	AVSSUSBHS pin
PVSSUSBHS	PVSSUSBHS	PVSSUSBHS pin
VCCUSBHS	VCCUSBHS	VCCUSBHS pin
VSS1USBHS	VSS1USBHS	VSS1USBHS pin
VSS2USBHS	VSS2USBHS	VSS2USBHS pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.39.6 アプリケーションでの USBX デバイスクラス HID モジュールの使用

コンフィギュレータは、USBX デバイスクラス HID モジュールの作成と登録に必要な処理を生成しますが、通信はデバイスがホストに接続されてから行う必要があります。

アプリケーションで USBX デバイスクラス HID モジュールを使用する際の一般的な手順は次のとおりです。

1) ux_system_slave スレーブデバイスポインタを取得します。
2) スレーブデバイスの ux_slave_device_state が設定されるまで待機します。
3) HID イベントの送信には、ux_device_class_hid_event_set API を使用します。
4) 受信した HID イベントのリードには、ux_device_class_hid_event_get API を使用します。
5) HID レポートの送信には、ux_device_class_hid_report_set API を使用します。
6) 受信した HID レポートのリードには、ux_device_class_hid_report_get API を使用します。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

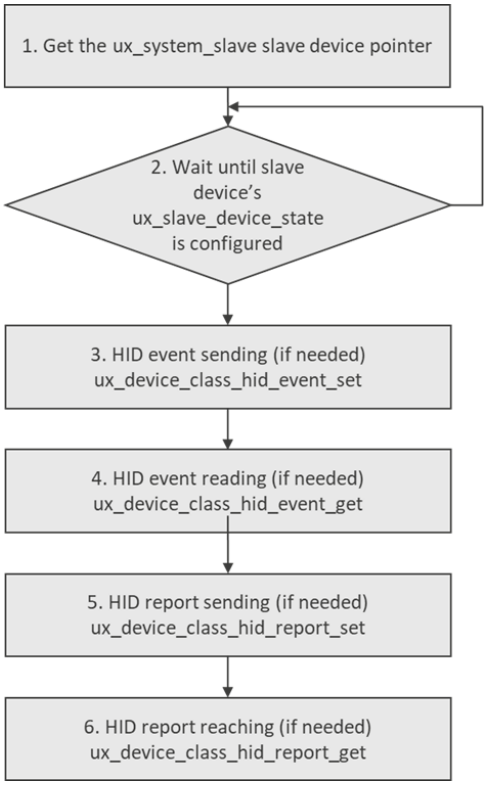


図 417:一般的な USBX デバイスクラス HID モジュールアプリケーションのフロー図

4.3.40 USBX デバイスクラス記憶装置

USBX ™デバイスクラス大容量記憶装置モジュールは、USB 大容量記憶装置アプリケーションに、USB フルスピード（USBFS）または USB ハイスピード（USBHS）のためのハイレベル API を提供します。USBX デバイスクラス大容量記憶装置モジュールは、Synergy MCU 上の USB およびデータ転送ペリフェラルを使用します。

4.3.40.1 USBX デバイスクラス大容量記憶装置モジュールの特長

- ThreadX® 対応フレームワーク
- ストレージメディアのパラメータのセットアップ
  - 最後の LBA
  - セクターあたりのバイト数
  - ストレージメディアのタイプ
  - 取り外し可能フラグ
- USB デバイスの構成（デバイス構成）
  - ベンダー ID
  - 製品 ID
  - デバイスリリース番号
  - シリアル番号文字列記述子のインデックス
- サポートされる USB 仕様（DCD）
  - USBFS
  - USBHS
- USB デバイス割り込み（DCD）

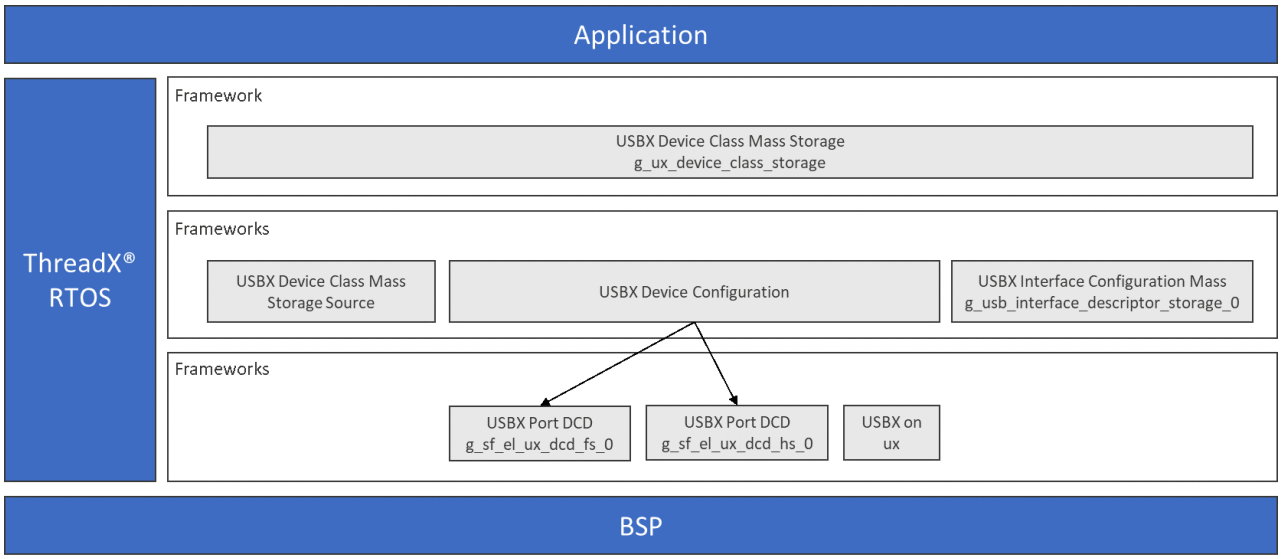


図 418:USBX デバイスクラス大容量記憶装置モジュールのブロック図

4.3.40.2 USBX デバイスクラス大容量記憶装置モジュールの API の概要

USBX デバイスクラス大容量記憶装置モジュールは、初期化プロセスを自動的に追加します。ユーザーアプリケーションに必要なのは、メディアアクセスのためのコールバック関数を準備することだけです。USBX デバイスクラス大容量記憶装置モジュールの機能は、必要にならない限り、使用する必要はありません。

注 :USBX デバイススタックの詳細については、『USBX Device Stack User's Manual』を参照してください。

4.3.40.3 USBX デバイスクラス大容量記憶装置モジュールの動作の概要

USBX デバイスクラス大容量記憶装置モジュールは、初期化プロセスを自動的に追加します。このプロセスは、指定されたパラメータで内部情報を初期化し、大容量記憶装置クラスを処理するための内部スレッドを作成します。この内部スレッドですべての USB メッセージを処理します。

USBX デバイスクラス大容量記憶装置モジュールの動作に関する重要な注意事項と制限事項

USBX デバイススタックまたは USBX ホストスタックは、制御ブロックの RAM を消費します。Synergy 構成ツールは、次の表に示すように、自動生成コードでメモリを USBX メモリプールに静的に割り当てます。[USBX on ux Configuration] セクションの Synergy 構成ツールで ux 上の USBX コンポーネントの [USBX Pool Memory Size] プロパティに、適切なメモリサイズ（バイト単位）を設定する必要があります。クラスを複数使用する場合は、このプロパティに合計メモリサイズを設定してください。

USBX メモリプールのメモリ（RAM）要件

USBX Class	S1 Parts	Other Parts
USBX Device Mass Storage (ux_device_class_storage)	6.1KB	19KB

注 : 上の表に示す情報は、デフォルトの USBX 構成でコンパイルした場合のものです。

注 : 上の表に示す USBX クラスのメモリサイズは、ビルド済みライブラリのものであり、ビルドには以下の構成が適用されています。

UX_THREAD_STACK_SIZE:S1 パーツには 512（バイト）、その他のパーツには 2048（バイト）。

- USBX デバイスストレージクラスは、複数の論理ユニット番号（LUN）をサポートし、同時に CD-ROM とフラッシュディスクとして機能するストレージデバイスを作成することを可能にします。
- このモジュールは、複合デバイスをサポートしていません。
- このモジュールの動作に関するその他の制限事項については、最新の *SSP リリースノート*を参照してください。

4.3.40.4 アプリケーションへの USBX デバイスクラス大容量記憶装置モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに USBX デバイスクラス大容量記憶装置モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

USBX デバイスクラス大容量記憶装置モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

USBX デバイスクラス大容量記憶装置モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ux_device_class_storage USBX Device Class Mass Storage	Threads	New Stack> X-Ware> USBX> Device> Classes> Mass Storage> USBX Device Class Mass Storage

次の図に示すように、USBX デバイスクラス大容量記憶装置モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。

（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

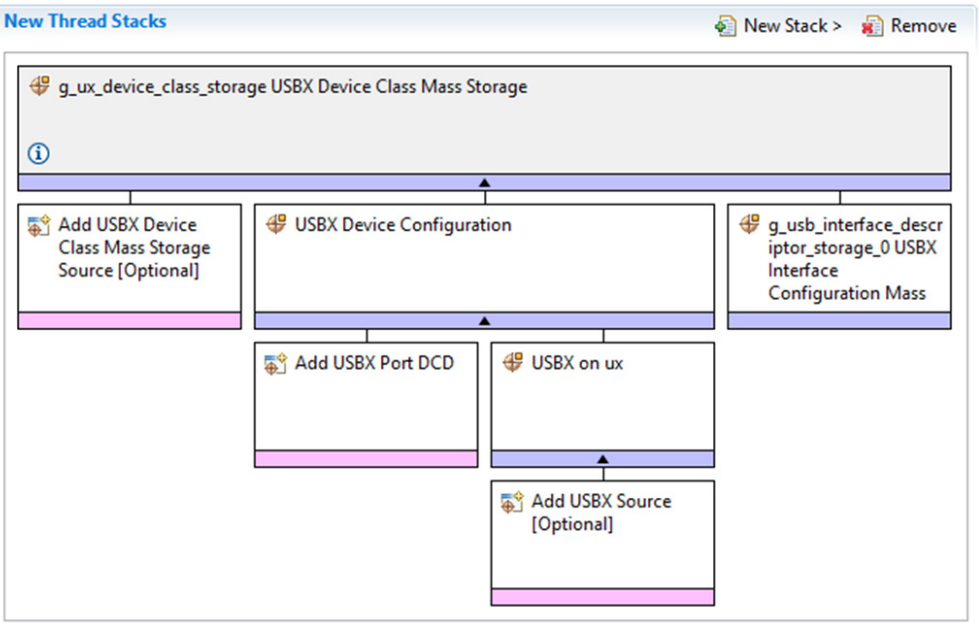


図 419:USBX デバイスクラス大容量記憶装置モジュールのスタック

4.3.40.5 USBX デバイスクラス大容量記憶装置モジュールの構成

ユーザーは必要な動作に合わせて USBX デバイスクラス大容量記憶装置モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注：次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

USBX デバイスクラス大容量記憶装置モジュールの構成設定

ISDE Property	Value	Description
Name	g_ux_device_class_storage	Specify the name of USBX Interface Descriptor for Mass Storage Class. It must be a valid C symbol.
Mass Storage Class Parameter Setup	Auto (Simple Auto Setup if LUN is 1), Manual (User Manual Setup if LUN is greater than 1) Default: Auto	Manual (User Manual Setup if LUN is greater than 1). Simple Auto Setup if LUN is 1.
User Setup Callback (Only valid if Parameter Setup is Auto)	ux_device_class_storage_user_setup	Specify the name of user callback function to setup the storage parameter setup. This parameter is only valid when the configuration "Mass Storage Class Parameter Setup" is "Auto".
Last LBA of Storage Media (Only valid if Parameter Setup is Auto)	0	Specify the last LBA of storage media device (the number of sectors available in the media - 1). This parameter is only valid when the configuration "Mass Storage Class Parameter Setup" is "Auto".
Bytes Per Sector of Storage Media (Only valid if Parameter Setup is Auto)	512	Specify the sector size of storage media. It can take multiple of 512 such as 512, 4K bytes. This parameter is only valid when the configuration "Mass Storage Class Parameter Setup" is "Auto".

ISDE Property	Value	Description
Type of Storage Media (Only valid if Parameter Setup is Auto)	0	Specify the type of storage media device. Typically, the value takes following values. Flash Drive (0), CD-ROM device (5) This parameter is only valid when the configuration "Mass Storage Class Parameter Setup" is "Auto".
Removable Flag of Storage Media (Only valid if Parameter Setup is Auto)	0x80	Specify the Removable Flag value of Storage Media. This parameter is only valid when the configuration "Mass Storage Class Parameter Setup" is "Auto".
Media Read Function Callback (Only valid if Parameter Setup is Auto)	ux_device_msc_media_read	Specify the C symbol name of Media Read callback for the USBX Device Mass Storage Class. The function is to be called back from the Class library when read access to the USB storage device is requested from user application. Refer USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device Storage Class" for the function definition.

ISDE Property	Value	Description
Media Write Function Callback (Only valid if Parameter Setup is Auto)	ux_device_msc_media_write	Specify the C symbol name of Media Write callback for the USBX Device Mass Storage Class. The function is to be called back from the Class library when write access to the USB storage device is requested from user application. Refer USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device Storage Class" for the function definition.
Media Status Function Callback (Only valid if Parameter Setup is Auto)	ux_device_msc_media_status	Specify the C symbol name of Media Status callback for the USBX Device Mass Storage Class. The function is to be called back from the Class library when status inquiry to the USB storage device is requested from user application. Refer USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device Storage Class" for the function definition.
USBX Device Storage Instance Activate Callback Function	NULL	Specify the name of instance_activate user callback function for the USBX Device Mass Storage Class module. Name must be a valid C symbol. See the USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device Storage Class" for more information about the instance_activate callback function.

ISDE Property	Value	Description
USBX Device Storage Instance Deactivate Callback Function	NULL	Specify the name of instance_deactivate user callback function for the USBX Device Mass Storage Class module. Name must be a valid C symbol. Refer to the USBX Stack User's Manual "Chapter 5: USBX Device Class Considerations USB Device Storage Class" for more information about the instance_activate callback function
Name of generated initialization function	ux_device_class_storage_init0	Name of generated initialization function selection
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection

注:設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注:ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

USBX デバイスクラス大容量記憶装置のローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

USBX デバイスクラス大容量記憶装置ソースの構成設定

ISDE Property	Value	Description
Maximum number of SCSI logical units	Value must be greater than 0 or empty Default: 2	UX_MAX_SLAVE_LUN This value represents the maximum number of SCSI logical units represented in the device storage class driver.
Show linkage warning	Enabled, Disabled Default: Enabled	Notification message for users will be shown if “Enabled" option is selected. This is just to warn users possible linkage errors by multiple symbol definitions. Select “Disabled” stops the notification message.

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX デバイス構成インスタンスの構成設定

ISDE Property	Value	Description
Vendor ID	0x045B	Specify Vendor ID assigned by USB-IF. This configuration is a part of the USB Device Descriptor (idVendor).
Product ID	0x0000	Specify Product ID assigned by manufacturer. This configuration is a part of the Device Descriptor (idProduct).
Device Release Number	0x0000	Specify Device Release Number in binary-coded decimal. This configuration is a part of the USB Device Descriptor (bcdDevice).

ISDE Property	Value	Description
Index of Manufacturing String Descriptor	0x00	Specify the Index of Manufacturer String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iManufacturer). Set zero if String Descriptor is not used. See section USBX-String-Framework-Configuration for more information.
Index of Product String Descriptor	0x00	Specify the Index of Product String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iProduct). Set zero if String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Index of Serial Number String Descriptor	0x00	Specify the Index of Serial Number String Descriptor defined in the USBX String Framework. This configuration is a part of the USB Device Descriptor (iSerialNumber). Set zero if the String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Class Code	Communications(CDC), HID, Mass Storage, Miscellaneous, Vendor specific Default: Communications(CDC)	Select the USB Device Class Code. This configuration is a part of the USB Configuration Descriptor (bDeviceClass).

ISDE Property	Value	Description
Index of String Descriptor describing this configuration	0x00	Specify the Index of String Descriptor describing this configuration. This configuration is a part of the USB Configuration Descriptor (iConfiguration). Set zero if String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Size of USB Descriptor in bytes for this configuration (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	Specify the size of USB Descriptor in bytes. Modify the value for Vendor-specific Class, otherwise you can set zero to calculate the size automatically in the auto-generated code from Synergy Configuration tool. This configuration is a part of the USB Configuration Descriptor (wTotalLength).
Number of Interfaces (Modify this value only for Vendor-specific Class, otherwise set zero)	0x00	Specify the Number of interfaces supported by this configuration. Modify the value for Vendor-specific Class, otherwise you can set zero to calculate the value automatically in the auto-generated code from Synergy Configuration tool. This configuration is a part of the USB Configuration Descriptor (bNumInterfaces).
Self-Powered	Enable, Disable Default: Enable	Enable this configuration if your USB Device is a self-powered device. This configuration is a part of the USB Configuration Descriptor (bmAttributes bit6).

ISDE Property	Value	Description
Remote Wakeup	Enable, Disable Default: Disable	Enable this configuration if your USB Device supports remote wakeup. This configuration is a part of the USB Configuration Descriptor (bmAttributes bit5).
Maximum Power Consumption (in 2mA units)	50	Set the maximum power consumption of your device to indicate the amount of bus power required. This configuration is 2mA units, thus, the maximum 500 mA can be specified. This configuration is a part of the USB Configuration Descriptor (bMaxPower).
Supported Language Code	0x0409	Specify the Language ID Code. For example, 0x0409 English - United States. This configuration is used for Language ID Framework code generation. See section "USBX Language Framework Configuration" for more information.
Name of USBX String Framework	NULL	Specify the name of user defined USBX String Framework. This must be a valid C symbol. Set NULL if the String Descriptor is not used. See section "USBX String Framework Configuration" for more information.
Total index number of USB String Descriptors in USB String Framework	0	Specify the total number of index for String Descriptor. See section "USBX String Framework Configuration" for more information.

ISDE Property	Value	Description
Name of USBX Language Framework	NULL	Specify the name of user defined USBX Language Framework. This must be a valid C symbol. If '0' is set to the property "Total Number of Language Support", this configuration is ignored. See section "USBX Language Framework Configuration" for more information.
Number of Languages to support (US English is applied if zero is set)	0	Specify the total number of languages to support. See section "USBX String Framework Configuration" for more information. If '0' is set here, US English (0x0409) is applied as the default language.
Name of generated initialization function	ux_device_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX インタフェース構成大容量記憶装置インスタンスの構成設定

ISDE Property	Value	Description
Name	g_usb_interface_descriptor_storage_0	Specify the name of USBX Interface Descriptor for CDC-ACM. It must be a valid C symbol.

ISDE Property	Value	Description
Interface Number of Bulk Only Data Interface	0x00	Specify the index number of Bulk-Only Data interface. This configuration is a part of the USB Interface Descriptor (bInterface). The number must not be duplicated with any other Interface Numbers if your USB device consists of a USB composite device.
Endpoint Number to be used for Bulk Out Transfer	Endpoint 1-9 Default: Endpoint 1	Specify the Endpoint Number of Bulk Out Endpoint. It must not be duplicated with ones for the other Endpoints.
Endpoint Number to be used for Bulk In Transfer	Endpoint 1-9 Default: Endpoint 2	Specify the Endpoint Number of Bulk In Endpoint. It must not be duplicated with ones for the other Endpoints.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBFS の sf_el_ux 上の USBX ポート DCD の構成設定

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for full-speed USB.
LDO Regulator (Only for S3 and S1 part MCUs)	Enable, Disable Default: Disable	Select the LDO regulator will be enabled.
Name	g_sf_el_ux_dcd_fs_0	Module name.
USB Controller Selection	USBFS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBHS 上の USBX ポート DCD の構成設定

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for high speed USB.
Name	g_sf_el_ux_dcd_hs_0	Module name.
USB Controller Selection	USBHS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ux 上の USBX インスタンスの構成設定

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	Name must be a valid C symbol.
USBX Pool Memory Size	18432	See section "Express Logic USBX Memory Requirements" for the required memory size for each classes.
User Callback for Host Event Notification (Only valid for USB Host)	NULL	Name must be a valid C symbol. The name of User defined USBX Host event notification can be given to this property.
Name of generated initialization function	ux_common_init0	Name of generated initialization function selection.

ISDE Property	Value	Description
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX デバイスクラス大容量記憶装置モジュールのクロック構成

USB ペリフェラルモジュールは UCLK をクロックソースとして使用します。UCLK は 48 MHz の動作に合わせて構成する必要があります。SSP 構成ウィンドウで、[Clocks] タブを選択してクロックソースの設定を確認します。

USBX デバイスクラス大容量記憶装置モジュールのピン構成

USB ペリフェラルモジュールは、MCU ピンを使用して外部デバイスと通信します。I/O ピンを選択し、外部デバイスの要件に合うように構成します。次の表では [SSP Configuration] ウィンドウでのピンの選択方法を示し、それ以降の表では USB ピンを例に使って選択プロセスを示します。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

USBFS および USBHS のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
USBFS	Pins	Select Peripherals > Connectivity: USBFS> USBFS0
USBHS	Pins	Select Peripherals > Connectivity: USBHS> USBHS0

注 : 選択シーケンスでは、USBFS0 または USBHS0 がドライバーに必要なハードウェアターゲットであることを想定しています。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select device as the Operation Mode

Property	Value	Description
USBDP	USBDP	USBDP pin
USBDM	USBDM	USBDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	None	VBUSEN pin
VBUS	None, P407 Default: P407	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID Pin
VCCUSB	VCCUSB	VCCUSB pin
VSSUSB	VSSUSB	VSSUSB pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select Device as the Operation Mode
USBHSDP	USBHSDP	USBHSDP pin
USBHSDM	USBHSDM	USBHSDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	PB00	VBUSEN pin

Property	Value	Description
VBUS	PB01	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID pin
USBHSRREF	USBHSRREF	USBHSRREF pin
AVCCUSBHS	AVCCUSBHS	AVCCUSBHS pin
AVSSUSBHS	AVSSUSBHS	AVSSUSBHS pin
PVSSUSBHS	PVSSUSBHS	PVSSUSBHS pin
VCCUSBHS	VCCUSBHS	VCCUSBHS pin
VSS1USBHS	VSS1USBHS	VSS1USBHS pin
VSS2USBHS	VSS2USBHS	VSS2USBHS pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.40.6 アプリケーションでの USBX デバイスクラス大容量記憶装置モジュールの使用

USBX デバイスクラス大容量記憶装置モジュールは、アプリケーションによる通常の初期化を必要としません。アプリケーションは、単純に、USBX デバイスクラス大容量記憶装置モジュールに必要な 3 つのユーザーコールバックを準備します。

4.3.41 USBX ホストクラス CDC-ACM

USBX ホストクラス CDC-ACM モジュールは、USBX ホストクラス CDC-ACM アプリケーション用のハイレベル API を提供し、USBX ホストクラス CDC-ACM ソース、USBX ホスト構成、USBX ソース、USBX ポート HCD、転送ドライバーを構成します。USBX ホストクラス CDC-ACM モジュールは、Synergy MCU 上の DMAC/DTC および USB ホストクラスペリフェラルを使用します。

4.3.41.1 USBX ホストクラス CDC-ACM モジュールの特長

CDC-ACM クラスは、複合デバイスフレームワークを使用して、インタフェースをグループ化します（制御とデータ）。そのため、デバイス記述子を定義する際は注意する必要があります。USBX は、インタフェースをバインドする方法を内部で判断するために、IAD 記述子を使用します。IAD 記述子は、インタフェースの前に宣言され、CDC-ACM クラスの最初のインタフェースと、アタッチされるインタフェースの数を含んでいる必要があります。CDC-ACM クラスは、新しい IAD 記述子と同じ機能を実行する共用体ファンクショナル記述子も使用します。共用体ファンクショナル記述子は、歴史的な経緯とホスト側との互換性のために宣言する必要がありますが、USBX では使用されません。

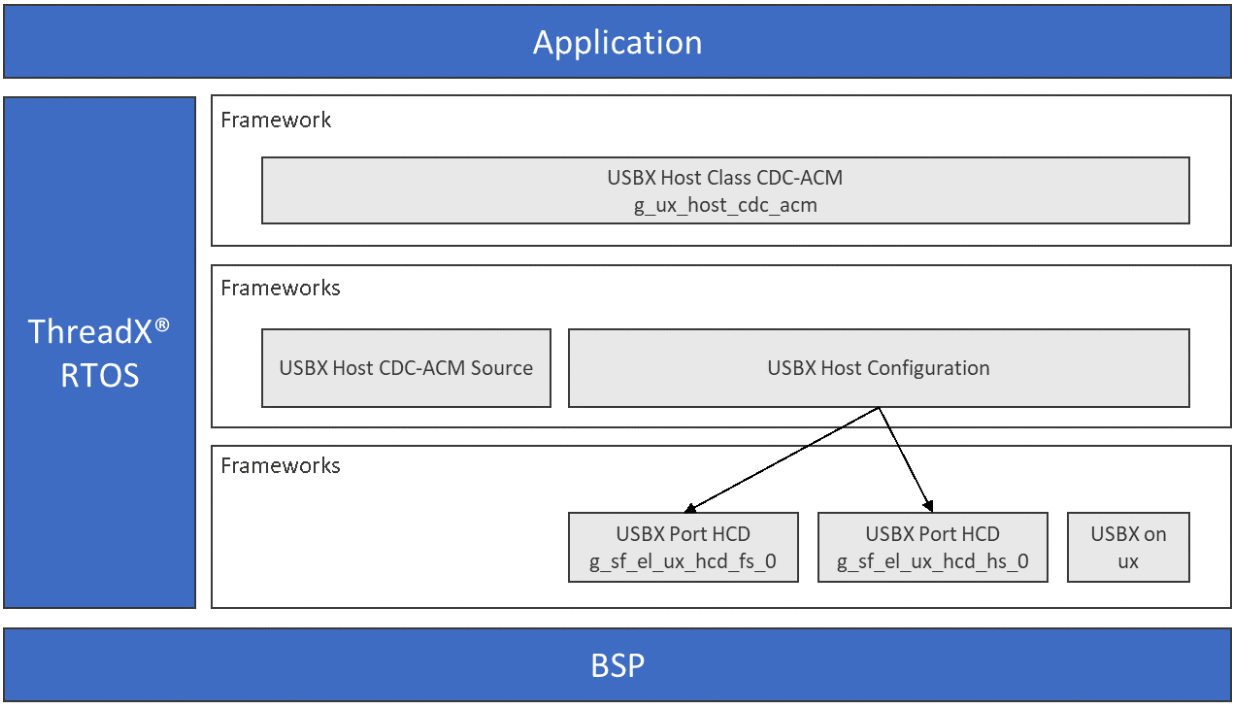


図 420:**USBX ホストクラス CDC-ACM モジュールのブロック図**

4.3.41.2 USBX ホストクラス CDC-ACM モジュールの API の概要

USBX ホストクラス CDC-ACM モジュールでは、リード、ライト、ioctl のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

USBX ホストクラス CDC-ACM モジュールの要約

Function Name	Example API Call and Description
ux_host_class_cdc_acm_read	status = ux_host_class_cdc_acm_read(cdc_acm, data_pointer, requested_length,&actual_length); This function reads from the cdc_acm interface. The call is blocking and only returns when there is either an error or when the transfer is complete.

Function Name	Example API Call and Description
ux_host_class_cdc_acm_write	status = ux_host_class_cdc_acm_write(cdc_acm, data_pointer,requested_length,&actual_length); This function writes to the cdc_acm interface. The call is blocking and only returns when there is either an error or when the transfer is complete.
ux_host_class_cdc_acm_ioctl	status = ux_host_class_cdc_acm_ioctl(cdc_acm, ioctl_function, &parameter_p); This function performs a specific ioctl function to the cdc_acm interface. The call is blocking and only returns when there is either an error or when the command is completed.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
UX_SUCCESS	The data transfer was completed.
UX_TRANSFER_TIMEOUT	Transfer timeout, reading/writing not completed.
UX_MEMORY_INSUFFICIENT	Not enough memory.
UX_HOST_CLASS_UNKNOWN	Wrong class instance.
UX_FUNCTION_NOT_SUPPORTED	Unknown IOCTL function.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.3.41.3 USBX ホストクラス CDC-ACM モジュールの動作の概要

USBX リソースの初期化

USBX には、独自のメモリマネージャがあります。メモリは、USBX のホスト側またはデバイス側が初期化される前に割り当てる必要があります。USBX メモリマネージャは、メモリのキャッシュが可能なシステムに対応できます。

USB ホストコントローラの定義

USBX をホストモードで動作させるには、USB ホストコントローラを 1 つ以上定義する必要があります。この定義は、アプリケーションの初期化ファイルに含める必要があります。SSP は、USB ホストコントローラドライバーがスレッドタスクに追加された場合に、USB ホストコントローラを定義します。

デバイスクラスの定義

USBX には、1 つ以上のデバイスクラスを定義する必要があります。USB クラスは、USB スタックが USB デバイスを構成した後に、USB デバイスを駆動するために必要です。USB クラスは、デバイスに固有です。USB デバイス記述子に含まれるインタフェースの数に応じて、USB を駆動するために 1 つ以上のクラスが必要になることがあります。

USB クラスのバインド

デバイスが構成されると、トポロジマネージャは、クラスマネージャがデバイスインタフェース記述子を調べ、デバイスの検出を続行することを許可します。1 つのデバイスに複数のインタフェース記述子が存在する場合があります。

インタフェースは、デバイスの機能を表します。たとえば、USB スピーカには、オーディオストリーミング用、オーディオ制御用、各種スピーカボタンの管理用の 3 つのインタフェースが存在します。

クラスマネージャには、デバイスインタフェースを 1 つ以上のクラスに結合する 2 つのメカニズムがあります。1 つはインタフェース記述子に含まれる PID と VID（製品 ID とベンダ ID）の組み合わせを使用する方法、もう 1 つはクラス、サブクラス、プロトコルの組み合わせを使用する方法です。

PID と VID の組み合わせは、一般的なクラスでは駆動できないインタフェースに有効です。クラス、サブクラス、プロトコルの組み合わせは、プリンタ、ハブ、ストレージ、オーディオ、ヒューマンインタフェースデザイン（HID）など、USB-IF が認定したクラスに属するインタフェースに使用されます。

クラスマネージャには、USBX の初期化で登録されたクラスのリストが格納されます。クラスマネージャは、デバイスのインタフェースを管理するクラスが決まるまでクラスを 1 つずつ呼び出します。1 つのクラスが管理できるインタフェースは、1 つだけです。USB オーディオスピーカの場合は、クラスマネージャは、各インタフェースに対してすべてのクラスを呼び出します。

クラスがインタフェースを受け入れると、そのクラスの新しいインスタンスが作成され、クラスマネージャがインタフェースのデフォルトの代替設定を検索します。1 つのデバイスの各インタフェースには、1 つ以上の代替設定が存在する場合があります。クラスで変更されない限り、デフォルトで代替設定 0 が使用されます。

デフォルトの代替設定の場合は、クラスマネージャが代替設定に含まれるすべてのエンドポイントをマウントします。各エンドポイントのマウントが成功すると、インタフェースの初期化を完了するクラスにクラスマネージャが戻ってジョブを完了します。

USBX ホストクラス CDC-ACM モジュールの動作に関する重要な注意事項と制限事項

USBX デバイススタックまたは USBX ホストスタックは、制御ブロックの RAM を消費します。Synergy 構成ツールは、次の表に示すように、自動生成コードでメモリを USBX メモリプールに静的に割り当てます。[USBX on ux Configuration] セクションの Synergy 構成ツールで ux 上の USBX コンポーネントの [USBX Pool Memory Size] プロパティに、適切なメモリサイズ（バイト単位）を設定する必要があります。クラスを複数使用する場合は、このプロパティに合計メモリサイズを設定してください。

USBX メモリプールのメモリ（RAM）要件

USBX Class	S1 Parts	Other Parts
USBX Host CDC-ACM (ux_host_class_cdc_acm)	N/A	30 KB

注 : 上の表に示す情報は、デフォルトの USBX 構成でコンパイルした場合のものです。

*注 : 上の表に示す USBX クラスのメモリサイズは、ビルド済みライブラリのものであり、ビルドには以下の構成が適用されています。*UX_THREAD_STACK_SIZE:S1 パーツには 512（バイト）、その他のパーツには 2048（バイト）。

- CDC-ACM インスタンスを取得するために、自動生成コードによって取得される USBX ホストクラス CDC-ACM 用のクラスコンテナを使用します。
- ux_host_class_cdc_acm_state フラグをインスタンスでポーリングし、ステータスがライブであることを確認します。
- CDC-ACM インスタンスでデータクラスインタフェースが使用可能であるかどうかを確認します。
- 必要に応じて、CDC-ACM の受信を設定します。
- USB デバイスとの CDC-ACM 通信を実行します。
- このモジュールでは、USB コントローラの割り込みが有効になっている必要があります。
- 既知の制限事項については、SSP のリリースノート（sf_el_ux のセクション）を参照してください。
- このモジュールは、USB コントローラの割り込みを使用します。Synergy 構成ツールで適切な割り込み優先順位レベルを設定してください。デフォルトでは、割り込みが無効になっています。
- このモジュールが使用されると、転送モジュール（DMAC または DTC として実装）の割り込みが使用されます。Synergy 構成ツールで適切な優先順位レベルを設定してください。適切に動作するために、レベルは USB コントローラより高くする必要があります。
- 長さがゼロのパケットには受信データが含まれていませんが、アプリケーションは受信操作を実行する必要があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.41.4 アプリケーションへの USBX ホストクラス CDC-ACM モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに USBX ホストクラス CDC-ACM モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

USBX ホストクラス CDC-ACM モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

USBX ホストクラス CDC-ACM モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ux_device_class_cdc_acm0 USBX Host Class CDC-ACM	Threads	New Stack> X-Ware> USBX> Device> Classes > CDC-ACM > USBX Host Class CDC-ACM

次の図に示すように、USBX ホストクラス CDC-ACM モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

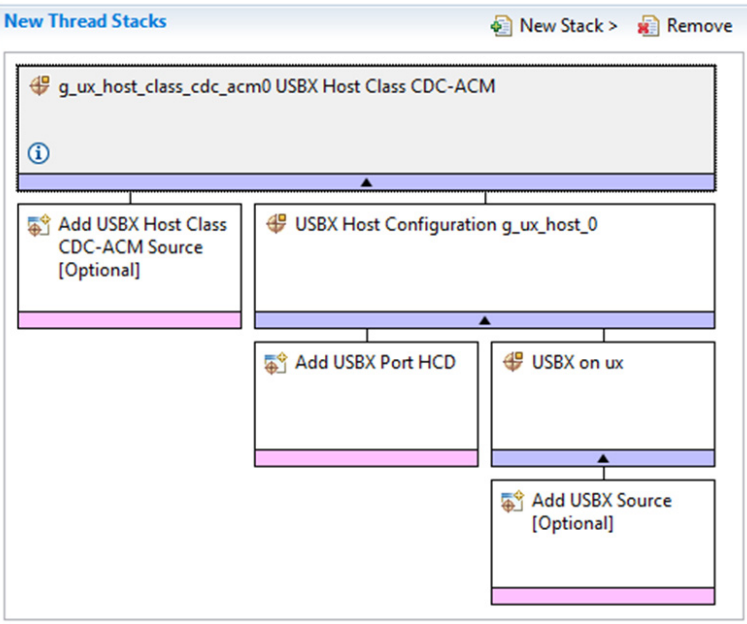


図 421:USBX ホストクラス CDC-ACM モジュールのスタック

4.3.41.5 USBX ホストクラス CDC-ACM モジュールの構成

ユーザーは必要な動作に合わせて USBX ホストクラス CDC-ACM モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できませ

ん。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

USBX ホストクラス CDC-ACM モジュールの構成設定

ISDE Property	Value	Description
Name	g_ux_host_class_cdc_acm0	Specify the name of USBX Host CDC-ACM Class module instance. It must be a valid C symbol.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

USBX ホストクラス CDC-ACM のローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

USBX ホストクラス CDC-ACM ソースの構成設定

ISDE Property	Value	Description
Show linkage warning	Enabled, Disabled Default: Enabled	Notification message for users will be shown if "Enabled" option is selected. This is just to warn users possible linkage errors by multiple symbol definitions. Select "Disabled" stops the notification message.

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX ホスト構成インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ux_host0	Specify the name of USBX Host Configuration instance. It must be a valid C symbol.
Name of generated initialization function	ux_host_hid_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBFS の sf_el_ux 上の USBX ポート HCD の構成設定

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for full-speed USB.
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	Select the VBUSEN pin signal logic.
LDO Regulator (Only for S3 and S1 part MCUs)	Enable, Disable Default: Disable	Select the LDO regulator will be enabled.
Name	g_sf_el_ux_hcd_fs_0	Module name.
USB Controller Selection	USBFS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBHS の sf_el_ux 上の USBX ポート HCD の構成設定

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for high speed USB.
FIFO size for Bulk/Isochronous Pipes	512, 1024, 1536, 2048 bytes Default: 512 bytes	Select the FIFO size for bulk and isochronous transfers.
Number of Isochronous Pipes Reserved	0,1,2 Default: 0	Select the number of isochronous pipes to reserve.
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	Select the VBUSEN pin signal logic.
Enable High Speed	Enable, Disable Default: Enable	Select if high speed should be enabled.
Name	g_sf_el_ux_hcd_hs_0	Module name.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ux 上の USBX インスタンスの構成設定

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	Name must be a valid C symbol.

ISDE Property	Value	Description
USBX Pool Memory Size	18432	See section "Express Logic USBX Memory Requirements" for the required memory size for each classes.
User Callback for Host Event Notification (Only valid for USB Host)	NULL	Name must be a valid C symbol. The name of User defined USBX Host event notification can be given to this property.
Name of generated initialization function	ux_common_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX ホストクラス CDC-ACM モジュールのクロック構成

USB ペリフェラルモジュールは UCLK をクロックソースとして使用します。UCLK は 48 MHz の動作に合わせて構成する必要があります。SSP 構成ウィンドウで、[Clocks] タブを選択してクロックソースの設定を確認します。

USBX ホストクラス CDC-ACM モジュールのピン構成

USB ペリフェラルモジュールは、MCU ピンを使用して外部デバイスと通信します。I/O ピンを選択し、外部デバイスの要件に合うように構成します。次の表では [SSP Configuration] ウィンドウでのピンの選択方法を示し、それ以降の表では USB ピンを例に使って選択プロセスを示します。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

USBFS および USBHS のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
USBFS	Pins	Select Peripherals > Connectivity: USBFS> USBFS0
USBHS	Pins	Select Peripherals > Connectivity: USBHS> USBHS0

注 : 選択シーケンスでは、USBFS0 または USBHS0 がドライバーに必要なハードウェアターゲットであることを想定しています。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select device as the Operation Mode
USBDP	USBDP	USBDP pin
USBDM	USBDM	USBDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	None	VBUSEN pin
VBUS	None, P407 Default: P407	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID Pin
VCCUSB	VCCUSB	VCCUSB pin
VSSUSB	VSSUSB	VSSUSB pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select Device as the Operation Mode
USBHSDP	USBHSDP	USBHSDP pin
USBHSDM	USBHSDM	USBHSDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	PB00	VBUSEN pin
VBUS	PB01	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID pin
USBHSRREF	USBHSRREF	USBHSRREF pin
AVCCUSBHS	AVCCUSBHS	AVCCUSBHS pin
AVSSUSBHS	AVSSUSBHS	AVSSUSBHS pin
PVSSUSBHS	PVSSUSBHS	PVSSUSBHS pin
VCCUSBHS	VCCUSBHS	VCCUSBHS pin
VSS1USBHS	VSS1USBHS	VSS1USBHS pin
VSS2USBHS	VSS2USBHS	VSS2USBHS pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.41.6 アプリケーションでの USBX ホストクラス CDC-ACM モジュールの使用

コンフィギュレータは、USBX ホストクラス CDC-ACM モジュールの作成と登録に必要な処理を生成しますが、通信はホストがホストに接続されてから行う必要があります。

アプリケーションで USBX ホストクラス CDC-ACM モジュールを使用する際の一般的な手順は次のとおりです。

1) ux_host_stack_class_instance_get API を使用して、接続されたデバイスの最初のインスタンスを取得します。
2) デバイスのステータスがライブになるまで待機します。
3) デバイスのクラスが CDC データクラスであることを確認します。
4) 次のデバイスがある場合は、再びステータスを確認します。
5) 受信データのリードには、ux_host_class_cdc_acm_read API を使用します。
6) データの送信には、ux_host_class_cdc_acm_write API を使用します。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

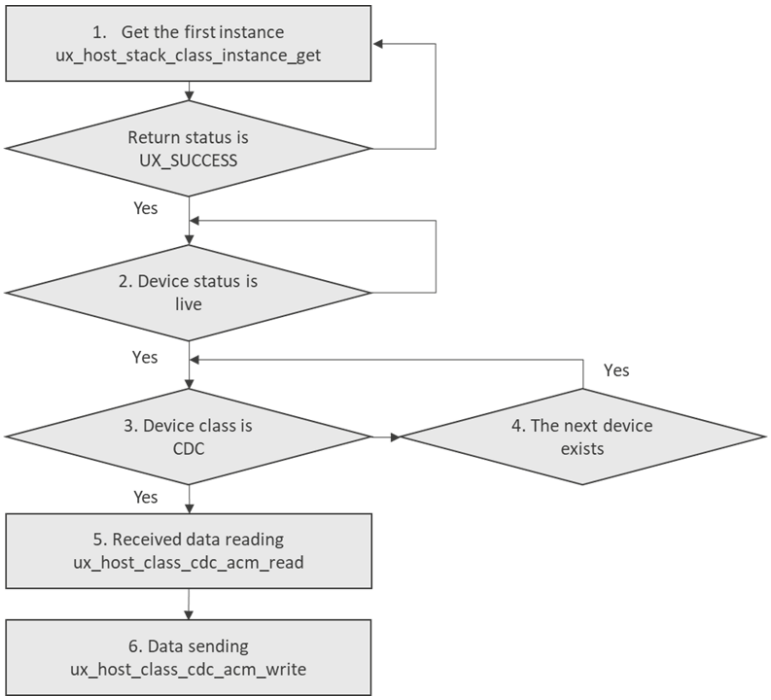


図 422:一般的な USBX ホストクラス CDC-ACM モジュールアプリケーションのフロー図

4.3.42 USBX ホストクラス HID

USBX ™ホストクラス HID モジュールは、ヒューマンインタフェースデバイス（HID）アプリケーション用のハイレベル API を提供し、USBX ホストクラス HID ソース、USBX ホスト構成、USBX ソース、USBX ポート HCD を構成します。USBX ホストクラス HID モジュールは、Synergy MCU 上の USB ペリフェラルを使用します。

4.3.42.1 USBX ホストクラス HID モジュールの特長

USBX ホストクラスヒューマンインタフェースデバイス（HID）モジュールは、USBX HID クラスをサポートします。以下の特長があります。

- HID レポートのデータ転送をサポートします。
- マルチインタフェース構成をサポートします。
- 以下のクライアントをサポートします。
  - キーボード
  - マウス
  - リモートコントロール

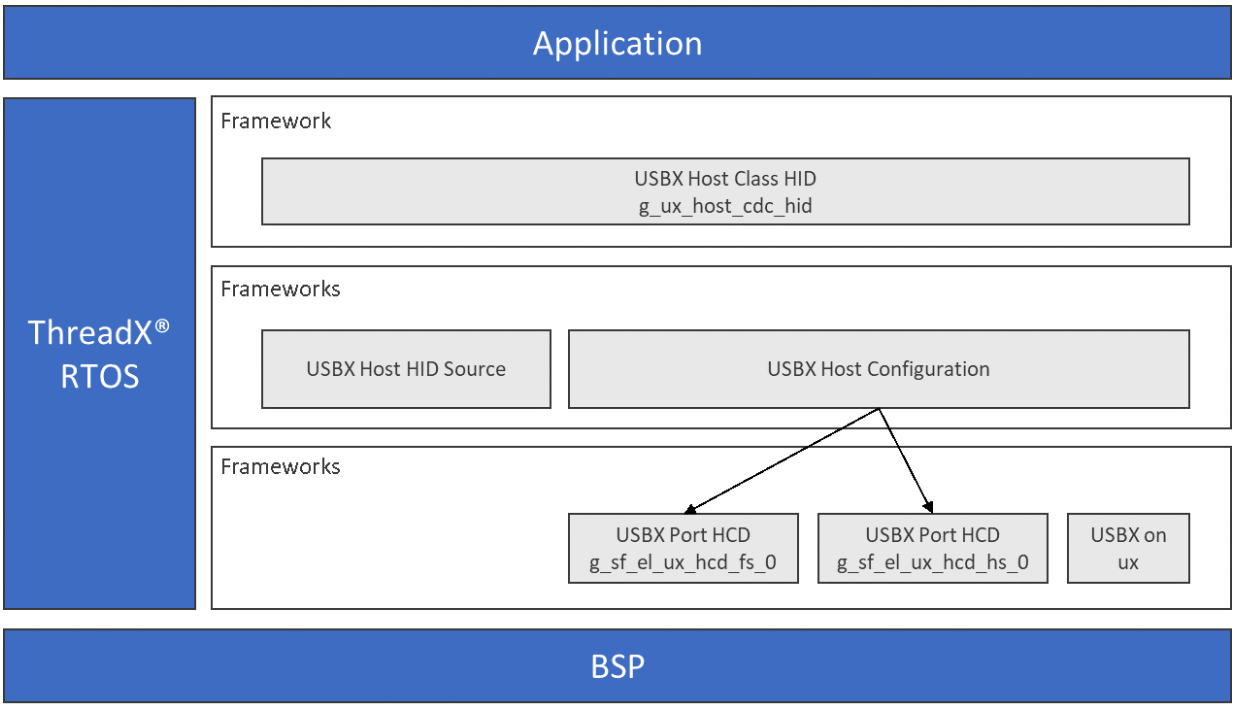


図 423:**USBX ホストクラス HID モジュールのブロック図**

4.3.42.2 USBX ホストクラス HID モジュールの API の概要

USBX ホストクラス HID モジュールでは、コールバックの登録、周期レポートの開始と停止、レポートの取得、レポートの取得と設定のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

USBX ホストクラス HID モジュールの API の要約

Function Name	Example API Call and Description
ux_host_class_hid_report_callback_register	ux_host_class_hid_report_callback_register (hid, &call_back); This function is used to register a callback from the HID class to the HID client when a report is received.
ux_host_class_hid_periodic_report_start	ux_host_class_hid_periodic_report_start(hid); This function is used to start the periodic (interrupt) endpoint for the instance of the HID class that is bound to this HID client.
ux_host_class_hid_periodic_report_stop	ux_host_class_hid_periodic_report_stop(hid); This function is used to stop the periodic (interrupt) endpoint for the instance of the HID class that is bound to this HID client.
ux_host_class_hid_report_get	ux_host_class_hid_report_get(hid, &client_report); This function is used to receive a report directly from the device without relying on the periodic endpoint.
ux_host_class_hid_report_set	ux_host_class_hid_report_set(hid, &client_report); This function is used to send a report directly to the device.
ux_host_class_hid_keyboard_key_get	ux_host_class_hid_keyboard_key_get(keyb oard_instance, &keyboard_char, &keyboard_state); This function is used to read the keyboard data received from the device.

Function Name	Example API Call and Description
ux_host_class_hid_mouse_buttons_get	ux_host_class_hid_mouse_buttons_get(&mouse_instance, &mouse_buttons); This function is used to read the mouse button information received from the device.
ux_host_class_hid_mouse_position_get	ux_host_class_hid_mouse_position_get(mouse_instance, &x_position, &y_position); This function is used to read the position information of the mouse received from the device.
ux_host_class_hid_remote_control_usage_get	ux_host_class_hid_remote_control_usage_get(remote_control_instance, &usage, &value); This function is used to read the remote controller information received from the device.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
UX_SUCCESS	The data transfer was completed.
UX_TRANSFER_TIMEOUT	Transfer timeout, reading/writing not completed.
UX_MEMORY_INSUFFICIENT	Not enough memory.
UX_HOST_CLASS_UNKNOWN	Wrong class instance.
UX_FUNCTION_NOT_SUPPORTED	Unknown IOCTL function.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.3.42.3 USBX ホストクラス HID モジュールの動作の概要

USBX リソースの初期化

USBX には、独自のメモリマネージャがあります。メモリは、USBX のホスト側またはデバイス側が初期化される前に割り当てる必要があります。USBX メモリマネージャは、メモリのキャッシュが可能なシステムに対応できます。

USB ホストコントローラの定義

USBX をホストモードで動作させるには、USB ホストコントローラを 1 つ以上定義する必要があります。この定義は、アプリケーションの初期化ファイルに含める必要があります。SSP は、USB ホストコントローラドライバーがスレッドタスクに追加された場合に、USB ホストコントローラを定義します。

デバイスクラスの定義

USBX には、1 つ以上のデバイスクラスを定義する必要があります。USB クラスは、USB スタックが USB デバイスを構成した後に、USB デバイスを駆動するために必要です。USB クラスは、デバイスに固有です。USB デバイス記述子に含まれるインタフェースの数に応じて、USB を駆動するために 1 つ以上のクラスが必要になることがあります。

USB クラスのバインド

デバイスが構成されると、トポロジマネージャは、クラスマネージャがデバイスインタフェース記述子を調べ、デバイスの検出を続行することを許可します。1 つのデバイスに複数のインタフェース記述子が存在する場合があります。

インタフェースは、デバイスの機能を表します。たとえば、USB スピーカには、オーディオストリーミング用、オーディオ制御用、各種スピーカボタンの管理用の 3 つのインタフェースが存在します。

クラスマネージャには、デバイスインタフェースを 1 つ以上のクラスに結合する 2 つのメカニズムがあります。1 つはインタフェース記述子に含まれる PID と VID（製品 ID とベンダ ID）の組み合わせを使用する方法、もう 1 つはクラス、サブクラス、プロトコルの組み合わせを使用する方法です。

PID と VID の組み合わせは、一般的なクラスでは駆動できないインタフェースに有効です。クラス、サブクラス、プロトコルの組み合わせは、プリンタ、ハブ、ストレージ、オーディオ、ヒューマンインタフェースデザイン（HID）など、USB-IF が認定したクラスに属するインタフェースに使用されます。

クラスマネージャには、USBX の初期化で登録されたクラスのリストが格納されます。クラスマネージャは、デバイスのインタフェースを管理するクラスが決まるまでクラスを 1 つずつ呼び出します。1 つのクラスが管理できるインタフェースは、1 つだけです。USB オーディオスピーカの場合は、クラスマネージャは、各インタフェースに対してすべてのクラスを呼び出します。

クラスがインタフェースを受け入れると、そのクラスの新しいインスタンスが作成され、クラスマネージャがインタフェースのデフォルトの代替設定を検索します。1 つのデバイスの各インタフェースには、1 つ以上の代替設定が存在する場合があります。クラスで変更されない限り、デフォルトで代替設定 0 が使用されます。

デフォルトの代替設定の場合は、クラスマネージャが代替設定に含まれるすべてのエンドポイントをマウントします。各エンドポイントのマウントが成功すると、インタフェースの初期化を完了するクラスにクラスマネージャが戻ってジョブを完了します。

USBX ホストクラス HID モジュールの動作に関する重要な注意事項と制限事項

USBX デバイススタックまたは USBX ホストスタックは、制御ブロックの RAM を消費します。Synergy 構成ツールは、次の表に示すように、自動生成コードでメモリを USBX メモリプールに静的に割り当てます。[USBX on ux Configuration] セクションの Synergy 構成ツールで ux 上の USBX コンポーネントの [USBX Pool Memory Size] プロパティに、適切なメモリサイズ（バイト単位）を設定する必要があります。クラスを複数使用する場合は、このプロパティに合計メモリサイズを設定してください。

USBX メモリプールのメモリ（RAM）要件

USBX Class	S1 Parts	Other Parts
USBX Host HID (ux_host_class_hid)	N/A	HID Mouse: 38 KB HID Keyboard: 46 KB

上の表に記載されている USBX ホスト HID のメモリサイズには、以下の構成（デフォルト設定）が適用されています。ソースモジュールを使用する場合は、サイズを縮小することができます。

- UX_HOST_CLASS_HID_DECOMPRESSION_BUFFER: 4096（バイト）
- UX_HOST_CLASS_HID_USAGE: 1024（ワード）

注 : 上の表に示す情報は、デフォルトの USBX 構成でコンパイルした場合のものです。

注 : 上の表に示す USBX クラスのメモリサイズは、ビルド済みライブラリのものであり、ビルドには以下の構成が適用されています。

UX_THREAD_STACK_SIZE:S1 パーツには 512（バイト）、その他のパーツには 2048（バイト）。

- デフォルトでは、このモジュールはキーボード、マウス、リモートコントロールクライアントをサポートします。
- HID インスタンスを取得するために、自動生成コードによって取得される USBX ホストクラス HID 用のクラスコンテナを使用します。
- ux_host_class_hid_state, フラグをインスタンスでポーリングし、ステータスがライブであることを確認します。
- クライアントのローカルインスタンスが HID インスタンスで使用可能であるかどうかを確認します。
- USB デバイスとの HID 通信を実行します。
- このモジュールでは、USB コントローラの割り込みを使用するため、割り込みが有効になっている必要があります。適切な動作を確保するために、Synergy 構成ツールで適切な割り込み優先順位レベルを設定してください。
- 転送モジュールが使用されると、割り込み（DMAC または DTC として実装）が使用されます。Synergy 構成ツールで優先順位レベルを設定してください。レベルは USB コントローラより高くする必要があります。そうしないと機能しません。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.42.4 アプリケーションへの USBX ホストクラス HID モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに USBX ホストクラス HID モジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

USBX ホストクラス HID モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

USBX ホストクラス HID モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ux_host_class_hid0 USBX Host Class HID	Threads	New Stack > X-Ware > USBX > Host > Classes > HID > USBX Host Class HID

次の図に示すように、USBX ホストクラス HID モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

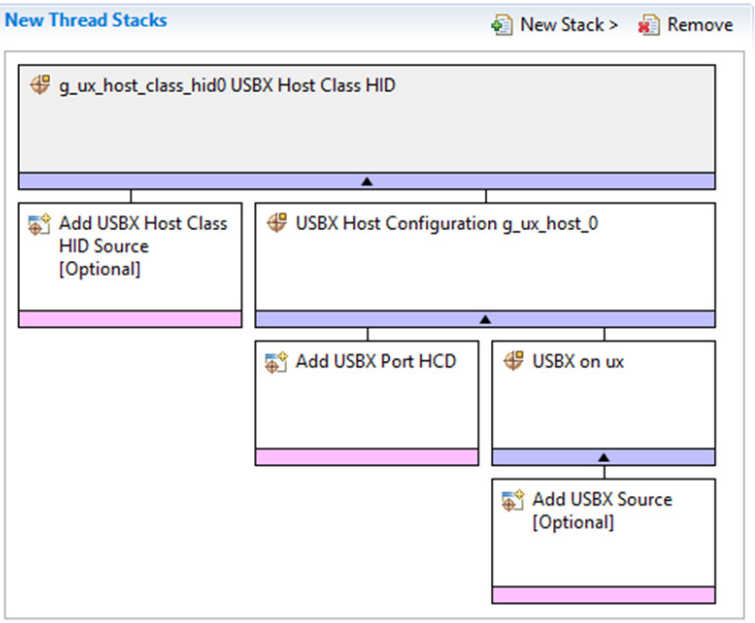


図 424:USBX ホストクラス HID モジュールのスタック

4.3.42.5 USBX ホストクラス HID モジュールの構成

ユーザーは必要な動作に合わせてUSBXホストクラスHIDモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注：次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

USBX ホストクラス HID モジュールの構成設定

ISDE Property	Value	Description
Name	g_ux_host_class_hid0	Specify the name of USBX Host Mass HID module instance. It must be a valid C symbol.
HID Client - Keyboard Support	Enable, Disable Default: Enable	Set Enable to support USB keyboard devices. This configuration registers the USBX HID Keyboard Client.
HID Client - Mouse Support	Enable, Disable Default: Enable	Set Enable to support USB mouse devices. This configuration registers the USBX HID Mouse Client.
HID Client - Remote Control Support	Enable, Disable Default: Enable	Set Enable to support USB remote control devices. This configuration registers the USBX Remote Control Client.

注：設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータから対応する [Properties] ウィンドウを調べることで決定されます。

USBX ホストクラス HID のローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動する

ために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

USBX ホストクラス HID ソースの構成設定

ISDE Property	Value	Description
HID Keyboard Thread Priority	Value must be greater than 0 or empty Default: 20	UX_THREAD_PRIORITY_KEYBOARD Define the priority of the HID keyboard thread.
Memory size for HID Report Decompression	Value must be greater than 0 or empty Default: 4096	UX_HOST_CLASS_HID_DECOMPRESSION_BUFFER Define the memory size to build a decompressed report
Number of Entries for HID Local Usage Item Table	Value must be greater than 0 or empty Default: 1024	UX_HOST_CLASS_HID_USAGES Define the size of HID local usage item table. One item entry consumes 4 bytes.
Show linkage warning	Enabled, Disabled Default: Enabled	Notification message for users will be shown if “Enabled" option is selected. This is just to warn users possible linkage errors by multiple symbol definitions. Select “Disabled” stops the notification message.

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX ホスト構成インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ux_host0	Specify the name of USBX Host Configuration instance. It must be a valid C symbol.
Name of generated initialization function	ux_host_hid_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBFS の sf_el_ux 上の USBX ポート HCD の構成設定

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for full-speed USB.
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	Select the VBUSEN pin signal logic.
LDO Regulator (Only for S3 and S1 part MCUs)	Enable, Disable Default: Disable	Select the LDO regulator will be enabled.
Name	g_sf_el_ux_hcd_fs_0	Module name.
USB Controller Selection	USBFS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBHS の sf_el_ux 上の USBX ポート HCD の構成設定

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for high speed USB.
FIFO size for Bulk/Isochronous Pipes	512, 1024, 1536, 2048 bytes Default: 512 bytes	Select the FIFO size for bulk and isochronous transfers.
Number of Isochronous Pipes Reserved	0,1,2 Default: 0	Select the number of isochronous pipes to reserve.
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	Select the VBUSEN pin signal logic.
Enable High Speed	Enable, Disable Default: Enable	Select if high speed should be enabled.
Name	g_sf_el_ux_hcd_hs_0	Module name.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ux 上の USBX インスタンスの構成設定

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	Name must be a valid C symbol.

ISDE Property	Value	Description
USBX Pool Memory Size	18432	See section "Express Logic USBX Memory Requirements" for the required memory size for each classes.
User Callback for Host Event Notification (Only valid for USB Host)	NULL	Name must be a valid C symbol. The name of User defined USBX Host event notification can be given to this property.
Name of generated initialization function	ux_common_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX ホストクラス HID モジュールのクロック構成

USB ペリフェラルモジュールは UCLK をクロックソースとして使用します。UCLK は 48 MHz の動作に合わせて構成する必要があります。SSP 構成ウィンドウで、[Clocks] タブを選択してクロックソースの設定を確認します。

USBX ホストクラス HID モジュールのピン構成

USB ペリフェラルモジュールは、MCU ピンを使用して外部デバイスと通信します。I/O ピンを選択し、外部デバイスの要件に合うように構成します。次の表では [SSP Configuration] ウィンドウでのピンの選択方法を示し、それ以降の表では USB ピンを例に使って選択プロセスを示します。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

USBFS および USBHS のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
USBFS	Pins	Select Peripherals > Connectivity: USBFS > USBFS0

Resource	ISDE Tab	Pin selection Sequence
USBHS	Pins	Select Peripherals > Connectivity: USBHS > USBHS0

注 : 選択シーケンスでは、USBFS0 または USBHS0 がドライバーに必要なハードウェアターゲットであることを想定しています。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select device as the Operation Mode
USBDP	USBDP	USBDP pin
USBDM	USBDM	USBDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	None	VBUSEN pin
VBUS	None, P407 Default: P407	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID Pin
VCCUSB	VCCUSB	VCCUSB pin
VSSUSB	VSSUSB	VSSUSB pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select Device as the Operation Mode
USBHSDP	USBHSDP	USBHSDP pin
USBHSDM	USBHSDM	USBHSDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	PB00	VBUSEN pin
VBUS	PB01	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID pin
USBHSRREF	USBHSRREF	USBHSRREF pin
AVCCUSBHS	AVCCUSBHS	AVCCUSBHS pin
AVSSUSBHS	AVSSUSBHS	AVSSUSBHS pin
PVSSUSBHS	PVSSUSBHS	PVSSUSBHS pin
VCCUSBHS	VCCUSBHS	VCCUSBHS pin
VSS1USBHS	VSS1USBHS	VSS1USBHS pin
VSS2USBHS	VSS2USBHS	VSS2USBHS pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.42.6 アプリケーションでの USBX ホストクラス HID モジュールの使用

コンフィギュレータは、USBX ホストクラス HID モジュールの作成と登録に必要な処理を生成しますが、通信はホストがホストに接続されてから行う必要があります。

アプリケーションで USBX ホストクラス HID モジュールを使用する際の一般的な手順は次のとおりです。

1) ux_host_stack_class_instance_get API を使用して、接続されたデバイスの最初のインスタンスを取得します。
2) ux_success. を待機します。
3) デバイスのステータスがライブになるまで待機します。
4) クライアントインスタンスがライブになるまで待機します。
5) デバイスがキーボードの場合
6) ux_host_class_hid_keyboard_key_get API を使用してキーボードデータを受信します。
7) デバイスがマウスの場合
8) ux_host_class_hid_mouse_buttons_get API と ux_host_class_hid_mouse_position_get API を使用して、マウスデータを受信します。
9) デバイスがリモートコントローラの場合
10) ux_host_class_hid_remote_control_usage_get API を使用してリモートコントローラデータを受信します。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

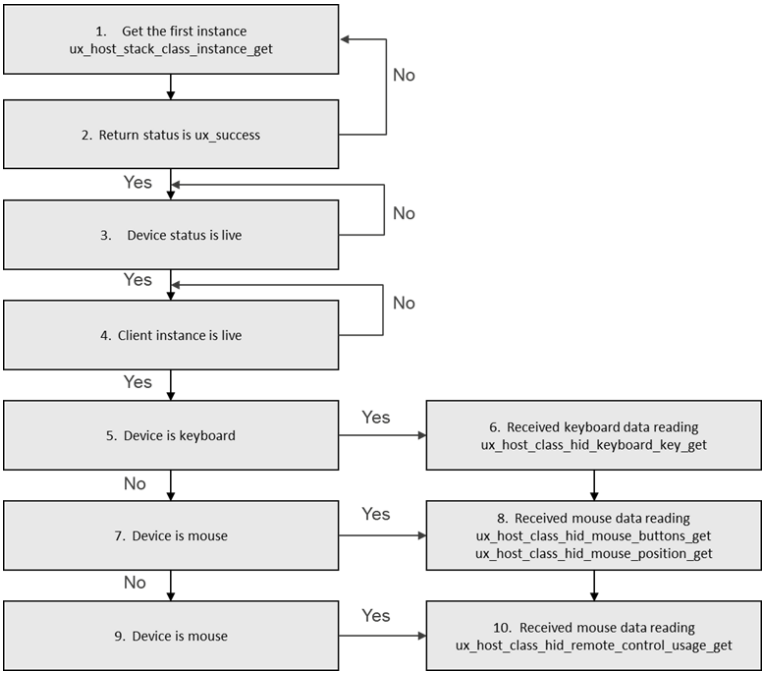


図 425:一般的な USBX ホストクラス HID モジュールアプリケーションのフロー図

4.3.43 USBX ホストクラスハブ

USBX ™ホストクラスハブモジュールは、USBX ホストクラスハブアプリケーション用のハイレベル API を提供し、USBX ホストクラスハブソース、USBX ホスト構成、USBX ソース、USBX ポートホストコントローラデバイスを構成します。USBX ホストクラスハブモジュールは、Synergy MCU 上の USB ペリフェラルを使用します。

4.3.43.1 USBX ホストクラスハブモジュールの特長

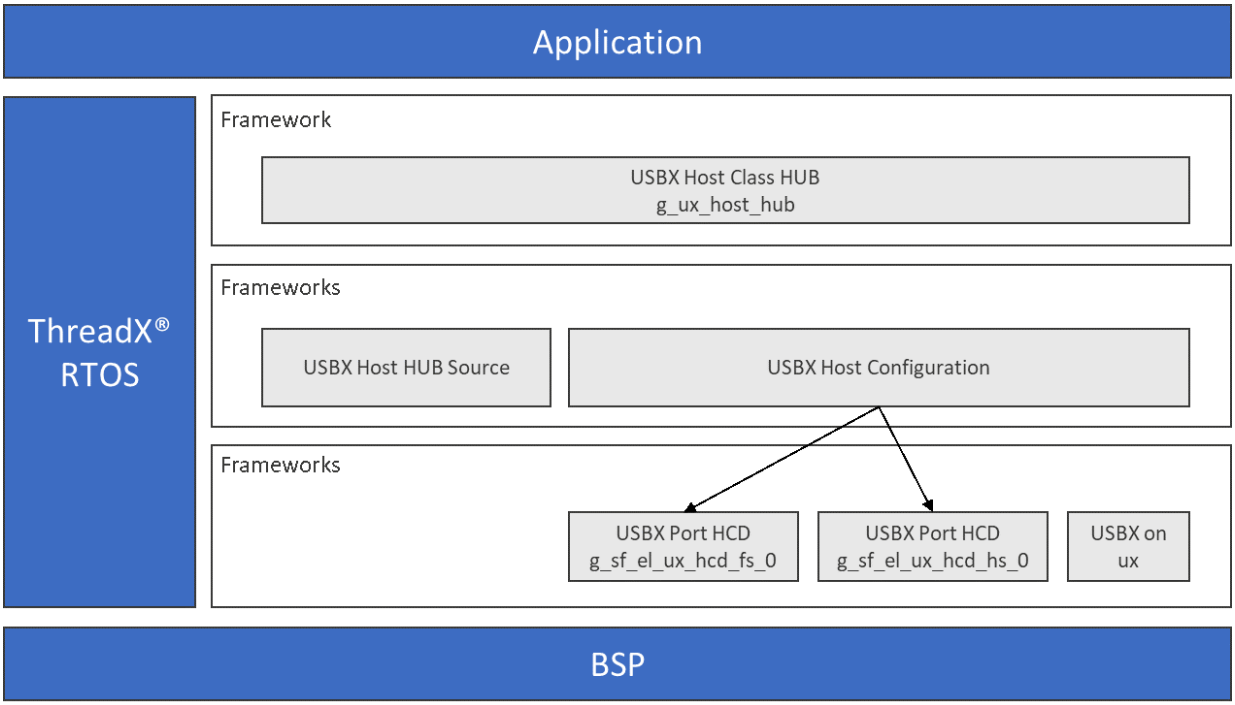


図 426:**USBX ホストクラスハブモジュールのブロック図**

4.3.43.2 USBX ホストクラスハブモジュールの API の概要

USBX ホストクラスハブモジュールには、ユーザーアプリケーション用の API はありません。

4.3.43.3 USBX ホストクラスハブモジュールの動作の概要

USBX リソースの初期化

USBX には、独自のメモリマネージャがあります。メモリは、USBX のホスト側またはデバイス側が初期化される前に割り当てる必要があります。USBX メモリマネージャは、メモリのキャッシュが可能なシステムに対応できます。

USB ホストコントローラの定義

USBX をホストモードで動作させるには、USB ホストコントローラを 1 つ以上定義する必要があります。この定義は、アプリケーションの初期化ファイルに含める必要があります。SSP は、USB ホストコントローラドライバーがスレッドタスクに追加された場合に、USB ホストコントローラを定義します。

デバイスクラスの定義

USBX には、1 つ以上のデバイスクラスを定義する必要があります。USB クラスは、USB スタックが USB デバイスを構成した後に、USB デバイスを駆動するために必要です。USB クラスは、デバイスに固有です。USB デバイス記述子に含まれるインタフェースの数に応じて、USB を駆動するために 1 つ以上のクラスが必要になることがあります。

USB クラスのバインド

デバイスが構成されると、トポロジマネージャは、クラスマネージャがデバイスインタフェース記述子を調べ、デバイスの検出を続行することを許可します。1 つのデバイスに複数のインタフェース記述子が存在する場合があります。

インタフェースは、デバイスの機能を表します。たとえば、USB スピーカには、オーディオストリーミング用、オーディオ制御用、各種スピーカボタンの管理用の 3 つのインタフェースが存在します。

クラスマネージャには、デバイスインタフェースを 1 つ以上のクラスに結合する 2 つのメカニズムがあります。1 つはインタフェース記述子に含まれる PID と VID（製品 ID とベンダ ID）の組み合わせを使用する方法、もう 1 つはクラス、サブクラス、プロトコルの組み合わせを使用する方法です。

PID と VID の組み合わせは、一般的なクラスでは駆動できないインタフェースに有効です。クラス、サブクラス、プロトコルの組み合わせは、プリンタ、ハブ、ストレージ、オーディオ、ヒューマンインタフェースデザイン（HID）など、USB-IF が認定したクラスに属するインタフェースに使用されます。

クラスマネージャには、USBX の初期化で登録されたクラスのリストが格納されます。クラスマネージャは、デバイスのインタフェースを管理するクラスが決まるまでクラスを 1 つずつ呼び出します。1 つのクラスが管理できるインタフェースは、1 つだけです。USB オーディオスピーカの場合は、クラスマネージャは、各インタフェースに対してすべてのクラスを呼び出します。

クラスがインタフェースを受け入れると、そのクラスの新しいインスタンスが作成され、クラスマネージャがインタフェースのデフォルトの代替設定を検索します。1 つのデバイスの各インタフェースには、1 つ以上の代替設定が存在する場合があります。クラスで変更されない限り、デフォルトで代替設定 0 が使用されます。

デフォルトの代替設定の場合は、クラスマネージャが代替設定に含まれるすべてのエンドポイントをマウントします。各エンドポイントのマウントが成功すると、インタフェースの初期化を完了するクラスにクラスマネージャが戻ってジョブを完了します。

USBX ホストクラスハブモジュールの動作に関する重要な注意事項と制限事項

ハブクラスは、自動生成コードによって登録されます。ユーザーアプリケーションでは、ハブクラスの登録以外に特別な動作は必要ありません。

- このモジュールでは、USB コントローラの割り込みが有効になっている必要があります。
- このモジュールは、USB コントローラの割り込みを使用します。適切な動作を確保するために、Synergy 構成ツールで適切な割り込み優先順位レベルを設定してください。
- 転送モジュールが実装されると、そのモジュール（DMAC または DTC として実装）の割り込みが使用されます。Synergy 構成ツールで適切なプライオリティレベルを設定してください。適切な動作を確保するために、優先順位レベルは USB コントローラより高くする必要があります。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.43.4 アプリケーションへの USBX ホストクラスハブモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに USBX ホストクラスハブモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

USBX ホストクラスハブモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

USBX ホストクラスハブモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ux_host_class_hub0 USBX Host Class Hub	Threads	New Stack> X-Ware™> USBX> Host > Classes > Hub > USBX Host Class Hub

次の図に示すように、USBX ホストクラスハブモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

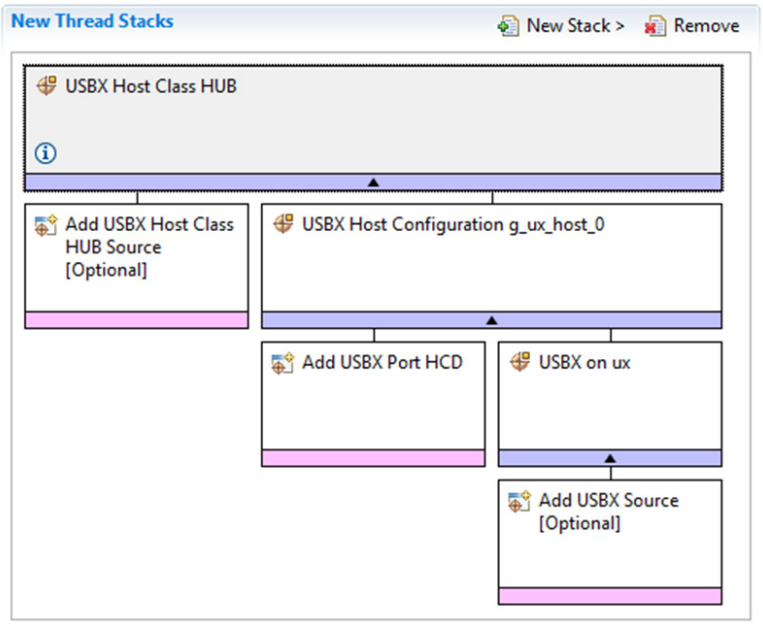


図 427:USBX ホストクラスハブモジュールのスタック

4.3.43.5 USBX ホストクラスハブモジュールの構成

ユーザーは必要な動作に合わせて USBX ホストクラスハブモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます (ブロックが赤で強調表示される)。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプ

ローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

USBX ホストクラスハブモジュールの構成設定

ISDE Property	Value	Description
No configurable properties()	-	-

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

USBX ホストクラスハブのローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

USBX ホストクラスハブソースの構成設定

ISDE Property	Value	Description
Show linkage warning	Enabled, Disabled Default: Enabled	Notification message for users will be shown if "Enabled" option is selected. This is just to warn users possible linkage errors by multiple symbol definitions. Select "Disabled" stops the notification message.

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX ホスト構成インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ux_host0	Specify the name of USBX Host Configuration instance. It must be a valid C symbol.
Name of generated initialization function	ux_host_hid_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBFS の sf_el_ux 上の USBX ポート HCD の構成設定

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for full-speed USB.
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	Select the VBUSEN pin signal logic.
LDO Regulator (Only for S3 and S1 part MCUs)	Enable, Disable Default: Disable	Select the LDO regulator will be enabled.
Name	g_sf_el_ux_hcd_fs_0	Module name.
USB Controller Selection	USBFS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBHS の sf_el_ux 上の USBX ポート HCD の構成設定

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for high speed USB.
FIFO size for Bulk/Isochronous Pipes	512, 1024, 1536, 2048 bytes Default: 512 bytes	Select the FIFO size for bulk and isochronous transfers.
Number of Isochronous Pipes Reserved	0,1,2 Default: 0	Select the number of isochronous pipes to reserve.
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	Select the VBUSEN pin signal logic.
Enable High Speed	Enable, Disable Default: Enable	Select if high speed should be enabled.
Name	g_sf_el_ux_hcd_hs_0	Module name.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ux 上の USBX インスタンスの構成設定

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	Name must be a valid C symbol.

ISDE Property	Value	Description
USBX Pool Memory Size	18432	See section "Express Logic USBX Memory Requirements" for the required memory size for each classes.
User Callback for Host Event Notification (Only valid for USB Host)	NULL	Name must be a valid C symbol. The name of User defined USBX Host event notification can be given to this property.
Name of generated initialization function	ux_common_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX ホストクラスハブモジュールのクロック構成

USB ペリフェラルモジュールは UCLK をクロックソースとして使用します。UCLK は 48 MHz の動作に合わせて構成する必要があります。SSP 構成ウィンドウで、[Clocks] タブを選択してクロックソースの設定を確認します。

USBX ホストクラスハブモジュールのピン構成

USB ペリフェラルモジュールは、MCU ピンを使用して外部デバイスと通信します。I/O ピンを選択し、外部デバイスの要件に合うように構成します。次の表では [SSP Configuration] ウィンドウでのピンの選択方法を示し、それ以降の表では USB ピンを例に使って選択プロセスを示します。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

USBFS および USBHS のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
USBFS	Pins	Select Peripherals > Connectivity: USBFS> USBFS0
USBHS	Pins	Select Peripherals > Connectivity: USBHS> USBHS0

注 : 選択シーケンスでは、USBFS0 または USBHS0 がドライバーに必要なハードウェアターゲットであることを想定しています。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select device as the Operation Mode
USBDP	USBDP	USBDP pin
USBDM	USBDM	USBDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	None	VBUSEN pin
VBUS	None, P407 Default: P407	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID Pin
VCCUSB	VCCUSB	VCCUSB pin
VSSUSB	VSSUSB	VSSUSB pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select Device as the Operation Mode
USBHSDP	USBHSDP	USBHSDP pin
USBHSDM	USBHSDM	USBHSDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	PB00	VBUSEN pin
VBUS	PB01	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID pin
USBHSRREF	USBHSRREF	USBHSRREF pin
AVCCUSBHS	AVCCUSBHS	AVCCUSBHS pin
AVSSUSBHS	AVSSUSBHS	AVSSUSBHS pin
PVSSUSBHS	PVSSUSBHS	PVSSUSBHS pin
VCCUSBHS	VCCUSBHS	VCCUSBHS pin
VSS1USBHS	VSS1USBHS	VSS1USBHS pin
VSS2USBHS	VSS2USBHS	VSS2USBHS pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.43.6 アプリケーションでの USBX ホストクラスハブモジュールの使用

通常、アプリケーションがハブモジュールを単独で使用することはなく、他のクラス（CDC-ACM、ストレージ、HID など）も同時に使用します。

コンフィギュレータは、USBX ホストクラスハブモジュールの登録に必要な処理を生成します。同時に使用するクラスに登録されている USBX ホスト構成モジュールと同じモジュールを指定します。

次の図は、USBX ホストクラス大容量記憶装置モジュールと同時に登録されたスタックを示しています。

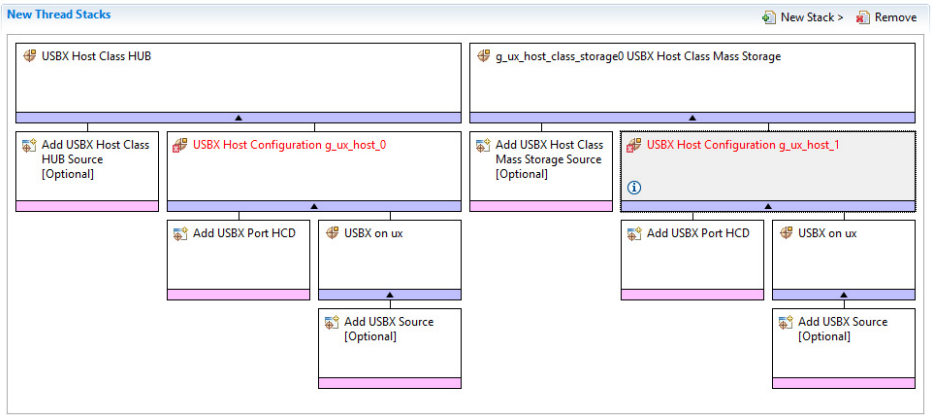


図 428:一般的な USBX ホストクラスハブモジュールアプリケーションのフロー図

4.3.44 USBX ホストクラス記憶装置

USBX ™ホストクラス大容量記憶装置モジュールは、USBX ホストクラス大容量記憶装置モジュールアプリケーション用のハイレベル API を提供し、Synergy MCU 上の USB およびデータ転送ペリフェラルを使用します。

4.3.44.1 USBX ホストクラス大容量記憶装置モジュールの特長

- USB 2.0 対応のホストコントローラによって以下をサポートします。
  - ルートハブ
  - 電源管理
  - エンドポイント
  - 転送
- ハイレベル API によってストレージ動作を簡略化します。
- 効率向上のために、MCU ハードウェアを使用したオプションのデータ転送をサポートします。

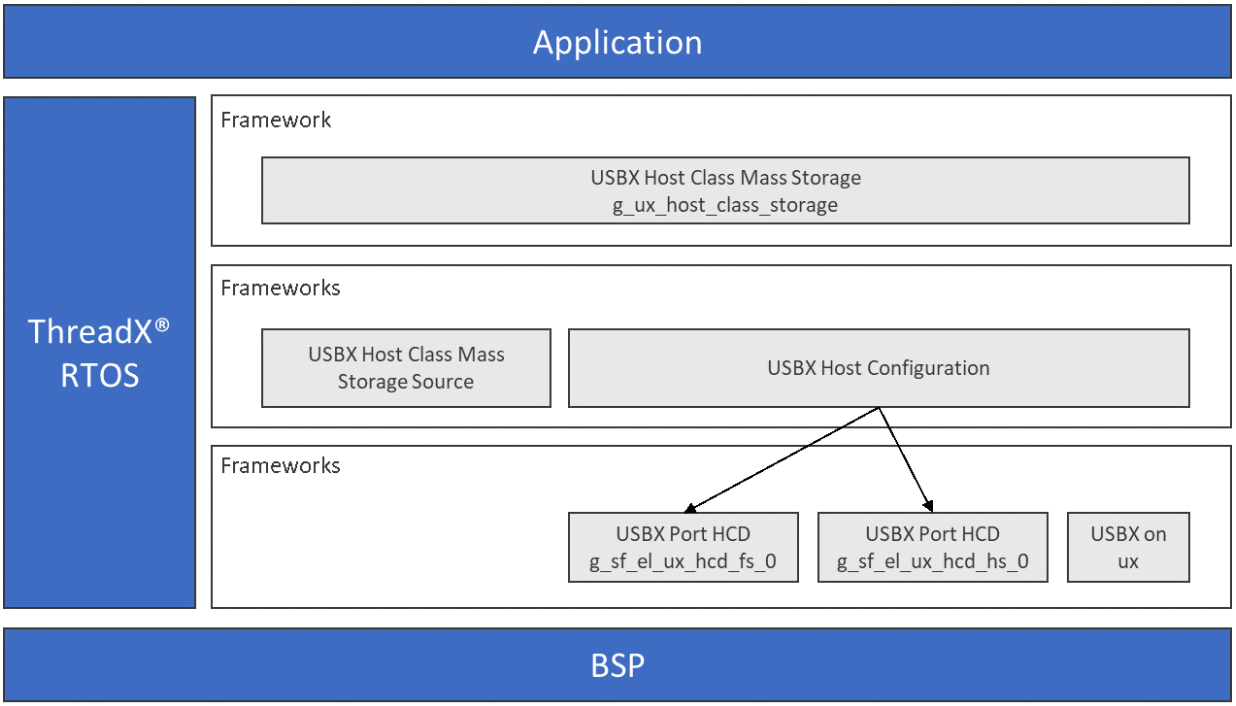


図 429:**USBX ホストクラス大容量記憶装置モジュールのブロック図**

4.3.44.2 USBX ホストクラス大容量記憶装置モジュールの API の概要

USBX ホストクラス大容量記憶装置モジュールには、ユーザーアプリケーション用の個別の API はありません。USB メディアは、接続されると、USBX ホストクラス大容量記憶装置インスタンスに含まれる FileX® メンバーにアタッチされます。ユーザーアプリケーションは、この FileX メンバーを使用して USB メディア上のファイルにアクセスします。FileX API の詳細については、『FileX User Guide for the Renesas Synergy ™ Platform』を参照してください。

4.3.44.3 USBX ホストクラス大容量記憶装置モジュールの動作の概要

USBX リソースの初期化

USBX には、独自のメモリマネージャがあります。メモリは、USBX のホスト側またはデバイス側が初期化される前に割り当てる必要があります。USBX メモリマネージャは、メモリのキャッシュが可能なシステムに対応できます。

USB ホストコントローラの定義

USBX をホストモードで動作させるには、USB ホストコントローラを 1 つ以上定義する必要があります。この定義は、アプリケーションの初期化ファイルに含める必要があります。SSP は、USB ホストコントローラドライバーがスレッドタスクに追加された場合に、USB ホストコントローラを定義します。

デバイスクラスの定義

USBX には、1 つ以上のデバイスクラスを定義する必要があります。USB クラスは、USB スタックが USB デバイスを構成した後に、USB デバイスを駆動するために必要です。USB クラスは、デバイスに固有です。USB デバイス記述子に含まれるインタフェースの数に応じて、USB を駆動するために 1 つ以上のクラスが必要になることがあります。

USB クラスのバインド

デバイスが構成されると、トポロジマネージャは、クラスマネージャがデバイスインタフェース記述子を調べ、デバイスの検出を続行することを許可します。1 つのデバイスに複数のインタフェース記述子が存在する場合があります。

インタフェースは、デバイスの機能を表します。たとえば、USB スピーカには、オーディオストリーミング用、オーディオ制御用、各種スピーカボタンの管理用の 3 つのインタフェースが存在します。

クラスマネージャには、デバイスインタフェースを 1 つ以上のクラスに結合する 2 つのメカニズムがあります。1 つはインタフェース記述子に含まれる PID と VID（製品 ID とベンダ ID）の組み合わせを使用する方法、もう 1 つはクラス、サブクラス、プロトコルの組み合わせを使用する方法です。

PID と VID の組み合わせは、一般的なクラスでは駆動できないインタフェースに有効です。クラス、サブクラス、プロトコルの組み合わせは、プリンタ、ハブ、ストレージ、オーディオ、ヒューマンインタフェースデザイン（HID）など、USB-IF が認定したクラスに属するインタフェースに使用されます。

クラスマネージャには、USBX の初期化で登録されたクラスのリストが格納されます。クラスマネージャは、デバイスのインタフェースを管理するクラスが決まるまでクラスを 1 つずつ呼び出します。1 つのクラスが管理できるインタフェースは、1 つだけです。USB オーディオスピーカの場合は、クラスマネージャは、各インタフェースに対してすべてのクラスを呼び出します。

クラスがインタフェースを受け入れると、そのクラスの新しいインスタンスが作成され、クラスマネージャがインタフェースのデフォルトの代替設定を検索します。1 つのデバイスの各インタフェースには、1 つ以上の代替設定が存在する場合があります。クラスで変更されない限り、デフォルトで代替設定 0 が使用されます。

デフォルトの代替設定の場合は、クラスマネージャが代替設定に含まれるすべてのエンドポイントをマウントします。各エンドポイントのマウントが成功すると、インタフェースの初期化を完了するクラスにクラスマネージャが戻ってジョブを完了します。

USBX ホストクラス大容量記憶装置モジュールの動作に関する重要な注意事項と制限事項

USBX デバイススタックまたは USBX ホストスタックは、制御ブロックの RAM を消費します。Synergy 構成ツールは、次の表に示すように、自動生成コードでメモリを USBX メモリプールに静的に割り当てます。[USBX on ux Configuration] セクションの Synergy 構成ツールで ux 上の USBX コンポーネントの [USBX Pool Memory Size] プロパティに、適切なメモリサイズ（バイト単位）を設定する必要があります。クラスを複数使用する場合は、このプロパティに合計メモリサイズを設定してください。

USBX メモリプールのメモリ（RAM）要件

USBX Class	S1 Parts	Other Parts
USBX Host Mass Storage (ux_host_class_storage)	N/A	Pre-built library: 42KB

ソース：39KB + UX_HOST_CLASS_STORAGE_MEMORY_BUFFER_SIZE + UX_HOST_CLASS_STORAGE_THREAD_STACK_SIZE |

注 : 上の表に示す情報は、デフォルトの USBX 構成でコンパイルした場合のものです。

注 : 上の表に示す USBX クラスのメモリサイズは、ビルド済みライブラリのものであり、ビルドには以下の構成が適用されています。

UX_THREAD_STACK_SIZE:S1 パーツには 512（バイト）、その他のパーツには 2048（バイト）。

注：ソースモジュールを使用してビルドした場合の USBX ホスト大容量記憶装置のメモリサイズも示していますが、これは、メモリサイズが UX_HOST_CLASS_STORAGE_MEMORY_BUFFER_SIZE 構成によって決まり、ビルド済みライブラリのサイズとは大きく異なるためです。サイズは、次の構成が適用された場合のものです。

UX_THREAD_STACK_SIZE: 2048（バイト）

- 大容量記憶装置インスタンスを取得するために、自動生成コードによって取得される USBX ホストクラス大容量記憶装置モジュール用のクラスコンテナを使用します。
- ux_host_class_storage_state フラグをポーリングし、ステータスがライブであることを確認します。
- クラスコンテナの最初のメディアがアタッチされるまで待機します。
- FileX API を使用してメディア上のファイルにアクセスします。
- このモジュールは、USB コントローラの割り込みを使用します。適切に動作するために、Synergy 構成ツールで適切な割り込み優先順位レベルを設定してください。
- このモジュールは、転送モジュール（DMAC または DTC として実装）の割り込みを使用します。Synergy 構成ツールで適切な優先順位レベルを設定してください。適切に動作するために、レベルは USB コントローラより高くする必要があります。
- USBX ホストクラス大容量記憶装置モジュールは、ストレージデバイス上で MBR パーティションタイプのみをサポートします。
- このモジュールの動作に関するその他の制限事項については、最新の SSP リリースノートを参照してください。

4.3.44.4 アプリケーションへの USBX ホストクラス大容量記憶装置モジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに USBX ホストクラス大容量記憶装置モジュールを組み込む方法について説明します。

注：このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

USBX ホストクラス大容量記憶装置モジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

USBX ホストクラス大容量記憶装置モジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ux_host_class_hid0 USBX Host Class HID	Threads	New Stack> X-Ware> USBX> Host > Classes > HID > USBX Host Class HID

次の図に示すように、USBX ホストクラス大容量記憶装置モジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピ

ンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。（ブロック内でテキストによって示されます。）ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

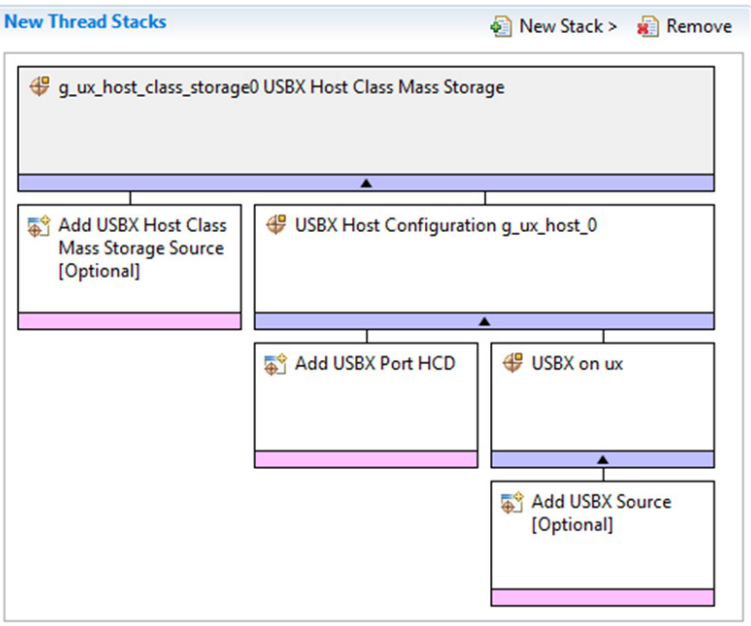


図 430:USBX ホストクラス大容量記憶装置モジュールのスタック

4.3.44.5 USBX ホストクラス大容量記憶装置モジュールの構成

ユーザーは必要な動作に合わせて USBX ホストクラス大容量記憶装置モジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

USBX ホストクラス大容量記憶装置モジュールの構成設定

ISDE Property	Value	Description
Name	g_ux_host_class_storage0	Module name.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

USBX ホストクラス大容量記憶装置のローレベルモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

USBX ホストクラス大容量記憶装置ソースの構成設定

ISDE Property	Value	Description
Use FileX Stub	Enabled, Disabled Default: Disabled	UX_NO_FILEX Enable this option only in cases where FileX is not available to use.
Maximum number of SCSI logical units	Value must be greater than 0 or empty Default: 1	UX_MAX_HOST_LUN The value represents the maximum number of SCSI logical units represented in the host storage class driver.
Maximum number of storage media instance	Value must be greater than 0 or empty Default: 1	UX_HOST_CLASS_STORAGE_MAX_MEDIA The value represents the maximum number of storage media instances represented in the host storage class driver.

ISDE Property	Value	Description
Storage memory size in bytes for FileX used for data transfer	Value must be greater than 512 or empty Default: 1024	UX_HOST_CLASS_STORAGE_MEMORY_BUFFER_SIZE The value represents the block of memory in bytes for FileX to use when doing transfers. The value can be changed to save on memory space but should not be smaller than the media sector size. Because USB devices are SCSI devices and there is a great deal of overhead when doing read/writes, it is better to increase it for higher data throughput.
Maximum transfer size in bytes in one BOT data-transport phase	Value must be greater than 512 or empty Default: 1024	UX_HOST_CLASS_STORAGE_MAX_TRANSFER_SIZE The value represents maximum size of memory chunk in bytes to send in one data-transport phase in the Bulk-Only Transport protocol. Large size of data transfer request is split into smaller chunks specified by this configuration. It is better to increase it for higher data throughput.
Stack size for the Mass Storage Class internal thread	Value must be greater than 512 or empty Default: 1024	UX_HOST_CLASS_STORAGE_THREAD_STACK_SIZE The value represents the stack size in bytes for Mass Storage Class internal thread named ux_storage_thread.

ISDE Property	Value	Description
Timeout (in milliseconds) for a BOT transfer request	Value must be greater than 0 or empty Default: 100000	UX_HOST_CLASS_STORAGE_TRANSFER_TIMEOUT_IN_MS The value represents timeout value in millisecond for a data transfer request in Bulk-Only Transport protocol.
Timeout (in milliseconds) for the status from a command in the Control/Bulk/Interrupt transport	Value must be greater than 0 or empty Default: 30000	UX_HOST_CLASS_STORAGE_CBI_STATUS_TIMEOUT_IN_MS The value represents timeout value in millisecond for the status from a command, which is returned by the interrupt endpoint in the Control/Bulk/ Interrupt transport. The transport is mainly used by storage devices that have very slow read/write commands.
Show linkage warning	Enabled, Disabled Default: Enabled	Notification message for users will be shown if “Enabled" option is selected. This is just to warn users possible linkage errors by multiple symbol definitions. Select “Disabled” stops the notification message.

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX ホスト構成インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ux_host0	Specify the name of USBX Host Configuration instance. It must be a valid C symbol.

ISDE Property	Value	Description
Name of generated initialization function	ux_host_hid_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBFS の sf_el_ux 上の USBX ポート HCD の構成設定

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for full-speed USB.
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	Select the VBUSEN pin signal logic.
LDO Regulator (Only for S3 and S1 part MCUs)	Enable, Disable Default: Disable	Select the LDO regulator will be enabled.
Name	g_sf_el_ux_hcd_fs_0	Module name.
USB Controller Selection	USBFS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBHS の sf_el_ux 上の USBX ポート HCD の構成設定

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for high speed USB.
FIFO size for Bulk/Isochronous Pipes	512, 1024, 1536, 2048 bytes Default: 512 bytes	Select the FIFO size for bulk and isochronous transfers.
Number of Isochronous Pipes Reserved	0,1,2 Default: 0	Select the number of isochronous pipes to reserve.
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	Select the VBUSEN pin signal logic.
Enable High Speed	Enable, Disable Default: Enable	Select if high speed should be enabled.
Name	g_sf_el_ux_hcd_hs_0	Module name.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ux 上の USBX インスタンスの構成設定

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	Name must be a valid C symbol.

ISDE Property	Value	Description
USBX Pool Memory Size	18432	See section "Express Logic USBX Memory Requirements" for the required memory size for each classes.
User Callback for Host Event Notification (Only valid for USB Host)	NULL	Name must be a valid C symbol. The name of User defined USBX Host event notification can be given to this property.
Name of generated initialization function	ux_common_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX ホストクラス大容量記憶装置モジュールのクロック構成

USB ペリフェラルモジュールは UCLK をクロックソースとして使用します。UCLK は 48 MHz の動作に合わせて構成する必要があります。SSP 構成ウィンドウで、[Clocks] タブを選択してクロックソースの設定を確認します。

USBX ホストクラス大容量記憶装置モジュールのピン構成

USB ペリフェラルモジュールは、MCU ピンを使用して外部デバイスと通信します。I/O ピンを選択し、外部デバイスの要件に合うように構成します。次の表では [SSP Configuration] ウィンドウでのピンの選択方法を示し、それ以降の表では USB ピンを例に使って選択プロセスを示します。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

USBFS および USBHS のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
USBFS	Pins	Select Peripherals > Connectivity: USBFS> USBFS0
USBHS	Pins	Select Peripherals > Connectivity: USBHS> USBHS0

注 : 選択シーケンスでは、USBFS0 または USBHS0 がドライバーに必要なハードウェアターゲットであることを想定しています。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select device as the Operation Mode
USBDP	USBDP	USBDP pin
USBDM	USBDM	USBDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	None	VBUSEN pin
VBUS	None, P407 Default: P407	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID Pin
VCCUSB	VCCUSB	VCCUSB pin
VSSUSB	VSSUSB	VSSUSB pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select Device as the Operation Mode
USBHSDP	USBHSDP	USBHSDP pin
USBHSDM	USBHSDM	USBHSDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	PB00	VBUSEN pin
VBUS	PB01	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID pin
USBHSRREF	USBHSRREF	USBHSRREF pin
AVCCUSBHS	AVCCUSBHS	AVCCUSBHS pin
AVSSUSBHS	AVSSUSBHS	AVSSUSBHS pin
PVSSUSBHS	PVSSUSBHS	PVSSUSBHS pin
VCCUSBHS	VCCUSBHS	VCCUSBHS pin
VSS1USBHS	VSS1USBHS	VSS1USBHS pin
VSS2USBHS	VSS2USBHS	VSS2USBHS pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.44.6 アプリケーションでの USBX ホストクラス大容量記憶装置モジュールの使用

コンフィギュレータは、USBX ホストクラス大容量記憶装置モジュールの作成と登録に必要な処理を生成しますが、通信はホストがホストに接続されてから行う必要があります。

アプリケーションで USBX ホストクラス大容量記憶装置モジュールを使用する際の一般的な手順は次のとおりです。

1) ux_host_class_instance_get API を使用して、接続されたデバイスの最初のインスタンスを取得します。
2) 成功するまで待機します。
3) デバイスのステータスがライブになるまで待機します。
4) メディアがマウントされるまで待機します。
5) FileX API を使用してメディア上のファイルにアクセスします。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

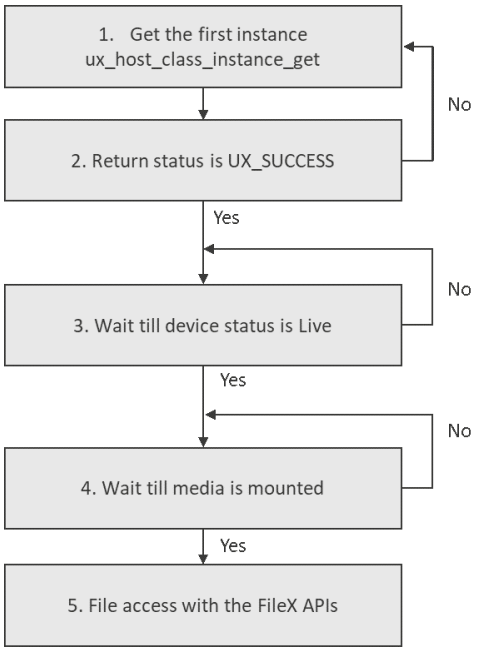


図 431:一般的な USBX ホストクラス大容量記憶装置モジュールアプリケーションのフロー図

4.3.45 USBX ホストクラスビデオ

USBX™ホストクラスビデオモジュールは、USBX ホストクラスビデオアプリケーション用のハイレベル API を提供し、USBX ホストクラスビデオソース、USBX ホスト構成、USBX ソース、USBX ポートホストコントローラデバイスを構成します。USBX ホストクラスビデオモジュールは、Synergy MCU 上の USB ペリフェラルを使用します。

4.3.45.1 USBX ホストクラスビデオモジュールの特長

ビデオクラスモジュールは USB カメラとの通信をモニタします。以下の特長があります。

- ビデオフォーマット、解像度、接続された USB カメラのフレームレートの情報収集のサポート

- ビデオフォーマットや解像度、フレームレートなどを任意の値に設定したビデオストリーミングの開始と停止のサポート

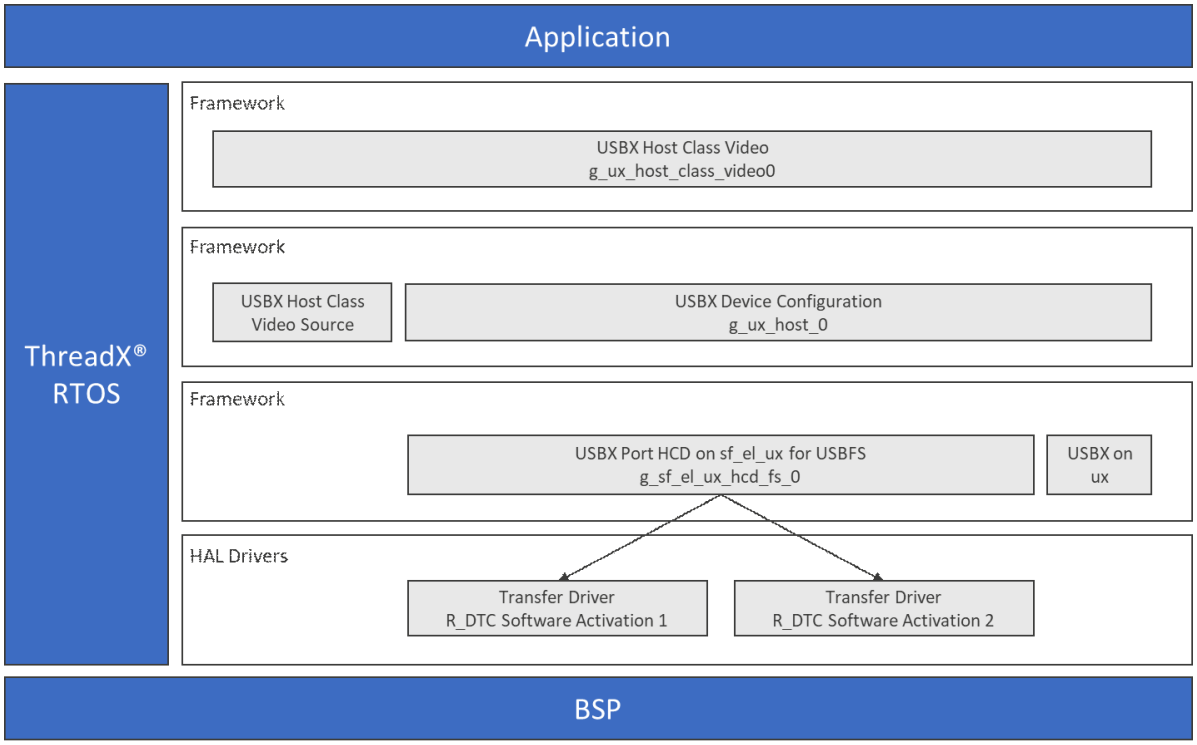


図 432:**USBX ホストクラスビデオモジュールのブロック図**

4.3.45.2 USBX ホストクラスビデオモジュールの API の概要

USBX ホストクラスモジュールでは、開始と停止、パラメータの設定、最大サイズの取得、バッファの追加、コールバックの設定のための API が定義されています。使用可能なすべての API のリスト、API 呼び出しの例、各 API の簡単な説明が、次の表に含まれます。ステータス戻り値の表は API 要約表の後にあります。

USBX ホストクラスビデオモジュールの API の要約

Function Name	Example API Call and Description
ux_host_class_video_start	status = ux_host_class_video_start(video); This function starts the video streaming. The video channel needs to be properly configured prior to calling this function.
ux_host_class_video_stop	status = ux_host_class_video_stop(video); This service stops the current video channel.

Function Name	Example API Call and Description
ux_host_class_video_frame_parameters_set	status = ux_host_class_video_frame_parameters_set(video, frame_format, width, height, frame_interval); This function sets the video parameters for the video device.
ux_host_class_video_max_payload_get	length = ux_host_class_video_max_payload_get(video); This function returns the maximum payload size for a given video parameter setting. After properly configures the video streaming parameters (such as video encoding, resolution, frame rate), application may use this function to obtain the maximum payload size. With the maximum payload size, application can allocate memory buffers for receiving incoming video frame data.
ux_host_class_video_transfer_buffer_add	Status = ux_host_class_video_transfer_buffer_add(video, buffer); This function passes a buffer to the video device, which is used to store incoming video stream data. The size of the buffer must be at least the maximum of the video payload size, which can be obtained by calling ux_host_class_video_max_payload_get.
ux_host_class_video_transfer_callback_set	ux_host_class_video_transfer_callback_set(video, callback_function); This function sets the video transfer callback function. This callback function is invoked once a transfer request has been fulfilled, and the application is ready to consume the video data.

注：関数データ構造体、型定義、定義、API データ、API 構造体、関数変数の動作と定義の詳細な説明については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

ステータス戻り値

Name	Description
UX_SUCCESS	The data transfer was completed.

Name	Description
UX_TRANSFER_TIMEOUT	Transfer timeout, reading/writing not completed.
UX_MEMORY_INSUFFICIENT	Not enough memory.
UX_HOST_CLASS_INSTANCE_UNKNOWN	The video instance is not valid.
UX_HOST_CLASS_VIDEO_PARAMETER_ERROR	The desired video parameters are not supported by this camera.

注 : ローレベルドライバーによって共通のエラーコードが返される場合があります。すべての関連ステータス戻り値の定義については、『SSP ユーザーズマニュアル』で関連モジュールの API リファレンスを参照してください。

4.3.45.3 USBX ホストクラスビデオモジュールの動作の概要

USBX リソースの初期化

USBX には、独自のメモリマネージャがあります。メモリは、USBX のホスト側またはデバイス側が初期化される前に割り当てる必要があります。USBX メモリマネージャは、メモリのキャッシュが可能なシステムに対応できます。

USB ホストコントローラの定義

USBX をホストモードで動作させるには、USB ホストコントローラを 1 つ以上定義する必要があります。この定義は、アプリケーションの初期化ファイルに含める必要があります。SSP は、USB ホストコントローラドライバーがスレッドタスクに追加された場合に、USB ホストコントローラを定義します。

デバイスクラスの定義

USBX には、1 つ以上のデバイスクラスを定義する必要があります。USB クラスは、USB スタックが USB デバイスを構成した後に、USB デバイスを駆動するために必要です。USB クラスは、デバイスに固有です。USB デバイス記述子に含まれるインタフェースの数に応じて、USB を駆動するために 1 つ以上のクラスが必要になることがあります。

USB クラスのバインド

デバイスが構成されると、トポロジマネージャは、クラスマネージャがデバイスインタフェース記述子を調べ、デバイスの検出を続行することを許可します。1 つのデバイスに複数のインタフェース記述子が存在する場合があります。

インタフェースは、デバイスの機能を表します。たとえば、USB スピーカには、オーディオストリーミング用、オーディオ制御用、各種スピーカボタンの管理用の 3 つのインタフェースが存在します。

クラスマネージャには、デバイスインタフェースを 1 つ以上のクラスに結合する 2 つのメカニズムがあります。1 つはインタフェース記述子に含まれる PID と VID（製品 ID とベンダ ID）の組み合わせを使用する方法、もう 1 つはクラス、サブクラス、プロトコルの組み合わせを使用する方法です。

PID と VID の組み合わせは、一般的なクラスでは駆動できないインタフェースに有効です。クラス、サブクラス、プロトコルの組み合わせは、プリンタ、ハブ、ストレージ、オーディオ、ヒューマンインタフェースデザイン（HID）など、USB-IF が認定したクラスに属するインタフェースに使用されます。

クラスマネージャには、USBX の初期化で登録されたクラスのリストが格納されます。クラスマネージャは、デバイスのインタフェースを管理するクラスが決まるまでクラスを 1 つずつ呼び出します。1 つのクラスが管理できるインタフェースは、1 つだけです。USB オーディオスピーカの場合は、クラスマネージャは、各インタフェースに対してすべてのクラスを呼び出します。

クラスがインタフェースを受け入れると、そのクラスの新しいインスタンスが作成され、クラスマネージャがインタフェースのデフォルトの代替設定を検索します。1 つのデバイスの各インタフェースには、1 つ以上の代替設定が存在する場合があります。クラスで変更されない限り、デフォルトで代替設定 0 が使用されます。

デフォルトの代替設定の場合は、クラスマネージャが代替設定に含まれるすべてのエンドポイントをマウントします。各エンドポイントのマウントが成功すると、インタフェースの初期化を完了するクラスにクラスマネージャが戻ってジョブを完了します。

USBX ホストクラスビデオモジュールの動作に関する重要な注意事項と制限事項

USBX デバイススタックまたは USBX ホストスタックは、制御ブロックの RAM を消費します。Synergy 構成ツールは、次の表に示すように、自動生成コードでメモリを USBX メモリプールに静的に割り当てます。[USBX on ux Configuration] セクションの Synergy 構成ツールで ux 上の USBX コンポーネントの [USBX Pool Memory Size] プロパティに、適切なメモリサイズ（バイト単位）を設定する必要があります。クラスを複数使用する場合は、このプロパティに合計メモリサイズを設定してください。

USBX メモリプールのメモリ（RAM）要件

USBX Class	S1 Parts	Other Parts
USBX Host Video (ux_host_class_video)	N/A	35KB

注 : 上の表に示す情報は、デフォルトの USBX 構成でコンパイルした場合のものです。

*注 : 上の表に示す USBX クラスのメモリサイズは、ビルド済みライブラリのものであり、ビルドには以下の構成が適用されています。*UX_THREAD_STACK_SIZE:S1 パーツには 512（バイト）、その他のパーツには 2048（バイト）。

注 :USBX ホストビデオのメモリプールサイズは、ボードにアタッチされたビデオデバイスの種類によって異なる場合があります。メモリサイズを計算する正確な式はありません。標準的な USBX ビデオアプリケーションに関する一般的な考え方は次のとおりです。

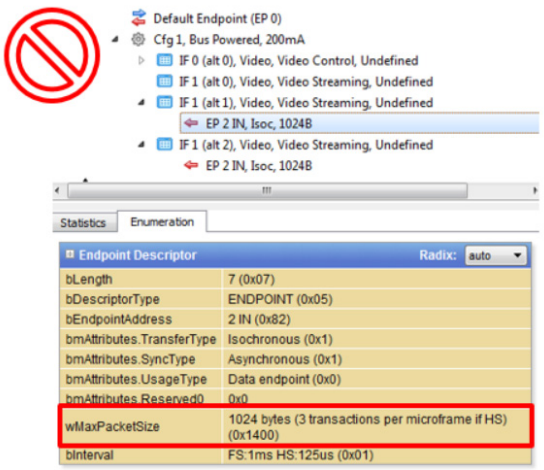
（1）デバイス記述子：デバイス記述子のメモリ要件は、使用中のビデオカメラによって異なります。デバイス記述子に必要なメモリサイズは、一部の USB カメラでは 2,000 になる可能性がありますが、シンプルなカメラではかなり少なくなります。

（2）エンドポイント：エンドポイントごとに約 100 バイト。エンドポイントの数は、使用中のデバイスによって異なります（5 つ以上の場合もあります）。

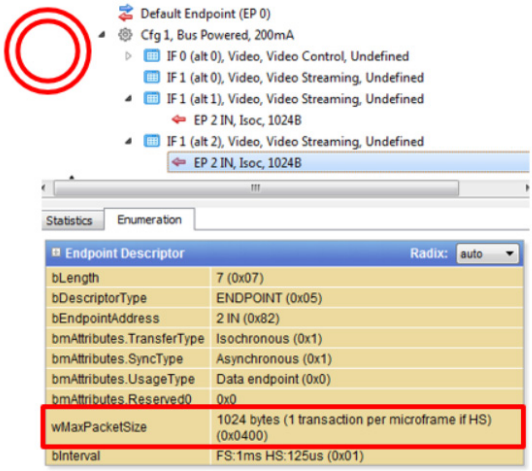
（3）転送要求バッファ：1 つにつき 100。USB コントロールは、1 つの転送要求を受け取ります。転送要求の数は、アプリケーションによって異なります。デフォルトでは、USBX ビデオクラスは 4 つの転送要求を作成します。

USB Host Class Video Module の機能制限

UVC（USB ビデオクラス）デバイスのインタフェースで代替設定を指定するには、ux_host_class_video_start(), ではなく ux_host_class_video_ioctl() USBX API を使用し、引数 “ioctl_function”. に UX_HOST_CLASS_VIDEO_IOCTL_CHANNEL_START を指定することが必要な場合があります。
また、.ux_host_class_video_parameter_channel_bandwidth_selection メンバーをアイソクロナスエンドポイントの wMaxPacketSize に合わせて設定し、引数パラメータを指定する必要もあります。たとえば、次の図に示すデバイス記述子を持つ UVC デバイスに対して、.ux_host_class_video_parameter_channel_bandwidth_selection を 1024 に設定した場合、IF1 Alt.1 は選択されません（高帯域幅の転送が要求されるため）。代わりに、IF1 Alt.2 が選択されます。アイソクロナス転送の制限については、Logic USBX Synergy ポートフレームワークの制限事項を参照してください。



(this will not be selected)



(this will be selected)

4.3.45.4 アプリケーションへの USBX ホストクラスビデオモジュールの組み込み

このセクションでは、SSP コンフィギュレータを使用してアプリケーションに USBX ホストクラスビデオモジュールを組み込む方法について説明します。

注 : このセクションを読むには、プロジェクトの作成、スレッドの追加、スレッドへのスタックの追加、およびスタック内でのブロックの構成について理解している必要があります。これらの項目に精通していない場合は、『SSP ユーザーズマニュアル』の最初のいくつかの章を参照して、SSP ベースのアプリケーションの作成時にこれらの重要な各手順を管理する方法を確認してください。

USBX ホストクラスビデオモジュールをアプリケーションに追加するには、次の表に示すスタック選択シーケンスを使用してスレッドに追加します。

USBX ホストクラスビデオモジュールの選択シーケンス

Resource	ISDE Tab	Stacks Selection Sequence
g_ux_host_class_video0 USBX Host Class Video	Threads	New Stack> X-Ware™> USBX> Host > Classes > Video > USBX Host Class Video

次の図に示すように、USBX ホストクラスビデオモジュールがスレッドスタックに追加されると、コンフィギュレータは必要なローレベルモジュールを自動的に追加します。追加の構成情報を必要とするモジュールは、ボックスのテキストが赤く強調表示されます。グレーの帯が表示されているモジュールは個別のスタンドアロンモジュールです。青い帯で示されるモジュールは共有または共通で、1 回のみ追加すれば複数のスタックで使用できます。ピンクの帯で示されるモジュールには、ローレベルモジュールの選択が必要です。これらはオプションまたは推奨です。(ブロック内でテキストによって示されます。) ローレベルモジュールの追加が必要な場合は、モジュールの説明に「Add」というテキストが含まれます。ピンクの帯で示されるモジュールをクリックすると、[New] アイコンが表示され、可能な選択肢が表示されます。

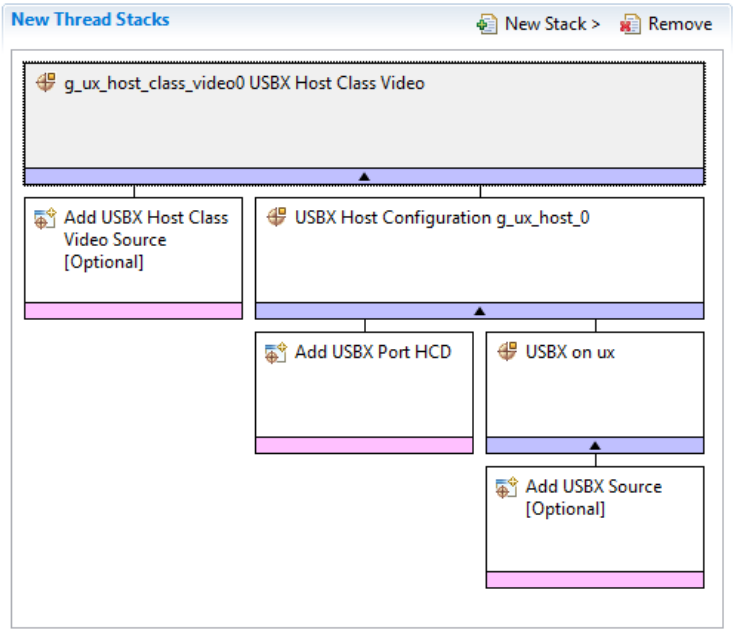


図 433:USBX ホストクラスビデオモジュールのスタック

4.3.45.5 USBX ホストクラスビデオモジュールの構成

ユーザーは必要な動作に合わせて USBX ホストクラスビデオモジュールを構成する必要があります。[SSP configuration] ウィンドウでは、割り込みや動作モードなど必須の構成選択項目が自動的に識別されます（ブロックが赤で強調表示される）。これらは、ローレベルモジュールが正常に作動するために構成する必要があります。変更しても競合が発生しないプロパティのみが変更可能になります。ロックされている他のプロパティは変更できません。これらはロックアイコンで識別され、ISDE 内の [Properties] ウィンドウでロックされているプロパティに対応します。このアプローチにより構成プロセスが簡略化され、以前の手動での構成アプローチに比べて大幅にエラーが発生しにくくなっています。ユーザーがアクセスできるすべてのプロパティの使用可能な構成設定とデフォルトは、SSP コンフィギュレータ内の [Properties] タブにありますが、簡単に参照できるようにこの後の表に示します。

注 : 次に示す構成表の値に目を通しながら、ISDE を開き、モジュールを作成し、プロパティ設定を調べることができます。これは正しい判断に役立ち、SSP での開発を学習するために有効な「実践的」アプローチになる可能性があります。

USBX ホストクラスビデオモジュールの構成設定

ISDE Property	Value	Description
Class Instance Name	g_ux_host_class_video0	Class instance name.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

注：ローレベルモジュールのプロパティ設定の大半は、通常は SSP コンフィギュレータで対応する [Properties] ウィンドウを調べることで決定されます。

USBX ホストクラスビデオモジュールの構成設定

IP レイヤーおよびローレベルドライバーでは少数の設定のみをデフォルト設定から変更する必要があり、これはスレッドスタックブロックに赤いテキストで示されます。一部の構成プロパティは、フレームワークが適切に作動するために特定の値に設定する必要があり、それらはロックされていてユーザーは変更できません。次の表に、モジュールの [properties] セクションのすべての設定を示します。

USBX ホストクラスビデオソースの構成設定

ISDE Property	Value	Description
Show linkage warning	Enabled, Disabled Default: Enabled	Notification message for users will be shown if "Enabled" option is selected. This is just to warn users possible linkage errors by multiple symbol definitions. Select "Disabled" stops the notification message.

注 : 設定例とデフォルトは、Synergy S7G2 MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX ホスト構成インスタンスの構成設定

ISDE Property	Value	Description
Name	g_ux_host0	Specify the name of USBX Host Configuration instance. It must be a valid C symbol.
Name of generated initialization function	ux_host_hid_init0	Name of generated initialization function selection.
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBFS の sf_el_ux 上の USBX ポート HCD の構成設定

ISDE Property	Value	Description
Full Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for full-speed USB.
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	Select the VBUSEN pin signal logic.
LDO Regulator (Only for S3 and S1 part MCUs)	Enable, Disable Default: Disable	Select the LDO regulator will be enabled.
Name	g_sf_el_ux_hcd_fs_0	Module name.
USB Controller Selection	USBFS	Select the USB controller.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBHS の sf_el_ux 上の USBX ポート HCD の構成設定

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), Priority 1:14, Priority 15 (lowest - not valid if using ThreadX), Disabled Default: Disabled	Select the interrupt priority for high speed USB.
FIFO size for Bulk/Isochronous Pipes	512, 1024, 1536, 2048 bytes Default: 512 bytes	Select the FIFO size for bulk and isochronous transfers.

ISDE Property	Value	Description
Number of Isochronous Pipes Reserved	0,1,2 Default: 0	Select the number of isochronous pipes to reserve.
VBUSEN pin Signal Logic	Active Low, Active High Default: Active High	Select the VBUSEN pin signal logic.
Enable High Speed	Enable, Disable Default: Enable	Select if high speed should be enabled.
Name	g_sf_el_ux_hcd_hs_0	Module name.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

ux 上の USBX インスタンスの構成設定

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	Name must be a valid C symbol.
USBX Pool Memory Size	18432	See section "Express Logic USBX Memory Requirements" for the required memory size for each classes.
User Callback for Host Event Notification (Only valid for USB Host)	NULL	Name must be a valid C symbol. The name of User defined USBX Host event notification can be given to this property.
Name of generated initialization function	ux_common_init0	Name of generated initialization function selection.

ISDE Property	Value	Description
Auto Initialization	Enable, Disable Default: Enable	Auto initialization selection.

注 : 設定例とデフォルトは、S7G2 Synergy MCU グループを使用するプロジェクトに対するものです。その他の MCU のデフォルト値と使用可能な構成設定は異なる可能性があります。

USBX ホストクラスビデオモジュールのクロック構成

USB ペリフェラルモジュールは UCLK をクロックソースとして使用します。UCLK は 48 MHz の動作に合わせて構成する必要があります。SSP 構成ウィンドウで、[Clocks] タブを選択してクロックソースの設定を確認します。

USBX ホストクラスビデオモジュールのピン構成

USB ペリフェラルモジュールは、MCU ピンを使用して外部デバイスと通信します。I/O ピンを選択し、外部デバイスの要件に合うように構成します。次の表では [SSP Configuration] ウィンドウでのピンの選択方法を示し、それ以降の表では USB ピンを例に使って選択プロセスを示します。

注 : 選択した動作モードによって、使用可能なペリフェラル信号と必要な MCU ピンが決定されます。

USBFS および USBHS のピン選択シーケンス

Resource	ISDE Tab	Pin selection Sequence
USBFS	Pins	Select Peripherals > Connectivity: USBFS> USBFS0
USBHS	Pins	Select Peripherals > Connectivity: USBHS> USBHS0

注 : 選択シーケンスでは、USBFS0 または USBHS0 がドライバーに必要なハードウェアターゲットであることを想定しています。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select device as the Operation Mode

Property	Value	Description
USBDP	USBDP	USBDP pin
USBDM	USBDM	USBDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	None	VBUSEN pin
VBUS	None, P407 Default: P407	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID Pin
VCCUSB	VCCUSB	VCCUSB pin
VSSUSB	VSSUSB	VSSUSB pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

USBHS のピン構成設定

Property	Value	Description
Operation Mode	Disabled, Custom, Device, Host, OTG Default: Custom	Select Device as the Operation Mode
USBHSDP	USBHSDP	USBHSDP pin
USBHSDM	USBHSDM	USBHSDM pin
OVRCURB	None	OVRCURB pin
OVRCURA	None	OVRCURA pin
VBUSEN	PB00	VBUSEN pin

Property	Value	Description
VBUS	PB01	VBUS pin
EXICEN	None	EXICEN pin
ID	None	ID pin
USBHSRREF	USBHSRREF	USBHSRREF pin
AVCCUSBHS	AVCCUSBHS	AVCCUSBHS pin
AVSSUSBHS	AVSSUSBHS	AVSSUSBHS pin
PVSSUSBHS	PVSSUSBHS	PVSSUSBHS pin
VCCUSBHS	VCCUSBHS	VCCUSBHS pin
VSS1USBHS	VSS1USBHS	VSS1USBHS pin
VSS2USBHS	VSS2USBHS	VSS2USBHS pin

注 : 設定例は、S7G2 Synergy MCU および SK-S7G2 キットを使用するプロジェクトに対するものです。他の Synergy MCU と他の Synergy キットでは使用可能なピン構成設定が異なる可能性があります。

4.3.45.6 アプリケーションでの USBX ホストクラスビデオモジュールの使用

コンフィギュレータは、USBX ホストクラスビデオモジュールの作成と登録に必要な処理を生成しますが、通信はホストがホストに接続されてから行う必要があります。

アプリケーションで USBX ホストクラスビデオモジュールを使用する際の一般的な手順は次のとおりです。

1) tx_event_flags_get API 関数を使用して、デバイスが挿入されるまで待機します。
2) ux_host_class_video_frame_parameters_set API 関数を使用して、カメラのパラメータを設定します。
3) ux_host_class_video_transfer_callback_set API 関数を使用して、ユーザーコールバック関数を設定します。
4) ux_host_class_video_max_payload_get API 関数を使用して、メモリバッファの最大サイズを取得します。
5) ux_host_class_video_start API 関数を使用して、ビデオの転送を開始します。
6) ux_host_class_video_transfer_buffer_add API 関数を使用して、ビデオデバイスにビデオバッファを追加します。
7) tx_semiphore_get API 関数を使用して、受信データフレームを待機します。
8) ux_host_class_video_transfer_buffer_add API 関数を使用して、ビデオバッファバックを追加します。
9) ux_host_class_video_stop API 関数を使用して、ビデオの転送を停止します。

これらの一般的な手順を、次の図の通常の動作フロー図に示します。

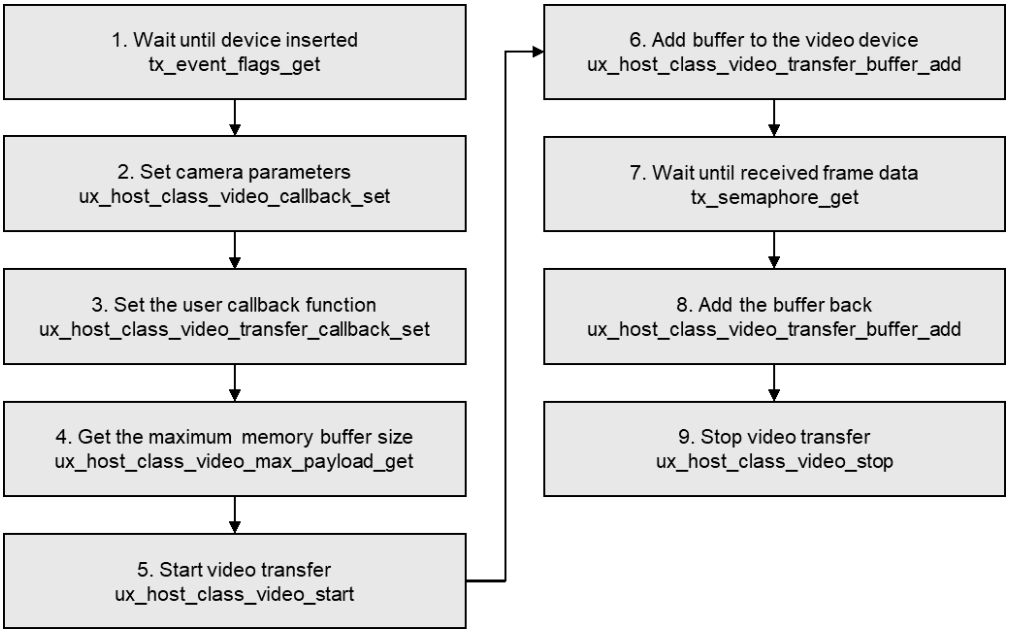


図 434:一般的な USBX ホストクラスビデオモジュールアプリケーションのフロー図

参考資料

Renesas Synergy™ Software Package（SSP）
v1.5.0ユーザーズマニュアル（参考資料）

発行年月日　2019年3月1日　Rev.1.10

発行　　ルネサス エレクトロニクス株式会社
〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

Renesas Synergy™ Software Package（SSP）
v1.5.0ユーザーズマニュアル（参考資料）

Renesas Electronics Corporation

R11UM0091JU0110