Smart Analog IC101

サンプルコードマイコン置き換え手順書

R21AN0013JJ0100
Rev1.00
2014.09.30

要旨

本ドキュメントでは、Smart Analog IC101(以下、SAIC101) API 関数を RL78/L13 グループ以外のマイコンで使用する場合の置き換え手順について説明します。

動作確認デバイス

Smart Analog IC 101（品名：RAA730101）

目次

1. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

- API 仕様書(R21AN0015JJ)
- サンプルコード導入手順書兼 API Builder SAIC101 仕様書（R21AN0012JJ）
- RL78/G13 シリアル・アレイ・ユニット（UART 通信）（R01AN0459JJ）アプリケーションノート
- RL78/G13 シリアル・アレイ・ユニット 3 線シリアル I/O （SPI マスタ送受信）【開発環境 : CubeSuite+、IAR、e2 studio】（R01AN1367JJ）アプリケーションノート

2. 他のマイコンへの置き換え手順

Smart Analog IC101 サンプルコードは、CubeSuite+用 RL78 ファミリ, 78K0R, 78K0 コード生成 (CubeSuite+ Code_Generator for RL78_78K)にて生成される RL78/L13 のシリアル通信モジュール用関数を API 関数から呼び出しています。RL78 ファミリ用コード生成ツールが生成する関数は RL78 ファミリ間で互換性があるため、対応している製品に関しては、コード開発支援ツール「API Builder SAIC101」に専用のマイコン定義ファイルを追加することにより、「API Builder SAIC101」を使用してサンプルコードの組み込みが可能となります。RL78 ファミリ用コード生成ツールに対応していない場合は、コード生成ツールで生成される関数に準拠した関数を作成し、サンプルコードおよび API ファイルをプロジェクトに手動で組み込む必要があります。

他のマイコンへの置き換え手順のフローを下記に示します。

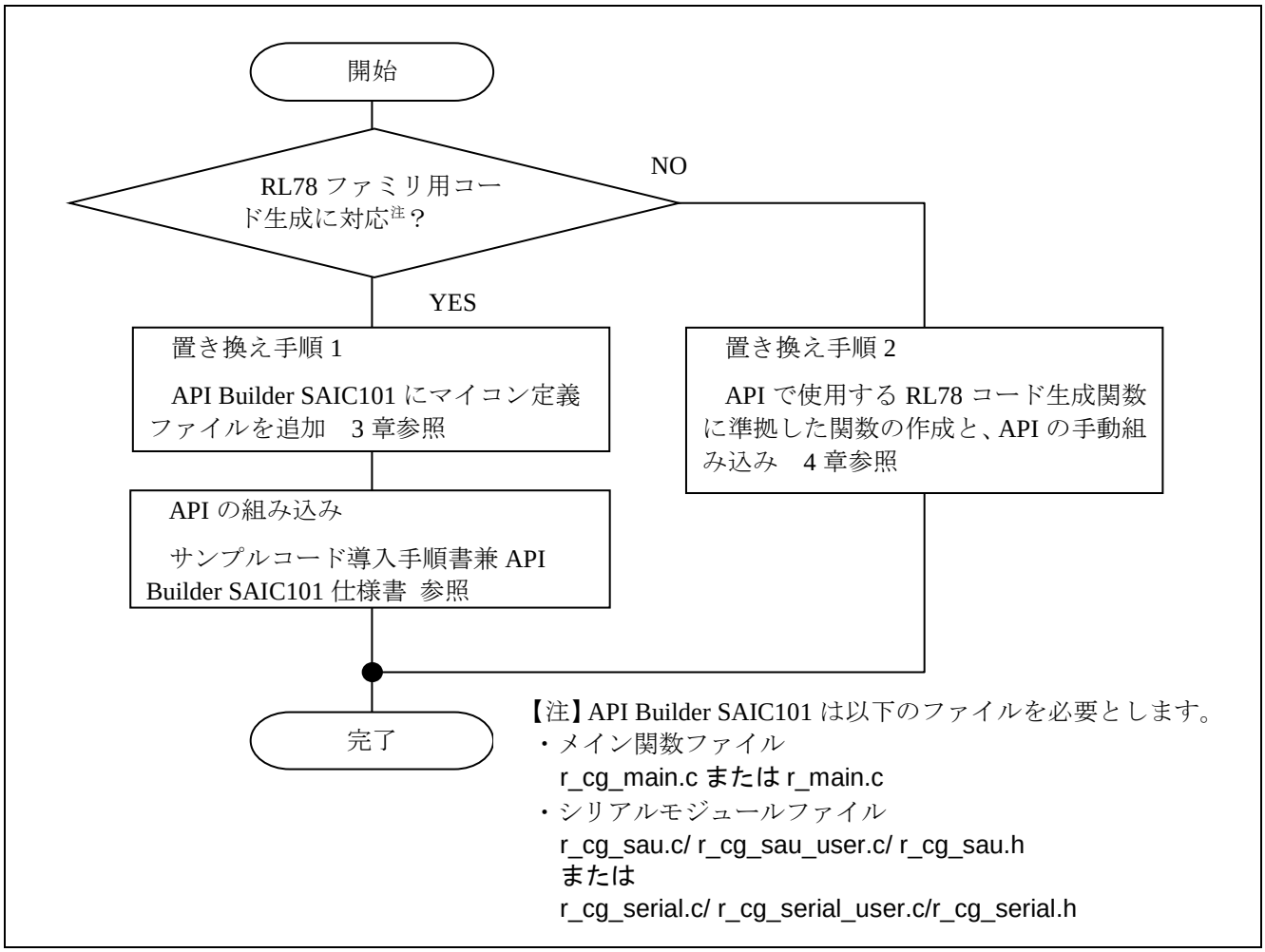


図 2-1　他のマイコンへの置き換え手順概要

3. API Builder SAIC101 の対応マイコンをユーザーにて追加する手順

API Builder SAIC101 が対応するマイコンは、RL78 マイコン(CubeSuite+ Code_Generator for RL78_78K が RL78/L13 と同等のシリアル通信のソースコードを出力するものに限る)であれば、マイコン定義ファイルを作成する事により、ユーザーにて追加が可能です。本章では RL78/G14(R5F104PJ)を例に挙げて追加手順を示します。

3.1 準備

まず初めに、追加したいマイコンの型名を確認します。ここで必要な型名は、CubeSuite+や e2studio で認識されている型名となります。

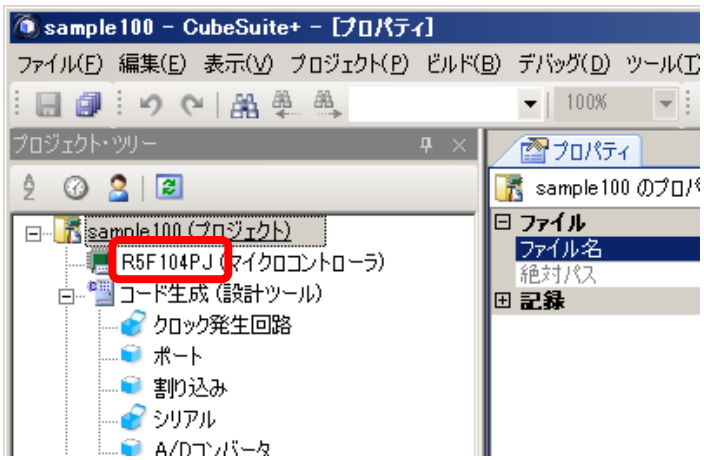


図 3-1 マイコンの型名確認

次に、マイコン定義ファイルを作成します。マイコン定義ファイルは API_Builder_SAIC101.exe が格納されているフォルダと同一階層にある「Chips」の中に作成します。「Chips」内にある「Template.csv」ファイルをコピーし、ファイル名を型名(ここでは R5F104PJ とします)に変更してください。

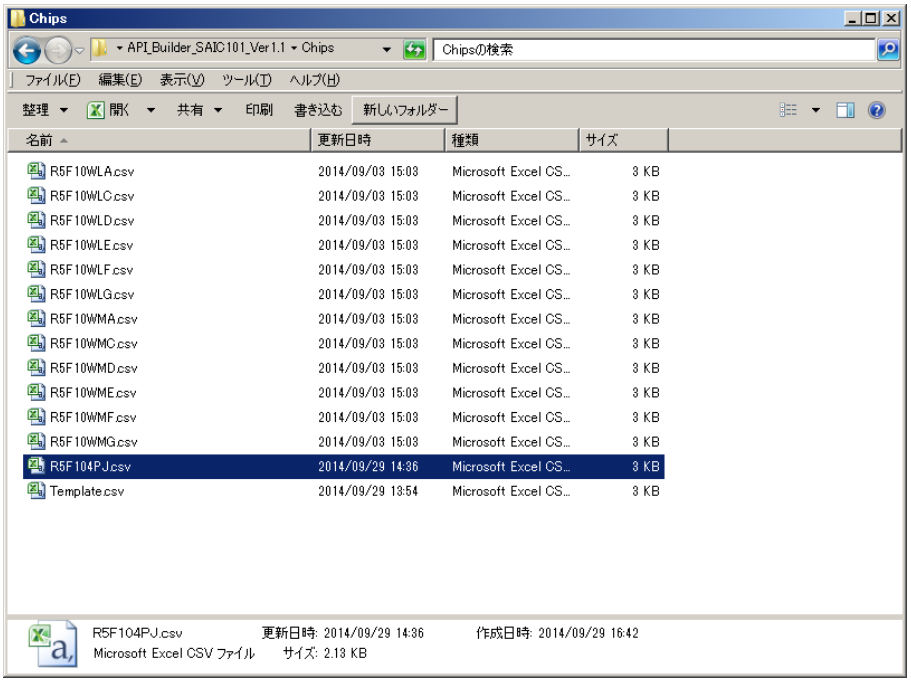


図 3-2 マイコン定義ファイル格納フォルダ

3.2　マイコン定義ファイルの編集

3.1 章で作成したマイコン定義ファイルを編集します。編集は Excel またはテキストエディタで行うことができます。ここでは、Excel を使用して編集が必要な箇所を示します。

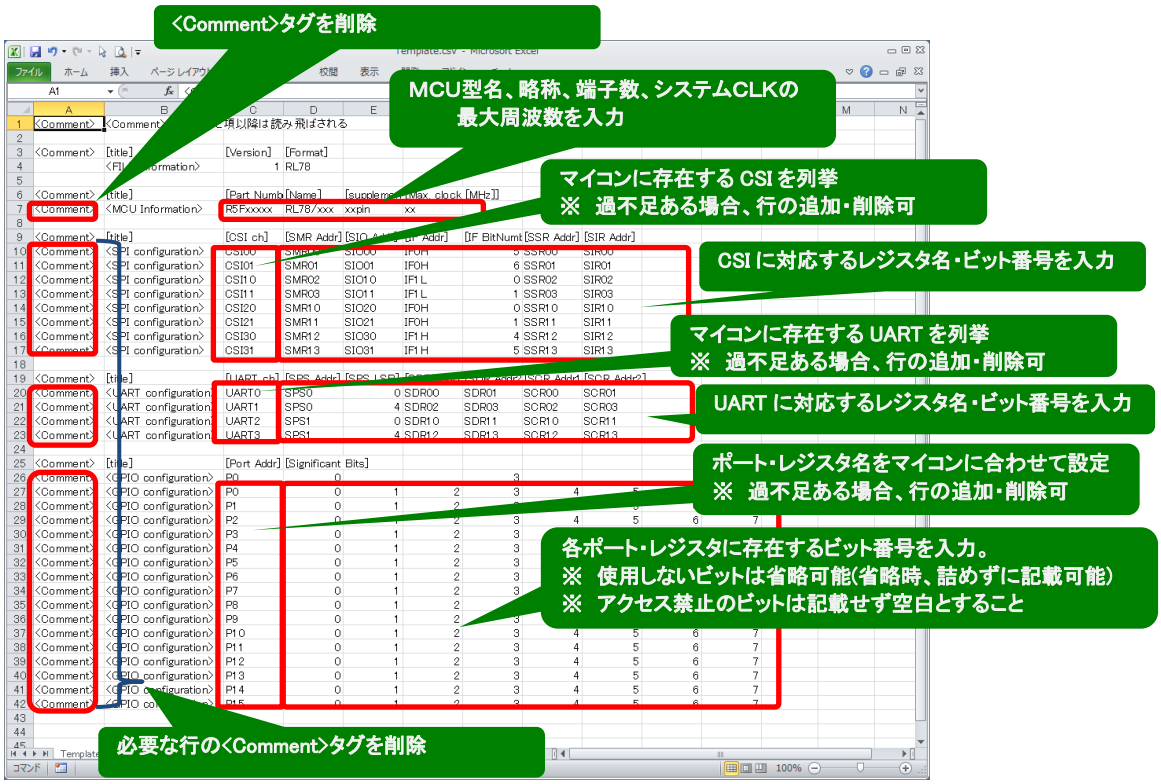


図 3-3 編集すべき箇所

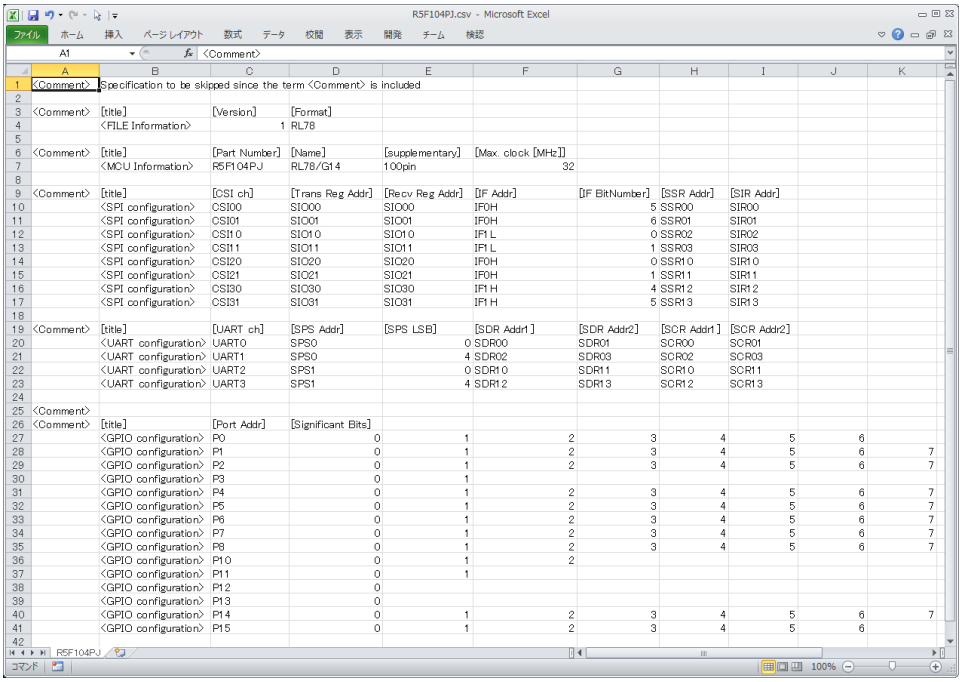


図 3-4 マイコン定義ファイルの変更例

3.3 マイコン定義ファイルの追加確認

3.2 節で編集したファイルを保存し、API Builder SAIC101 を起動します。

(マイコン定義ファイルは、API Builder SAIC101 の起動時に読み込まれます。既に起動している場合は一度終了してください。)

マイコン定義ファイルが正しく編集されていない場合、起動時に図 3-5 のメッセージが表示されます。本エラー発生時にはマイコン定義ファイルが読み込まれないためマイコンが追加できません。再度、マイコン定義ファイルを修正してください。

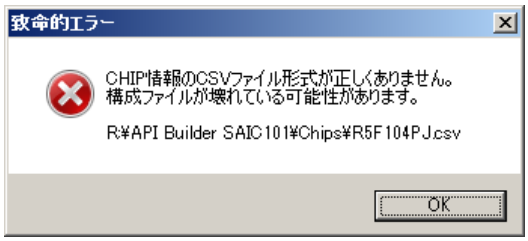


図 3-5 マイコン定義ファイルの編集に失敗した場合の表示

正常に API Builder SAIC101 が起動し、CubeSuite+または e2studio で作成したプロジェクトを読み込み、図 3-6 に示すようにマイコン型名が表示されると確認作業は完了となります。

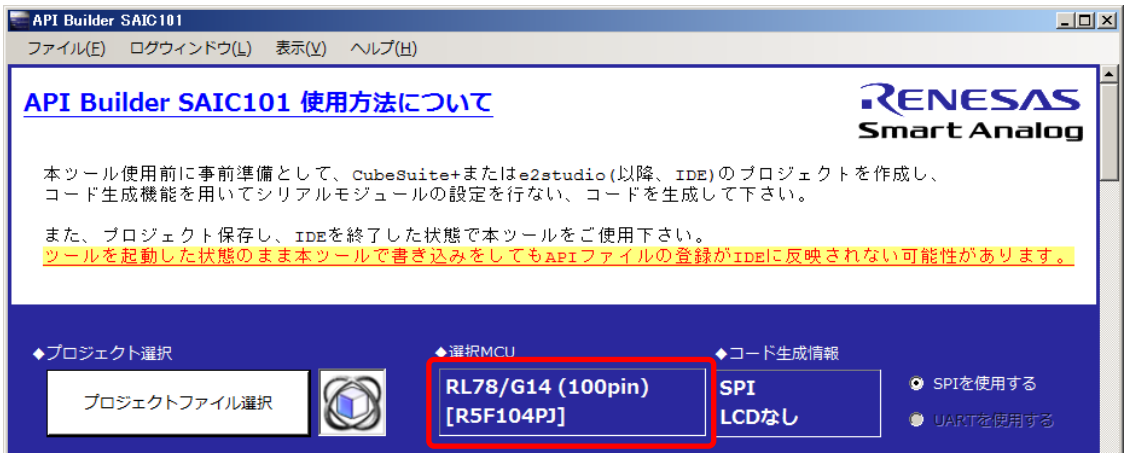


図 3-6 新たなマイコン定義ファイルによりマイコンが追加された事を確認

以降は、サンプルコード導入手順書兼 API Builder SAIC101 仕様書(R21AN0012JJ)を参考に設定を行ってください。

4. RL78 ファミリ用コード生成非対応マイコンへの置き換え手順

SAIC101 API は組み込むプロジェクトに CubeSuite+用 RL78 ファミリ, 78K0R, 78K0 コード生成(CubeSuite+ Code_Generator for RL78_78K)を用いて RL78/L13 用の設定を使用することを前提としています。このため RL78 ファミリ用コード生成に対応していないマイコンへ置き換える場合は、コード生成にて出力される関数と同等の処理、入出力の関数の作成が必要となります。また、ユーザーシステムに合わせてグローバル変数の追加や定義の修正が必要になります。

4.1 API 変更内容

ここでは、API ファイル(表 4-1)への変更内容を説明します。

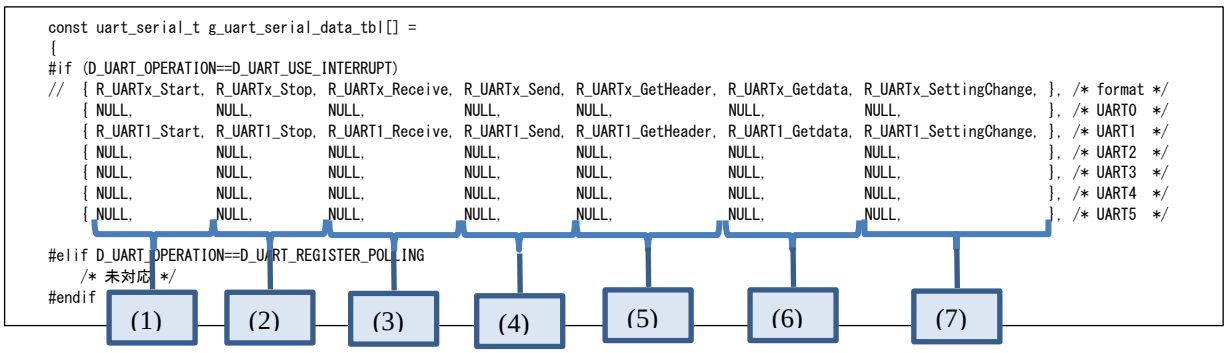
表 4-1 API ファイル

通信方式	API ファイル
UART	r_sa_uart_control_register.c / r_sa_uart_control_register.h / r_sa_uart_control_register_user.c
SPI	r_sa_spi_control_register.c / r_sa_spi_control_register.h / r_sa_spi_control_register_user.c

4.1.1 シリアルモジュール情報格納グローバル定数置き換え

通信方式に従い、UART をご使用の場合は図 4-1 を、SPI をご使用の場合であれば図 4-2 を参考に、シリアルモジュール情報格納グローバル定数の置き換えを行ってください。この配列の要素番号はシリアルモジュール情報格納グローバル変数指定用列挙体で指定されます。置き換えの際は、使用対象のチャネル番号に対応した要素番号のシリアルモジュール関数を置き換えてください。なお、ユーザーシステムに応じたシリアルモジュール関数の生成については 4.2.1 章を参照ください。

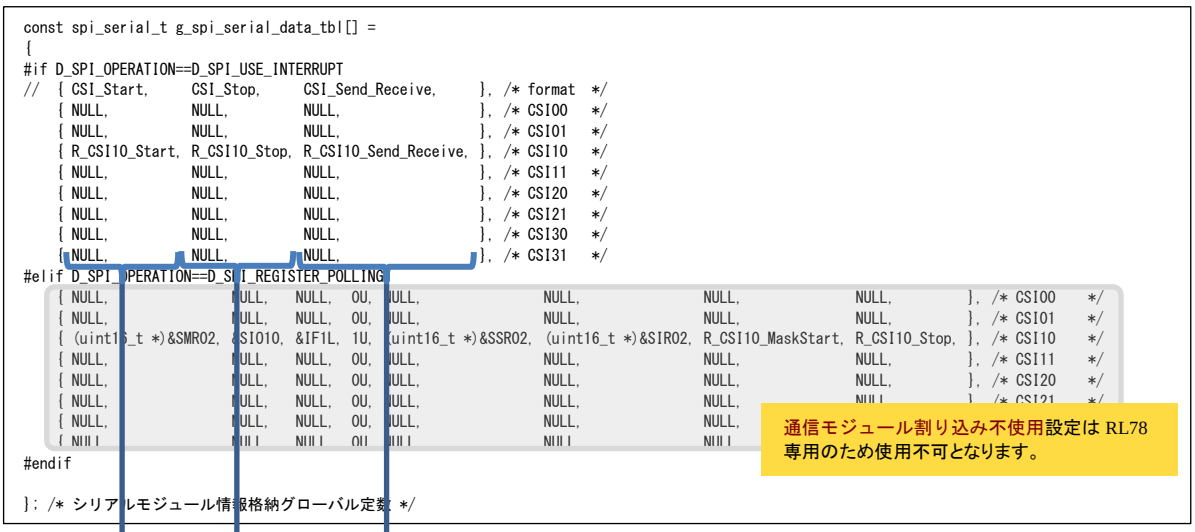
対象ファイル：r_sa_uart_control_register_user.c



No.	関数名フォーマット	処理	備考
(1)	R_UARTx_Start	UARTx モジュール動作開始	4.2.1、4.3.1 章を参照
(2)	R_UARTx_Stop	UARTx モジュール動作停止	
(3)	R_UARTx_Receive	UARTx 受信関数	
(4)	R_UARTx_Send	UARTx 送信関数	
(5)	R_UARTx_GetHeader	UARTx ヘッダ取得関数	
(6)	R_UARTx_Getdata	UARTx データ取得関数	
(7)	R_UARTx_SettingChange	UARTx 設定変更関数	

図 4-1 シリアルモジュール情報格納グローバル定数置き換え箇所(UART)

対象ファイル：r_sa_spi_control_register_user.c



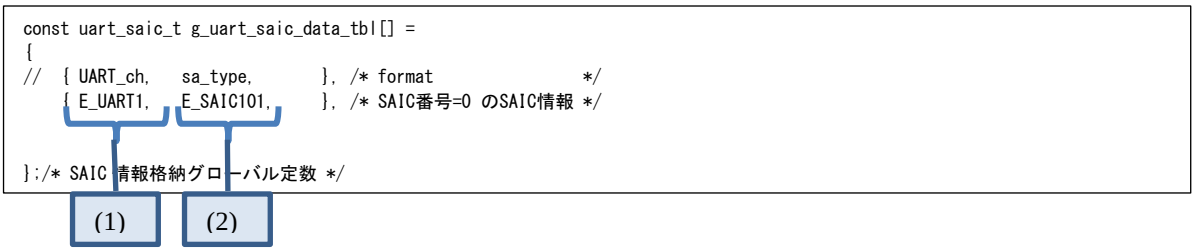
関数名フォーマット	処理	備考
R_CSIxx_Start,	CSIxx の動作開始処理	4.2.1、4.3.2 章を参照
R_CSIxx_Stop	CSIxx の動作停止処理	
R_CSIxx_Send_Receive	CSIxx のデータ送受信関数	

図 4-2 シリアルモジュール情報格納グローバル定数置き換え箇所(SPI)

4.1.2 SAIC 情報格納グローバル定数置き換え

通信方式に従い、UART をご使用の場合は図 4-3 を、SPI をご使用の場合は図 4-4 を参考に、SAIC 情報格納グローバル定数の置き換えを行ってください。

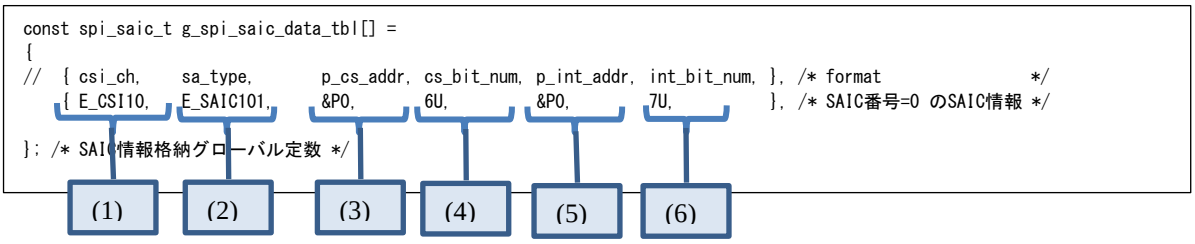
対象ファイル：r_sa_uart_control_register_user.c



No.	記述例	内容
(1)	E_UART1	4.1.1 章で置き換えた関数が格納された配列の要素番号をシリアルモジュール情報格納グローバル定数指定用列挙体 e_uart_ch_t で指定してください。
(2)	E_SAIC101	(1)で指定したシリアルモジュールに接続されている SAIC の種類を SAIC 型名指定用列挙体 e_saic_type_t で指定してください。

図 4-3 SAIC 情報格納グローバル定数置き換え箇所(UART)

対象ファイル：r_sa_spi_control_register_user.c



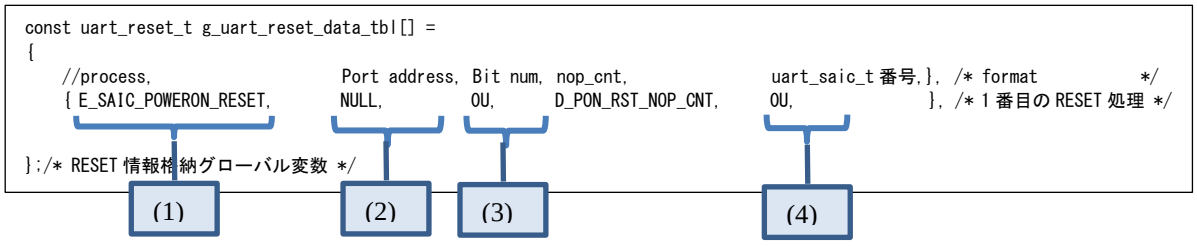
No.	記述例	内容
(1)	E_CSI10	4.1.1 章で置き換えた関数が格納された配列の要素番号をシリアルモジュール情報格納グローバル定数指定用列挙体 e_csi_ch_t で指定してください。
(2)	E_SAIC101	(1)で指定したシリアルモジュールに接続されている SAIC の種類を SAIC 型名指定用列挙体 e_saic_type_t で指定してください。
(3)	&P0	CS 端子に接続しているポートレジスタのアドレスを記述してください。
(4)	6U	CS 端子に接続しているポートレジスタのビット番号を記述してください。
(5)	&P0	INT 端子に接続しているポートレジスタのアドレスを記述してください。
(6)	7U	INT 端子に接続しているポートレジスタのビット番号を記述してください。

図 4-4 SAIC 情報格納グローバル定数置き換え箇所(SPI)

4.1.3 RESET 情報格納グローバル定数置き換え

通信方式に従い、UART をご使用の場合は置き換え不要、SPI をご使用の場合であれば図 4-5 を参考に、リセット情報格納グローバル定数の置き換えを行ってください

対象ファイル：r_sa_spi_control_register_user.c



No.	記述例	内容
(1)	E_SAIC_POWERON_RESET	リセット方法をリセット方法指定用列挙体 e_reset_process_t で指定してください。
(2)	NULL	リセット方法が外部リセットの場合、RESET 端子に接続しているポートレジスタのアドレスを 記述。それ以外の場合は NULL を記述してください。
(3)	0U	リセット方法が外部リセットの場合、RESET 端子に接続しているポートレジスタのビット番号を記述。それ以外の場合は 0 を記述してください。
(4)	0U	内部リセット処理使用時の SAIC 番号。内部リセット処理以外の時は現在の API では使用していませんが、SAIC 番号を指定してください。(API Builder SAIC101 で使用しています。)

図 4-5 RESET 情報格納グローバル定数置き換え箇所(SPI)

4.1.4 includeするヘッダファイルの修正

API は、RL78 のコード生成機能で出力された型や関数を使用するため、コード生成機能で出力されるヘッダファイルをインクルードしています。これらをユーザーの作成したシリアル関数が定義されたヘッダファイル、また、4.2.4 章、4.2.5 章で示す内容を反映したヘッダファイルに置き換えてください。

表 4-2 修正対象ファイル

通信方式	ファイル
UART	r_sa_uart_control_register_user.c r_sa_uart_control_register.c
SPI	r_sa_spi_control_register_user.c r_sa_spi_control_register.c

表 4-3 includeするヘッダファイル必須変更項目

項目	既存の定義	変更方法
シリアル通信関数定義	#include "r_cg_sau.h"	ユーザの作成したシリアル関数が定義されたヘッダファイルに置き換え
型定義	#include "r_cg_macrodriver.h"	4.2.4 章、4.2.5 章で示す内容を反映したヘッダファイルに置き換え

4.1.5 ユーザー環境依存設定用マクロ宣言の修正

API 仕様書(R21AN0015JJ)の「4.2 ユーザー環境依存設定用マクロ宣言」の内容を参考に使用するマイコンに応じて適切な設定を行ってください。

また、API では内部のソフトウェアウエイトのループ回数を、ループ処理に必要な最低限のステップ数から算出しており、RL78 マイコンで必要な 7 クロックとしています。ユーザーの使用するマイコンによりループ処理にかかるステップ数が異なりますので、表 4-4、表 4-5 に示す定義を修正してください。

対象ファイル：r_sa_uart_control_register_user.c

表 4-4 ソフトウェアウエイトループ回数定義(UART)

既存の定義	変更方法
#define D_PON_RST_NOP_CNT ((uint32_t)((D_WAIT_PON_RST_TIME_MS / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	**7.0F** 部分を、使用するマイコンのループ処理に必要な最低限のステップ数に修正してください。入力範囲は float 型
#define D_UART_4800BPS_HALF_BIT ((uint16_t)((D_UART_4800BPS_HALF_BIT_MS / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	
#define D_UART_250kbps_HALF_BIT ((uint16_t)((D_UART_250kbps_HALF_BIT_MS / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	
#define D_UART_5MS_NOP_CNT ((uint16_t)((D_UART_5MS / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	
#define D_UART_1800US_NOP_CNT ((uint16_t)((D_UART_1800US / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	
#define D_UART_270US_NOP_CNT ((uint16_t)((D_UART_270US / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	
#define D_UART_250US_NOP_CNT ((uint16_t)((D_UART_250US / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	

対象ファイル：r_sa_spi_control_register_user.c

表 4-5 ソフトウェアウエイトループ回数定義(SPI)

既存の定義	変更方法
#define D_PON_RST_NOP_CNT ((uint32_t)((D_WAIT_PON_RST_TIME_MS / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	**7.0F** 部分を、使用するマイコンのループ処理に必要な最低限のステップ数に修正してください。入力範囲は float 型
#define D_HARD_RESET_NOP_CNT ((uint32_t)((D_WAIT_HARD_RESET_TIME_MS / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	
#define D_SPI_5MS_NOP_CNT ((uint16_t)((D_SPI_5MS / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	
#define D_SPI_1800US_NOP_CNT ((uint16_t)((D_SPI_1800US / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	
#define D_SPI_820US_NOP_CNT ((uint16_t)((D_SPI_820US / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	
#define D_SPI_250US_NOP_CNT ((uint16_t)((D_SPI_250US / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	
#define D_SPI_3US_NOP_CNT ((uint16_t)((D_SPI_3US / (1.0F / D_CPU_CLK_MHZ)) * 1000.0F / **7.0F**))	

4.2 ユーザーが作成したソースファイルに必要な作業

ユーザーが作成したソースファイルへの変更内容を説明します。

4.2.1 シリアル通信用関数の作成

API は、シリアル通信機能のハードウェアアクセスに、RL78 のコード生成機能で出力された関数を用いています。また、UART に関してはハードウェアアクセスを行う追加の関数を作成しています。置き換えに当たってはこれらの関数に相当する関数が必要となります。下記に API を使用するために必要な関数を記載します。関数の詳細は 4.3 章を参照してください。

表 4-6 シリアル通信用関数一覧

通信方式	サンプル上の関数名	引数	戻り値	内容
UART	R_UART1_Start	なし	なし	UART 通信を待機状態にします。
	R_UART1_Stop	なし	なし	UART 通信を終了します。
	R_UART1_Receive	uint8_t * const rx_buf, uint16_t rx_num	MD_STATUS	データの UART 受信を開始します。
	R_UART1_Send	uint8_t * const tx_buf, uint16_t tx_num	MD_STATUS	データの UART 送信を開始します。
	R_UART1_GetHeader	uint8_t *packet_data uint8_t rx_buffer[] uint16_t read_pos	uint8_t	受信したデータがあった場合、ヘッダデータ解析のため 1 バイト取得します。
	R_UART1_Getdata	uint16_t rx_cnt	uint8_t	指定した受信データ数以上のデータを受信しているかチェックします。
	R_UART1_SettingChange	uint8_t setting	なし	UARTモジュールの通信設定を変更します。
SPI	R_CSI10_Start	なし	なし	3 線シリアル I/O 通信を待機状態にします。
	R_CSI10_Stop	なし	なし	3 線シリアル I/O 通信を終了します。
	R_CSI10_Send_Receive	uint8_t * const tx_buf, uint16_t tx_num, uint8_t * const rx_buf	MD_STATUS	データの CSI 送受信を開始します。

4.2.2 API 関数の呼び出し、グローバル変数への代入

- API 関数を使用する際は最初に SmartAnalog 初期化関数 (R_SAIC_SPI_Init, R_SAIC_UART_Init)を実行してください。
- 初期状態の SAIC101 のボー・レートが分からない場合、SmartAnalog 初期化関数の後かつ UART 通信開始前に、SAIC101 の UART のボー・レート、パリティ等の通信設定をマイコンと SAIC101 とで合わせるため通信設定ネゴシエーション関数(R_SAIC_UART_Negotiation)を呼び出す必要があります。
- UART 受信関数の引数で指定されたバイト数の受信を終えたタイミングで、UART 受信完了フラグ(g_uart_rx_end_flag)の e_uart_ch_t に対応するビットをセットしてください。
- UART 送信関数の引数で指定されたバイト数の送信を終えたタイミングで、UART 送信完了フラグ(g_uart_tx_end_flag)の e_uart_ch_t に対応するビットをセットしてください。
- 3 線式シリアル(クロック同期シリアル)通信の送受信関数の引数で指定されたバイト数の送受信を終えたタイミングで、e_csi_ch_t に対応した引数を指定して CS 端子を High にする API 関数 R_SAIC_SPI_CSDisable を呼び出す必要があります。また、必要に応じてシリアル通信モジュールを停止する処理を追加してください。
- 3 線式シリアル通信でオーバラン・エラーが発生した場合には、e_csi_ch_t に対応した引数を指定し CS 端子を High にする API 関数 R_SAIC_SPI_CSDisable の呼び出し、およびオーバラン・エラーフラグ(g_csi_overrun_flag)へ e_csi_ch_t に対応するビットのセットを行ってください。また、必要に応じてシリアル通信モジュールを停止する処理を追加してください。

4.2.3 シリアル通信にて API が使用するグローバル変数の定義

API ではシリアルの通信状態や、エラー状態を表 4-7 のグローバル変数によって判定しています。これらのグローバル変数はユーザー側で宣言し、API 関数から参照できるようにしてください。また、4.2.2 章にも示したように、対応する内容の処理をユーザーのソースコードへ追加してください。

表 4-7 シリアル通信にて API が使用するグローバル変数一覧

通信方式	型名	変数名	内容	使用する API 関数
UART	uint8_t	g_uart_tx_end_flag	UART 送信完了フラグ。送信完了した場合、e_uart_ch_t に対応するビットをセット。API でフラグをクリアする。	r_saic_uart_send_command
UART	uint8_t	g_uart_rx_end_flag	UART 受信完了フラグ。受信完了した場合、e_uart_ch_t に対応するビットをセット。API でフラグをクリアする。	r_saic_uart_write_read r_saic_uart_send_command r_saic_uart_get_response R_SAIC_UART_ADC_GetResult R_SAIC_UART_ADC_GetReceive r_saic_uart_flash_read r_saic_uart_flash_write r_saic_uart_flash_all_erase r_saic_uart_all_flash_to_reg r_saic_uart_buffer_refresh R_SAIC_UART_FLASH_WRITE_01H R_SAIC_UART_FLASH_WRITE_1FH
SPI	uint8_t	g_csi_overrun_flag	SPI オーバーランフラグ。オーバーランが発生した場合、e_csi_ch_t に対応するビットをセット。API でフラグをクリアする。	r_saic_spi_overrun_err_check

4.2.4 幅指定整数型

API では、C99 に準拠した幅指定整数型(intN_t,、uintN_t)を用いています(宣言は stdint.h ではなくコード生成機能が生成する macrodriver.h で行っています)。お使いのコンパイラで幅指定整数型をサポートしていない場合(stdint.h 定義がない場合)には、表 4-8 の定義を追加してください。

表 4-8 使用する幅指定整数型の一覧

型	説明	定義
int8_t	8 ビット幅符号付き整数	typedef signed char int8_t;
uint8_t	8 ビット幅符号無し整数	typedef unsigned char uint8_t;
int16_t	16 ビット幅符号付き整数	typedef signed short int16_t;
uint16_t	16 ビット幅符号無し整数	typedef unsigned short uint16_t;
int32_t	32 ビット幅符号付き整数	typedef signed long int32_t;
uint32_t	32 ビット幅符号無し整数	typedef unsigned long uint32_t;

4.2.5 RL78 マイコン固有の書き方の修正

API 関数内では、一部 RL78 マイコン、コンパイラに特化した記述、コード生成で使用する定義を使用しています。他のマイコンに置き換える際には対応する記述に置き換えてください。

表 4-9 固有の定義一覧

内容	関数/変数名/定義	説明
RL78 固有記述	NOP()	RL78 コンパイラ用 nop 命令実行記述。 他のマイコンの場合はマクロ定義によって適切な定義に置き換える必要があります。
コード生成で使用する定義	MD_STATUS	シリアル関数の戻り値型の内容の定義。 他のマイコンの場合、以下の定義を追加してください typedef unsigned short MD_STATUS; 使用する戻り値は、「MD_OK」「MD_ARGERROR」です。 宣言を以下に示します。 #define MD_STATUSBASE (0x00U) /* register setting OK */ #define MD_OK (MD_STATUSBASE + 0x00U) /* Error list definition */ #define MD_ERRORBASE (0x80U) /* error agrument input error */ #define MD_ARGERROR (MD_ERRORBASE + 0x01U)

4.3 関数仕様

4.3.1 UART 通信

UART 通信で必要な関数の仕様を以下に記します。ここでは UART1 を例として説明します。

[関数名] R_UART1_Start

概要	UART1 動作開始処理
ヘッダ	r_cg_macrodriver.h、 r_cg_sau.h、 r_cg_userdefine.h
宣言	void R_ UART1_Start(void)
説明	シリアル・アレイ・ユニットの該当チャネルを動作開始させ、通信待機状態にします。
引数	なし
リターン値	なし
参考文献	RL78/G13 シリアル・アレイ・ユニット（UART 通信）R01AN0459JJ0300
補足説明	シリアル通信機能の動作開始と停止を API で行わせなくて良い場合、本関数の作成は不要です。その時は、必ずシリアルモジュール情報格納グローバル定数の該当箇所にはダミー関数を登録してください。

[関数名] R_UART1_Receive

概要	UART1 受信ステータス初期化関数
ヘッダ	r_cg_macrodriver.h、 r_cg_sau.h、 r_cg_userdefine.h
宣言	MD_STATUS R_UART1_Receive(uint8_t *rxbuf, uint16_t rxnum)
説明	UART1 受信の初期設定をします。
引数	uint8_t *rxbuf : [受信データバッファのアドレス] uint16_t rxnum : [受信データバッファのサイズ]
リターン値	[MD_OK]の場合：受信設定完了 [MD_ARGERROR]の場合：受信設定失敗
参考文献	RL78/G13 シリアル・アレイ・ユニット（UART 通信）R01AN0459JJ0300
補足説明	レジスタリードなどの API が呼ばれると、各 API はこの関数を呼び出し UART 通信機能は受信待ち状態となり、指定されたバイト数のデータを受信するまでバックグランドで受信動作を続けます。 受信データを格納するバッファはユーザー側で準備し、ポインタで渡す必要があります。 引数 rxnum で指定された回数の通信を行った後に、4.2.2 章で説明されている通信終了時の処理が必要となります。

[関数名] R_UART1_Send

概要	UART1 データ送信関数
ヘッダ	r_cg_macrodriver.h、 r_cg_sau.h、 r_cg_userdefine.h
宣言	MD_STATUS R_UART1_Send(uint8_t* txbuf, uint16_t txnum)
説明	UART1 受信の初期設定をします。
引数	uint8_t *txbuf : [送信データバッファのアドレス] uint16_t txnum : [送信データバッファのサイズ]
リターン値	[MD_OK]の場合：送信設定完了 [MD_ARGERROR]の場合：送信設定失敗
参考文献	RL78/G13 シリアル・アレイ・ユニット（UART 通信）R01AN0459JJ0300
補足説明	殆どの API がこの関数を呼び出します。各 API はこの関数を呼び出すと UART 通信機能は送信待ち状態となり、指定されたバイト数のデータを送信するまでバックグランドで送信動作を続けます。 送信データを格納するバッファはユーザー側で準備し、ポインタで渡す必要があります。 引数 txnum で指定された回数の通信を行った後に、4.2.2 章で説明されている通信終了時の処理が必要となります。

[関数名] R_UART1_Stop

概要	UART1 動作停止処理
ヘッダ	r_cg_macrodriver.h、 r_cg_sau.h、 r_cg_userdefine.h
宣言	void R_UART1_Stop (void)
説明	シリアル・アレイ・ユニットの該当チャネルを動作停止させます。
引数	なし
リターン値	なし
参考文献	RL78/G13 シリアル・アレイ・ユニット（UART 通信）R01AN0459JJ0300
補足説明	シリアル通信機能の動作開始と停止を API で行わせなくて良い場合、本関数の作成は不要です。その時は、必ずシリアルモジュール情報格納グローバル定数の該当箇所にはダミー関数を登録してください。

[関数名] R_UART1_SettingChange

概要	UART1 設定変更処理
ヘッダ	r_cg_macrodriver.h、r_cg_sau.h、r_cg_userdefine.h、r_sa_uart_control_register.h
宣言	void R_UART1_SettingChange(uint8_t setting)
説明	引数 setting の値によって該当 UART チャネルの 設定を変更する
引数	uint8_t setting
リターン値	なし
参考文献	ー
補足説明	引数 setting の値によって UART 通信機能の通信フォーマットを変更します。

引数 setting の値	UART 通信設定
0	4800bps, Parity=None
1	4800bps, Parity=Odd
2	4800bps, Parity=Even
3	250000bps, Parity=None
4	250000bps, Parity=Odd
5	250000bps, Parity=Even

[関数名] R_UART1_GetHeader

概要	ヘッダデータ取得関数
ヘッダ	r_cg_macrodriver.h、r_cg_sau.h、r_cg_userdefine.h、r_sa_uart_control_register.h
宣言	uint8_t R_UART1_GetHeader(uint8_t *packet_data, uint8_t rx_buffer[], uint16_t read_pos)
説明	受信したデータがあった場合、ヘッダデータ解析のため 1 バイト取得します。
引数	uint8_t *packet_data :ヘッダ格納ポインタ uint8_t rx_buffer[] :受信データバッファのアドレス uint16_t read_pos :受信データバッファ読み出し位置
リターン値	uint8_t : 0=Invalid, 1=Valid
参考文献	RL78/G13 シリアル・アレイ・ユニット（UART 通信）R01AN0459JJ0300
補足説明	UART 通信機能で受信待ち状態になると、API の受信データ解析処理によって必ずこの関数が呼び出されます。この関数は新規受信が 1 バイト以上あった場合、ヘッダデータを解析するために未読データの中で最も古い 1 バイトのデータを受信バッファからヘッダ格納ポインタに値を代入します。SAIC101 の受信データタイプと一致するか、タイムアウトするまでループ処理で呼び出されます。 関数内ではグローバル変数 g_uart1_rx_count を使用します。この変数はコード生成により、シリアルモジュールファイル [r_cg_sau.c]、または[r_cg_serial.c]に宣言されます。 UART1 受信ステータス初期化関数（R_UART1_Receive）を呼び出すことで値を初期化し、受信割り込み関数内で受信したバイト数を加算します。

```
uint8_t R_UART1_GetHeader( uint8_t *packet_data, uint8_t rx_buffer[], uint16_t
read_pos )
{
    uint8_t ret = 0U;

    if (read_pos < g_uart1_rx_count)
    {
        *packet_data = rx_buffer[read_pos];
        ret = 1;
    }

    return (ret);
}
```

引数で指定した受信データバッファ読み出し位置と、現在の受信データカウンタ値（g_uart1_rx_count）を比較

読み出し位置より後に受信データが格納されていれば読み出し位置のデータをポインタに格納

読み出し位置より後に受信データが格納されていれば戻り値 1 を返す。

[関数名] R_UART1_GetData

概要	指定データ数受信完了チェック関数
ヘッダ	r_cg_macrodriver.h、r_cg_sau.h、r_cg_userdefine.h、r_sa_uart_control_register.h
宣言	uint8_t R_UART1_Getdata(uint16_t rx_cnt)
説明	指定した受信データ数以上のデータを受信しているかチェックします。
引数	uint16_t rx_cnt : 指定受信データ数
リターン値	uint8_t : 0=Invalid, 1=Valid
参考文献	ー
補足説明	UART 通信機能で受信待ち状態になり、API の受信データ解析処理でヘッダデータ解析後、SAIC101 の受信データタイプ 1 またはタイプ 2 の時に、引数 rx_cnt で指定された受信データ数のデータが受信されているかをチェックします。SAIC101 から受信データタイプと合致するデータ数を受信するか、タイムアウトするまでループ処理で呼び出されます。

```
uint8_t R_UART1_Getdata(uint16_t rx_cnt)
{
    uint8_t ret = 0U;

    if (rx_cnt <= g_uart1_rx_count)
    {
        ret = 1U;
    }

    return (ret);
}
```

引数の指定受信データ数と、現在の受信データカウンタ値（g_uart1_rx_count）を比較

指定受信データ数以上受信していれば、戻り値 1 を返す。

4.3.2 SPI 通信

SPI 通信で必要な関数の仕様を以下に記します。ここでは CSI10 を例として説明します。

[関数名] R_CSI10_Start

概要	CSI10 の動作開始処理
ヘッダ	r_cg_macrodriver.h、r_cg_sau.h、r_cg_userdefine.h
宣言	void R_CSI10_Start(void)
説明	SAU の該当チャネル を CSI として動作開始させ、通信待機状態にします。
引数	なし
リターン値	なし
参考文献	RL78/G13 シリアル・アレイ・ユニット 3 線シリアル I/O （SPI マスタ送受信）【開発環境：CubeSuite+、IAR、e2 studio】R01AN1367JJ0201
補足説明	シリアル通信機能の動作開始と停止を API で行わせなくて良い場合、本関数の作成は不要です。その時は、必ずシリアルモジュール情報格納グローバル定数の該当箇所にはダミー関数を登録してください。

[関数名] R_CSI10_Send_Receive

概要	CSI10 のデータ送受信関数
ヘッダ	r_cg_macrodriver.h、r_cg_sau.h、r_cg_userdefine.h
宣言	MD_STATUS R_CSI10_Send_Receive(uint8_t *txbuf、uint16_t txnum、uint8_t *rxbuf)
説明	CSI10 のデータ送受信設定をします。
引数	uint8_t *txbuf : [送信データバッファのアドレス] uint16_t txnum : [送信データバッファのサイズ] uint8_t *rxbuf : [受信データバッファのアドレス]
リターン値	[MD_OK]の場合：送受信設定完了 [MD_ARGERROR]の場合：送受信設定失敗
参考文献	RL78/G13 シリアル・アレイ・ユニット 3 線シリアル I/O （SPI マスタ送受信）【開発環境：CubeSuite+、IAR、e2 studio】R01AN1367JJ0201
補足説明	送受信データを格納するバッファはユーザー側で準備し、ポインタで渡す必要があります。 引数 txnum で指定された回数の通信を行った後に、4.2.2 章で説明されている通信終了時の処理が必要となります。

[関数名] R_CSI10_Stop

概要	CSI10 の動作停止処理
ヘッダ	r_cg_macrodriver.h、r_cg_sau.h、r_cg_userdefine.h
宣言	void R_CSI10_Stop(void)
説明	該当する CSI チャネル を停止させます。
引数	なし
リターン値	なし
参考文献	ー
補足説明	シリアル通信機能の動作開始と停止を API で行わせなくて良い場合、本関数の作成は不要です。その時は、必ずシリアルモジュール情報格納グローバル定数の該当箇所にはダミー関数を登録してください。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
Rev.1.00	2014.09.30	---	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。 たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。 なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。
注2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/