RL78/F24

RS-CANFD lite モジュール Software Integration System

要旨

本アプリケーションノートは RS-CANFD lite モジュールについて説明します。

動作確認デバイス

RL78/F24 グループ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

対象コンパイラ

- Renesas Electronics C/C++ Compiler Package for RL78 Family
- IAR C/C++ Compiler for Renesas RL78

目次

1. 概要

本モジュールは、RS-CANFD lite モジュールを使用して CAN フレームの送受信を行うための手段を提供します。

2. API 情報

本モジュールは、下記の条件で動作を確認しています。

2.1 ハードウェアの要求

ご使用になる MCU が以下の機能をサポートしている必要があります。

- RS-CANFD lite

2.2 ハードウェアリソースの要求

RS-CANFD lite に加えて、以下が要求されます。

- CAN チャネルに割り当てられた端子 2 本

2.3 ソフトウェアの要求

本モジュールは以下のモジュールに依存しています。

- ボードサポートパッケージ（r_bsp）v1.20 以上

2.4 制限事項

本モジュールがサポートしていない RS-CANFD lite 機能を以下に示します。

- PNF (Pretended Network Filter List)
- 受信ルールのエントリ無効化
- GRSTC レジスタによるグローバルリセット
- ワンショット送信機能
- 通信エラー・カウンタ
- 通信完了カウンタ
- テスト機能（リッスンオンリモード、ループバック、RAM テスト、など）
- CAN-RAM の ECC 機能

2.5 対応ツールチェーン

本モジュールは以下に示すツールチェーンで動作確認を行っています。

- Renesas CS+ for CC V8.07.00
- IAR Embedded Workbench for Renesas RL78 4.21.3

2.6 ヘッダファイル

すべての API 呼び出しとそれをサポートするインターフェース定義は r_rscanfd_rl78_if.h に記載されています。r_rscanfd_rl78_config.h ファイルで、ビルド時に設定変更可能なコンフィギュレーションオプションを選択あるいは定義できます。

上記 2 ファイルはユーザアプリケーションにインクルードする必要があります。

2.7 整数型

本モジュールでは ANSI C99 を使用しています。これらの型は stdint.h で定義されています。

2.8 コンパイル時の設定

本モジュールのコンフィギュレーションオプションの設定は、r_rscanfd_rl78_config.h で行います。

オプション名および設定値に関する説明を下表に示します。

定義	ﾃﾞﾌｫﾙﾄ値	説明	対象レジスタ
CAN_CFG_PARAM_CHECKING_ENABLE	BSP_CFG_PARAM_CHECKING_ENABLE	この定義を”1”に設定するとパラメータチェック処理のコードを生成し、”0”に設定すると生成しません。 “BSP_CFG_PARAM_CHECKING_ENABLE”の場合、BSP での設定に依存します。	-
CAN_CFG_CLOE_AND_FDOE	0	この定義を”1”に設定すると FD 専用モードが有効になり、”2”に設定するとクラシカル CAN 専用モードが有効になります。	C0FDCFGH, C0FDCFGL
CAN_CFG_REFE	0	この定義を”1”に設定すると受信エッジフィルタが有効になります。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	C0FDCFGH, C0FDCFGL
CAN_CFG_TDCO	0x00	TDC (Transceiver Delay Compensation)のオフセットを設定します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	C0FDCFGH, C0FDCFGL
CAN_CFG_ESIC	0	ESI を設定します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	C0FDCFGH, C0FDCFGL
CAN_CFG_TDCE_AND_TDCOC	0	この定義を”1”または”2”に設定すると TDC (Transceiver Delay Compensation)が有効になります。(1: SSP offset = 測定値+ CAN_CFG_TDCO 2: SSP offset = CAN_CFG_TDCO) 詳細はユーザーズマニュアル ハードウェア編を参照してください。	C0FDCFGH, C0FDCFGL
CAN_CFG_ITRCP	0	FIFO 用インターバルタイマのクロック源の分周比を設定します。 “0”に設定するとタイマは無効になります。	GCFGH, GCFGL
CAN_CFG_TSSS	0	タイムスタンプカウンタのクロック源を選択します。	GCFGH, GCFGL

		“0”に設定すると pclk、”1”に設定するとビットタイムクロックを選択します。CAN-FD 通信を使用する場合は”1”に設定しないでください。	
CAN_CFG_TSP	0	タイムスタンプカウンタに使用されるクロック源の分周比を設定します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	GCFGH, GCFGL
CAN_CFG_CMPOC	0	受信したメッセージのペイロードが受信メッセージバッファのペイロードサイズより大きい場合の動作を選択します。 “0”に設定すると受信メッセージを廃棄、”1”に設定するとメッセージバッファのペイロードサイズを超過する分は切り捨てられます。	GCFGH, GCFGL
CAN_CFG_DCS	0	CAN 通信のクロック源を選択します。 “0”に設定すると内部クロック、”1”に設定すると外部クロック(X1 クロックダイレクト)を選択します。	GCFGH, GCFGL
CAN_CFG_MME	0	この定義を”1”に設定するとミラーモードが有効になります。	GCFGH, GCFGL
CAN_CFG_DRE	0	この定義が”1”で CAN_CFG_DCE が”1”の場合、DLC チェックを通過すると受信ルールの DLC 値 (CAN_CFG_RULEx_GAFLDLC の設定値)がバッファに格納されます。	GCFGH, GCFGL
CAN_CFG_DCE	0	この定義を”1”に設定すると DLC チェックが有効になります。	GCFGH, GCFGL
CAN_CFG_TPRI	0	送信優先順位を設定します。 “0”に設定すると ID 優先、”1”に設定するとメッセージバッファ番号優先となります。	GCFGH, GCFGL
CAN_CFG_NBRP	0	ノミナルボーレートのプリスケーラ分周比を設定します。 （分周比＝CAN_CFG_NBRP＋1） 詳細はユーザーズマニュアル ハードウェア編を参照してください。	C0NCFGH, C0NCFGL
CAN_CFG_NMNL_TSEG1	63	ノミナルボーレートのタイムセグメント 1 を設定します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	C0NCFGH, C0NCFGL

CAN_CFG_NMNL_TSEG2	16	ノミナルボーレートのタイムセグメント 2 を設定します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	C0NCFGH, C0NCFGL
CAN_CFG_NMNL_SJW	16	ノミナルボーレートの再同期ジャンプ幅を設定します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	C0NCFGH, C0NCFGL
CAN_CFG_DBRP	0	データボーレートのプリスケーラ分周比を設定します。（分周比＝CAN_CFG_DBRP＋1） 詳細はユーザーズマニュアル ハードウェア編を参照してください。	C0DCFGH, C0DCFGL
CAN_CFG_DATA_TSEG1	13	データボーレートのタイムセグメント 1 を設定します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	C0DCFGH, C0DCFGL
CAN_CFG_DATA_TSEG2	6	データボーレートのタイムセグメント 2 を設定します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	C0DCFGH, C0DCFGL
CAN_CFG_DATA_SJW	6	データボーレートの再同期ジャンプ幅を設定します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	C0DCFGH, C0DCFGL
CAN_CFG_RMPLS	0	受信メッセージバッファのペイロードデータサイズを設定します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	RMNB
CAN_CFG_NRXMB	0	受信メッセージバッファ数を設定します。 “0”に設定すると受信メッセージバッファは使用できません。	RMNB
CAN_CFG_RFx_RFIGCV (x=0, 1)	0	受信 FIFO 割り込みを発生させるための受信 FIFO の深さを設定します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	RFCCk
CAN_CFG_RFx_RFIM (x=0, 1)	0	受信 FIFO 割り込みの生成条件を選択します。 “0”に設定すると CAN_CFG_RFx_RFIGCV で設定した条件に達したとき、”1”に	RFCCk

		設定すると 1 メッセージ受信完了ごとになります。	
CAN_CFG_RFx_RFDC (x=0, 1)	0	1 つの受信 FIFO に格納できるメッセージ数を設定します。 “0”に設定すると受信 FIFO は使用できません。 詳細はユーザーズマニュアルハードウェア編を参照してください。	RFCCk
CAN_CFG_RFx_RFPLS (x=0, 1)	0	受信 FIFO に格納できるペイロードの最大サイズを設定します。 詳細はユーザーズマニュアルハードウェア編を参照してください。	RFCCk
CAN_CFG_RFx_RFIE (x=0, 1)	0	この定義を”1”に設定すると受信 FIFO 割り込みが使用できます。	RFCCk
CAN_CFG_CFITT	0	送受信 FIFO が送信モードに設定されているとき、メッセージの送信間隔を設定します。 詳細はユーザーズマニュアルハードウェア編を参照してください。	CFCC
CAN_CFG_CFDC	0	1 つの送受信 FIFO に格納できるメッセージ数を設定します。 “0”に設定すると送受信 FIFO は使用できません。 詳細はユーザーズマニュアルハードウェア編を参照してください。	CFCC
CAN_CFG_CFTML	0	送受信 FIFO にリンクさせる通常の送信バッファ位置を設定します。	CFCC
CAN_CFG_CFIGCV	0	FIFO 割り込みを発生させるための送受信 FIFO の深さを設定します。 詳細はユーザーズマニュアルハードウェア編を参照してください。	CFCC
CAN_CFG_CFIM	0	FIFO 割り込みの生成条件を選択します。	CFCC

		• “0”に設定 — 受信モードの場合：CAN_CFG_CFIGCV で設定した条件に達したとき — 送信モードの場合：最後のメッセージ送信が成功したとき • ”1”に設定 — 受信モードの場合：1 メッセージ受信完了ごと — 送信モードの場合：1 メッセージ送信成功ごと	
CAN_CFG_CFITR	0	インターバル送信タイマの基準クロックの分解能を選択します。 “0”に設定すると基準クロック×1、”1”に設定すると基準クロック×10 になります。	CFCC
CAN_CFG_CFITSS	0	インターバル送信タイマの基本クロック源を選択します。 “0”に設定すると基準クロック(×1/×10 周期)、”1”に設定すると関連チャネルのビットタイムクロックになります。 CAN-FD 通信を使用する場合は”1”に設定しないでください。	CFCC
CAN_CFG_CFM	0	送受信 FIFO のモードを選択します。 “0”に設定すると受信モード、”1”に設定すると送信モードになります。	CFCC
CAN_CFG_CFPLS	0	送受信 FIFO に格納できるペイロードの最大サイズを設定します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	CFCC
CAN_CFG_CFTXIE	0	この定義を”1”に設定すると送受信 FIFO の送信割り込みが有効になります。	CFCC
CAN_CFG_CFRXIE	0	この定義を”1”に設定すると送受信 FIFO の受信割り込みが有効になります。	CFCC
CAN_CFG_THLDTE	0	送信成功後に送信履歴リストに格納する条件を選択します。 “0”に設定すると送信 FIFO、”1”に設定すると送信バッファ＋送信 FIFO になります。	THLCC
CAN_CFG_THLIM	0	割り込み生成条件を選択します。 “0”に設定すると送信履歴リスト深さの 3/4 に達したとき、”1”に	THLCC

		設定すると正常に格納するごとになります。	
CAN_CFG_THLIE	0	この定義を”1”に設定すると送信履歴割り込みが有効になります。	THLCC
CAN_CFG_THLE	0	この定義を”1”に設定すると送受信 FIFO の受信割り込みが有効になります。	THLCC
CAN_CFG_CMPOFIE	0	この定義を”1”に設定すると CAN-FD メッセージのペイロードオーバフロー割り込みを許可します。	GCTRH, GCTRL
CAN_CFG_THLEIE	0	この定義を”1”に設定すると送信履歴リストロスト割り込みを許可します。	GCTRH, GCTRL
CAN_CFG_MEIE	0	この定義を”1”に設定するとメッセージ・ロスト割り込みを許可します。	GCTRH, GCTRL
CAN_CFG_DEIE	0	この定義を”1”に設定すると受信フレームの DLC エラー割り込みを許可します。	GCTRH, GCTRL
CAN_CFG_ERRD	0	チャネルエラーフラグレジスタ(C0ERFL)のエラーフラグビット(ビット 14～8)の表示モードを設定します。 “0”に設定すると最初に発生したエラーのみ表示、”1”に設定すると累積したエラーの表示になります。	C0CTRH, C0CTRL
CAN_CFG_BOM	0	RS-CANFD lite チャネルのバスオフモードからの復帰のタイミングを設定します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	C0CTRH, C0CTRL
CAN_CFG_TDCVFIE	0	この定義を”1”に設定すると送信遅延補正バイオレーション割り込みを許可します。 クラシカル専用モード時は、この定義を”1”にしないでください。	C0CTRH, C0CTRL
CAN_CFG_TAIE	0	この定義を”1”に設定すると送信アボート割り込みを許可します。	C0CTRH, C0CTRL
CAN_CFG_ALIE	0	この定義を”1”に設定するとアービトレーション・ロスト割り込みを許可します。	C0CTRH, C0CTRL
CAN_CFG_BLIE	0	この定義を”1”に設定するとバスロック割り込みを許可します。	C0CTRH, C0CTRL
CAN_CFG_OLIE	0	この定義を”1”に設定するとオーバロード割り込みを許可します。	C0CTRH, C0CTRL

CAN_CFG_BORIE	0	この定義を”1”に設定するとバスオフ復帰割り込みを許可します。	C0CTRH, C0CTRL
CAN_CFG_BOEIE	0	この定義を”1”に設定するとバスオフ開始割り込みを許可します。	C0CTRH, C0CTRL
CAN_CFG_EPIE	0	この定義を”1”に設定するとエラー・パッシブ割り込みを許可します。	C0CTRH, C0CTRL
CAN_CFG_EWIE	0	この定義を”1”に設定するとエラーワーニング割り込みを許可します。	C0CTRH, C0CTRL
CAN_CFG_BEIE	0	この定義を”1”に設定するとバス・エラー割り込みを許可します。	C0CTRH, C0CTRL
CAN_CFG_TMIEx (x=0～3)	0	この定義を”1”に設定すると送信メッセージバッファからの送信完了割り込みを許可します。 CAN_CFG_CFE=1 かつ CAN_CFG_CFM=1 の場合、CAN_CFG_CFTML の設定値に対応するバッファ番号の設定を”1”にしないでください。	TMIEC
CAN_CFG_TSCCFG	0	タイムスタンプ値がキャプチャされる時点を選択します。 詳細はユーザーズマニュアル ハードウェア編を参照してください。	GFDCFG
CAN_CFG_RPED	0	この定義を”1”に設定するとプロトコル例外イベント検出を禁止します。	GFDCFG
CAN_CFG_RMIE_ALL	0	この定義を”1”に設定すると全受信メッセージバッファからの受信完了割り込みを許可します。 バッファごとに許可する場合は”2”を設定します。 （CAN_CFG_RMIE_VALUE を有効にします。）	RMIEC
CAN_CFG_RMIE_VALUE	0x0000	この定義のビット n を”1”に設定すると受信メッセージバッファ n からの受信完了割り込みを許可します。	RMIEC
CAN_CFG_INTRCANGRVC_USE	0	CAN グローバル受信メッセージバッファ割り込みの使用/未使用を設定します。 “0”に設定すると未使用、”1”に設定すると使用になります。	RCANGRFRMK
CAN_CFG_INTRCANGRVC_LEVEL	2	CAN グローバル受信メッセージバッファ割り込みの割り込みレベルを設定します。	RCANGRFRPR1, RCANGRFRPR0
CAN_CFG_INTRCAN0ERR_USE	0	CAN0 チャネルエラー割り込みの使用/未使用を設定します。	RCANGRVCMK

		“0”に設定すると未使用、”1”に設定すると使用になります。	
CAN_CFG_INTRCAN0ERR_LEVEL	2	CAN0 チャネルエラー割り込みの割り込みレベルを設定します。	RCANGRVCPR1, RCANGRVCPR0
CAN_CFG_INTRCAN0WUP_USE	0	CAN0 ウェイクアップ割り込みの使用/未使用を設定します。“0”に設定すると未使用、”1”に設定すると使用になります。	RCANGERRMK
CAN_CFG_INTRCAN0WUP_LEVEL	2	CAN0 ウェイクアップ割り込みの割り込みレベルを設定します。	RCANGERRPR1, RCANGERRPR0
CAN_CFG_INTRCAN0CFR_USE	0	CAN0 送受信 FIFO 受信割り込みの使用/未使用を設定します。“0”に設定すると未使用、”1”に設定すると使用になります。	RCAN0TRMMK
CAN_CFG_INTRCAN0CFR_LEVEL	2	CAN0 送受信 FIFO 受信割り込みの割り込みレベルを設定します。	RCAN0TRMPR1, RCAN0TRMPR0
CAN_CFG_INTRCAN0TRM_USE	0	CAN0 チャネル送信割り込みの使用/未使用を設定します。“0”に設定すると未使用、”1”に設定すると使用になります。	RCAN0CFRMK
CAN_CFG_INTRCAN0TRM_LEVEL	2	CAN0 チャネル送信割り込みの割り込みレベルを設定します。	RCAN0CFRPR1, RCAN0CFRPR0
CAN_CFG_INTRCANGRFR_USE	0	CAN グローバル受信 FIFO 割り込みの使用/未使用を設定します。“0”に設定すると未使用、”1”に設定すると使用になります。	RCAN0ERRMK
CAN_CFG_INTRCANGRFR_LEVEL	2	CAN グローバル受信 FIFO 割り込みの割り込みレベルを設定します。	RCAN0ERRPR1, RCAN0ERRPR0
CAN_CFG_INTRCANGERR_USE	0	CAN グローバルエラー割り込みの使用/未使用を設定します。“0”に設定すると未使用、”1”に設定すると使用になります。	RCAN0WUPMK
CAN_CFG_INTRCANGERR_LEVEL	2	CAN グローバルエラー割り込みの割り込みレベルを設定します。	RCAN0WUPPR1, RCAN0WUPPR0
CAN_CFG_CALLBACK_XXX	my_can_xxx_callback	ユーザコールバック関数名を設定します。	-
CAN_CFG_CRXD0_PU	0	CRXD0 端子のプルアップ抵抗を設定します。	PUxx
CAN_CFG_CRXD0_PITHL	0	CRXD0 端子の入力バッファ閾値を設定します。(0: Schmitt1, 1: Schmitt 3)	PITHLx

2.9 コードサイズ

以下にコードサイズを示します。

条件： ・パラメータチェック有効
・受信バッファ、受信 FIFO 使用
・送受信 FIFO は受信モードで使用
・各割り込みハンドラ有効

[CC-RL]

使用ツール

ルネサスエレクトロニクス製 CS+ for CC V8.07.00

ルネサスエレクトロニクス製 C/C++ compiler for R78 family V.1.11.00

（最適化レベル：既定の最適化を行う(オプション指定なし)）

ROM サイズ：3409byte

RAM サイズ：0byte

スタック：22byte

[IAR]

使用ツール

IAR システムズ製 IAR Embedded Workbench for Renesas RL78 4.21.3

IAR システムズ製 IAR C/C++ Compiler for Renesas RL78 4.21.3.2447

（最適化レベル：中）

ROM サイズ：3722byte

RAM サイズ：0byte

スタック：20byte

2.10 API データ構造体

本モジュールが使用するデータ・タイプ、構造体などを以下に示します。

2.10.1 データ・タイプ

本モジュールが使用するデータ・タイプを以下に示します。

定義ファイルは r_rscanfd_rl78_if.h です。

データ・タイプ	実際のデータ・タイプ	説明
int8_t	signed char	BSP が stdint.h を呼び出して定義
int16_t	signed short	同上
uint8_t	unsigned char	同上
uint16_t	unsigned short	同上
e_can_err_t e_can_txb_result_t	enum	エラー・コード、戻り値
can_rxfifo_t	unsigned char	受信 FIFO バッファ番号
can_txbuf_t	unsigned char	送信バッファ番号
can_rxbuf_t	unsigned char	受信バッファ番号
can_length_t	unsigned char	CAN のデータ長
can_storage_t	unsigned short	受信フレーム格納バッファタイプ
st_can_tx_frame_t st_can_rx_frame_t	共用体	CAN の送受信データ 詳細は 2.10.2 構造体、共用体を参照。

2.10.2 構造体、共用体

本モジュールが使用する構造体、共用体を以下に示します。

定義ファイルは r_rscanfd_rl78_if.h です。

2.10.2.1 u_can_data_t

データ・タイプ名

u_can_data_t

説明

CAN の送受信データバイト格納用共用体。

CAN フレームのデータ 64 バイト分（32 ワード）を格納する領域を定義します。

ワードアクセスする場合は DW を使用し、バイトアクセスする場合は DB を使用します。

定義

```
typedef union
{
    uint16_t    DW[32u]; /* Data Word   */
    uint8_t     DB[64u]; /* Data Byte   */
} u_can_data_t;
```

2.10.2.2 u_can_tx_head_t, u_can_rx_head_t

データ・タイプ名

u_can_tx_head_t

u_can_rx_head_t

説明

CAN の送受信 ID 格納用共用体。

送信用：u_can_tx_head_t

受信用：u_can_rx_head_t

CAN フレームの CAN ID, IDE, RTR を格納する領域を定義します。

定義

```
typedef union
{
    uint16_t       Word[CAN_TX_HEAD_WORD_NUM]; /* Word access */
    struct
    {
        /* ---- ID, THLEN, RTR and IDE (2 words) ---- */
        uint32_t    ID   :29;   /* CAN ID                        */
        uint32_t    THLEN:1;    /* THLEN  if 1 then store in THL   */
        uint32_t    RTR  :1;    /* RTR  0:Data 1:Remote(Classical) */
        uint32_t    IDE  :1;    /* IDE  0:Standard 1:Extended     */

        /* ---- Classical/FD, DLC (0.5 word) ---- */
        uint8_t     FDCTR:3;    /* FDF/BRS/ESI                    */
        uint8_t     :1;
        uint8_t     DLC  :4;    /* DLC  0-15                     */

        /* ---- Label and Time Stamp (1.5 words) ---- */
        uint8_t     IFL:2;      /* Information label              */
        uint8_t     :6;
        uint16_t    LBL;        /* TX label                      */
    } Bits;                                 /* Bit access */
} u_can_tx_head_t;

typedef union
{
    uint16_t       Word[CAN_RX_HEAD_WORD_NUM]; /* Word access */
    struct
    {
        /* ---- ID, RTR and IDE (2 words) ---- */
        uint32_t    ID :29;     /* CAN ID                    */
        uint32_t    :1;
        uint32_t    RTR:1;      /* RTR  0:Data 1:Remote(Classical) */
        uint32_t    IDE:1;      /* IDE  0:Standard 1:Extended */

        /* ---- Classical/FD, DLC (0.5 words) ---- */
        uint8_t     ESI:1;      /* ESI  0:Error Active 1:Error Passive */
        uint8_t     BRS:1;      /* BRS  0:Only Nominal 1:Use Data Baud Rate */
        uint8_t     FDF:1;      /* FDF  0:Classical 1:CAN-FD  */
        uint8_t     :1;
        uint8_t     DLC:4;      /* DLC  0-15                 */

        /* ---- Label and Time Stamp (2.5 words) ---- */
        uint8_t     IFL:2;      /* Information label          */
        uint8_t     :6;
        uint16_t    LBL;        /* RX label                  */
        uint16_t    TS;         /* Time Stamp                */
    } Bits;                                 /* Bit access */
} u_can_rx_head_t;
```

	送信時 u_can_tx_head_t	受信時 u_can_rx_head_t
ID	ID 29 ビット　標準 ID の場合、上位 18 ビットは 0 を指定してください。	ID 29 ビット　標準 ID の場合、上位 18 ビットは 0 が読み出されます。
THLEN	送信履歴の格納有無 0: 格納しない 1: 格納する	なし

	指定時はマクロ CAN_THL_XXX が使用できます。	
RTR	メッセージのデータ・フォーマット（RTR ビット） 0：データ・フレーム　1：リモート・フレーム ※リモート・フレームはクラシカル CAN フレームのみ	
IDE	メッセージの ID フォーマット（IDE ビット）　0：標準 ID　1：拡張 ID	
DLC	送信メッセージの DLC 値 FDF ビットの値で DLC に対応するデータ長が異なります。 DLC 指定の際は、以下のマクロを使用できます。 FDF=0： CAN_DLC_LEN0～CAN_DLC_LEN8 FDF=1： CAN_DLC_LEN0～CAN_DLC_LEN8, CAN_FD_DLC_LEN12～ CAN_FD_DLC_LEN64	受信メッセージの DLC 値
FDCTR	FDF/BRS/ESI ビット 000b：クラシカル CAN フレーム 100b～111b：CAN FD フレーム 100b：データビットレートと ESI を使用しない 101b：データビットレートを使用しない 110b：ESI を使用しない 111b：データビットレートと ESI を使用する 指定時はマクロ CAN_FDCTR_XXX が使用できます。	なし
ESI	なし	ESI ビット 0：エラー・アクティブノード 1：エラー・パッシブノード
BRS	なし	BRS ビット 0：データ領域のビットレートは変わらない 1：データ領域のビットレートは変わる
FDF	なし	FDF ビット 0：クラシカル CAN フレーム 1：CAN FD フレーム
IFL	送信履歴のラベル情報(2bit)	受信メッセージのラベル情報(2bit)
LBL	送信履歴のラベル情報(16bit)	受信メッセージのラベル情報(16bit)
TS	なし	受信メッセージのタイム・スタンプ値

2.10.2.3 st_can_tx_frame_t, st_can_rx_frame_t

データ・タイプ名

st_can_tx_frame_t

st_can_rx_frame_t

説明

CAN の送受信データ格納用構造体。

送信用：st_can_tx_frame_t

受信用：st_can_rx_frame_t

1 フレーム分の CAN フレーム情報を格納する領域を定義します。

定義

```
typedef struct
{
    u_can_tx_head_t   Head;
    u_can_data_t      Data;
} st_can_tx_frame_t;

typedef struct
{
    u_can_rx_head_t   Head;
    u_can_data_t      Data;
} st_can_rx_frame_t;
```

2.10.2.4 st_can_filter_t, st_can_filter_opt_t

データ・タイプ名

st_can_filter_t

st_can_filter_opt_t

説明

CAN の受信ルール格納用構造体。

受信フィルタ：st_can_filter_t

受信フィルタオプション：st_can_filter_opt_t

1 フレーム分の CAN フレーム情報を格納する領域を定義します。

定義

```
typedef struct
{
    uint32_t ID    :29;    /* CAN ID                                    */
    uint32_t :3;
    uint32_t ID_MASK:29;    /* CAN ID Mask                               */
    uint32_t :3;
    uint8_t  RTR_TYPE;      /* RTR_TYPE  0:Data 1:Remote (classical) 2:Any */
    uint8_t  IDE_TYPE;      /* IDE_TYPE  0:Standard 1:Extend 2:Any       */
} st_can_filter_t;

/* RX rule filter option */
typedef struct
{
    uint16_t LBL;          /* RX label                  */
    uint16_t IFL :2;       /* Information label           */
    uint16_t DLC :4;       /* DLC (Effective if DCE=1)     */
    uint16_t LB  :1;       /* LB  0:RX frames  1:TX frames (Effective if MME=1)
*/
    uint16_t :9;
} st_can_filter_opt_t;
```

		内容	備考
st_can_filter_t	ID	ID 29 ビット　標準 ID の場合、上位 18 ビットは 0 を指定してください。	

	ID_MASK	ID の比較ビット 1 を指定したビットが比較対象となります。 ID_MASK=0 を指定すると、ID に指定した値にかかわらず、RTR_TYPE および IDE_TYPE に該当するフレームが格納されます。	ID 全ビットを比較しない、または比較する場合は、以下のマクロを使用できます。 CAN_MATCH_NO_ID_BIT CAN_MATCH_ALL_ID_BIT
	RTR_TYPE	格納メッセージのデータ・フォーマット（RTR ビット） 0：データ・フレーム 1：リモート・フレーム 2：任意 ※リモート・フレームはクラシカル CAN フレームのみ	指定の際は、以下のマクロを使用できます。 CAN_RTR_DATA_FRAME CAN_RTR_RMT_RTR1_FRAME CAN_RTR_ANY_FRAME
	IDE_TYPE	格納メッセージの ID フォーマット（IDE ビット） 0：標準 ID 1：拡張 ID 2：任意	指定の際は、以下のマクロを使用できます。 CAN_IDE_STD_FORMAT CAN_IDE_EXT_FORMAT CAN_IDE_ANY_FORMAT
st_can_filter_opt_t	LBL	ラベル情報(16bit)	
	IFL	ラベル情報(2bit)	
	DLC	DLC 値 CAN_CFG_DCE=1 のとき、RS-CANFD が DLC チェックを行う際に参照します。 CAN_CFG_DCE=0 のときは 0 以外を設定しても影響はありません。	指定の際は、以下のマクロを使用できます。 CAN_DLC_LEN0～ CAN_FD_DLC_LEN64
	LB	ループバック CAN_CFG_MME=1 のとき、LB=1 にすると、受信フレームの代わりに送信フレームが受信ルールに従って格納されます。 CAN_CFG_MME=0 のときは 1 を設定すると送信フレーム、受信フレーム共に格納されません。	指定の際は、以下のマクロを使用できます。 CAN_AFL_LB_NOT_LOOPBACK CAN_AFL_LB_LOOPBACK

2.10.2.5 st_can_txhist_t

データ・タイプ名

st_can_txhist_t

説明

送信履歴格納用構造体。

1 エントリ分の送信履歴情報を格納する領域を定義します。

定義

```
typedef struct
{
    can_txbuf_t     txbuf_idx;  /* TX buffer index      */
    uint16_t       TS;         /* Time Stamp           */
    uint16_t       LBL;        /* TX label             */
    uint16_t       IFL :2;     /* Information label    */
    uint16_t       :14;
} st_can_txhist_t;
```

		内容	備考
st_can_txhist_t	txbuf_idx	0-3: 送信バッファ 0xFF: 送信バッファではない（送受信 FIFO）	
	TS	送信履歴のタイムスタンプ値	
	LBL	送信履歴のラベル情報(16bit)	
	IFL	送信履歴のラベル情報(2bit)	

2.10.3 マクロ

本モジュールが使用するマクロを以下に示します。

2.10.3.1 パラメータ用マクロ

本モジュールの API 関数の引数として使用するマクロを以下に示します。

定義ファイルは r_rscanfd_rl78_if.h です。

マクロ名	値	説明	使用関数
CAN_WUP_UNUSE, CAN_WUP_USE	0～1	CAN Wakeup の使用有無	R_CAN_Sleep
CAN_RXFIFO0 ～CAN_RXFIFO1	0～1	受信 FIFO バッファ番号 （例. 受信 FIFO バッファ 0 → CAN_RXFIFO0）	R_CAN_ReadRXFIFO
CAN_TXBUF0 ～CAN_TXBUF3	0～3	送信バッファ番号	R_CAN_SendByTXMB 他
CAN_TXBUF_NOT	0xFF	送信バッファではない（送受信 FIFO）	R_CAN_TxHistory
CAN_RXBUF0 ～CAN_RXBUF15	0～15	受信バッファ番号	R_CAN_ReadRXMB
CAN_DLC_LEN0～ CAN_DLC_LEN8	0～8	DLC 値（クラシカル CAN フレーム）	
CAN_DLC_LEN0～ CAN_DLC_LEN8, CAN_FD_DLC_LEN12～ CAN_FD_DLC_LEN64	0～8, 9～15	DLC 値（CAN FD フレーム） ※CAN_FD_DLC_LENx x=12,16,20,24,32,48,64	
CAN_STORE_XX	0x8000 0x0001 0x0002 0x0100	R_CAN_AddRxRule 参照	R_CAN_AddRxRule
CAN_STORE_XX_AND_YY	0x8001 0x8002 0x8100 0x0003 0x0101 0x0102	R_CAN_AddRxRule 参照	R_CAN_AddRxRule
CAN_MAX_WORD_NUM	32	CAN フレームデータの最大ワード数	

2.10.3.2 コンフィギュレーション用マクロ

2.8 コンパイル時の設定を参照してください。

2.11 戻り値

API 関数は一部の関数を除き e_can_err_t 型の戻り値を返します。e_can_err_t 型は、API 関数の宣言と共に r_rscanfd_rl78_if.h に記載されています。

以下に API 関数の戻り値一覧を示します。

型	定義	説明
e_can_err_t	CAN_SUCCESS	正常終了
	CAN_SUCCESS_WITH_LOST	正常終了（メッセージ・ロストあり）
	CAN_ERR_WAITING	完了待ち中
	CAN_ERR_INVALID_ARG	パラメータ・エラー

	CAN_ERR_INVALID_MODE	CAN モード不正
	CAN_ERR_ILLEGAL_STS	ステータス・エラー CAN モジュールで何らかのエラーが発生
	CAN_ERR_BUF_BUSY	バッファが BUSY 状態
	CAN_ERR_BUF_FULL	バッファ・フル
	CAN_ERR_BUF_EMPTY	未読メッセージなし（バッファ空）
	CAN_ERR_OVERWRITE	オーバライト発生
e_can_txb_result	CAN_TXB_TRANSMITTING	送信中、または送信要求なし　【注】
	CAN_TXB_ABORT_OVER	送信アボート完了　【注】
	CAN_TXB_END	送信完了（送信アボート要求なし）　【注】
	CAN_TXB_END_WITH_ABORT	送信完了（送信アボート要求あり）　【注】
	CAN_TXB_ARG_ERROR	パラメータ・エラー　【注】

【注】 R_CAN_GetTXMBResult 専用

3. API 関数

3.1 関数一覧

本モジュールには以下の API 関数があります。

関数名	説明
R_CAN_Create	CAN モジュールを制御する前に必要な初期化処理を実行します。
R_CAN_SetConfig	コンフィギュレーションに従い、CAN モジュールの初期設定を実行します。
R_CAN_AddRxRule	CAN モジュールの受信ルールを追加します。
R_CAN_StartComm	CAN モジュールを動作させ、チャネル 0 を通信可能なモードに遷移させます。
R_CAN_StopComm	チャネル 0 をリセットモードに遷移し、CAN モジュールを停止させます。
R_CAN_Sleep	CAN モジュールをグローバルスリープモードに遷移させます。
R_CAN_SendByTXMB	チャネル 0 の送信バッファからデータを送信します。
R_CAN_AbortTXMB	チャネル 0 の送信バッファのアボート処理を行います。
R_CAN_GetTXMBResult	チャネル 0 の送信バッファから送信結果を取得します。
R_CAN_SendByCFIFO	チャネル 0 の送受信 FIFO バッファからデータを送信します。
R_CAN_AbortCFIFO	チャネル 0 の送受信 FIFO のアボート処理を行います。
R_CAN_ReadTxHistory	チャネル 0 の送信履歴リストからエントリを読み込みます。
R_CAN_ReadRXMB	受信バッファから受信データを読み込みます。
R_CAN_ReadRXMBInHandler	受信バッファから受信データを読み込みます。（割り込みハンドラ内）
R_CAN_ReadRXFIFO	受信 FIFO バッファから受信データを読み込みます。
R_CAN_ReadCFIFO	チャネル 0 の送受信 FIFO バッファから受信データを読み込みます。
r_can_glb_rxfifo_isr	CAN 受信 FIFO 割り込みハンドラです。
r_can_glb_rxmb_isr	CAN 受信バッファ割り込みハンドラです。
r_can_glb_error_isr	CAN グローバルエラー割り込みハンドラです。
r_can_ch0_transmit_isr	CAN0 送信割り込みハンドラです。
r_can_ch0_cfifo_rx_isr	CAN0 送受信 FIFO 受信完了割り込みハンドラです。
r_can_ch0_error_isr	CAN0 エラー割り込みハンドラです。
r_can_ch0_wakeup_isr	CAN0 ウェイクアップ割り込みハンドラです。
CAN_CFG_CALLBACK_XXXX	割り込みハンドラからのコールバックです。
R_CAN_GetChStatus	チャネル 0 のステータスを取得します。
R_CAN_GetChBusErrFlag	チャネル 0 のバスエラーフラグを取得します。
R_CAN_GetTDCResult	送信遅延補正（TDC）結果を取得します。
R_CAN_GetTSCounter	タイムスタンプカウンタ値を取得します。
R_CAN_GetVersion	本ドライバのバージョンを取得します。

3.2 R_CAN_Create

CAN モジュールを制御する前に必要な初期化処理を実行します。

Format

void R_CAN_Create(void);

Parameters

なし。

Return Values

なし。

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

CAN モジュールを使用するために周辺機能の初期設定を行います。

- CAN の入力クロック供給を行う CAN0EN=1
- CAN に関連する割り込みマスク・フラグを disable にする
  RCANGRVCMK, RCAN0ERMK, RCAN0WUPMK, RCAN0CFRMK, RCAN0TRMMK, RCANGRFRMK, RCANGERRMK
- CAN に関連する割り込み要求フラグをクリアする
  RCANGRVCIF, RCAN0ERIF, RCAN0WUPIF, RCAN0CFRIF, RCAN0TRMIF, RCANGRFRIF, RCANGERRIF

Reentrant

再入不可です。

Example

R_CAN_Create();

Special Notes:

1. 割り込みの優先順位指定フラグは関数 R_CAN_SetConfig で設定します。

3.3 R_CAN_SetConfig

コンフィギュレーションに従い、CAN モジュールの初期設定を実行します。

Format

e_can_err_t R_CAN_SetConfig(void);

Parameters

なし。

Return Values

CAN_SUCCESS	正常終了
CAN_ERR_INVALID_MODE	グローバル・リセット・モードが解除されている
CAN_ERR_ILLEGAL_STS	グローバル・モードまたはチャネル・モードの遷移失敗

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

CAN モジュールの初期設定を行います。

本関数により、グローバル・モードはグローバル・リセット・モードに遷移し、使用するチャネルのチャネル・モードはチャネル・リセット・モードに遷移します。

遷移完了後、CAN モジュールのレジスタ設定を行います。

- チャネルのモード選択：CAN FD モード／FD 専用モード／クラシカル CAN 専用モード
- ボーレート設定
- 受信ルール初期化（受信ルール数を 0 にする）
- 各バッファの設定（段数、データ長、割り込み許可／禁止など）

など。

CAN レジスタの設定後、割り込みレジスタの初期設定を行います。

- CAN に関連する割り込みの優先順位指定フラグを設定する
  RCANGRVCPRx, RCAN0ERPRx, RCAN0WUPPRx, RCAN0CFRPRx, RCAN0TRMPRx, RCANGRFRPRx, RCANGERRPRx
  (x=0, 1)

Reentrant

再入不可です。

Example

```
e_can_err rtn;
rtn = R_CAN_SetConfig ();
```

Special Notes:

本関数は、関数 R_CAN_Create 呼び出し後に呼び出してください。

3.4 R_CAN_AddRxRule

CAN モジュールの受信ルールを追加します。

Format

```
e_can_err_t R_CAN_AddRxRule(const st_can_filter_t *      p_filter,
                            const can_storage_t          storage,
                            const can_rxbuf_t            rxbuf_idx,
                            const st_can_filter_opt_t * p_opt);
```

Parameters

p_filter

受信ルールを格納する構造体のアドレス。詳細は「2.10.2 構造体、共用体」を参照。

storage

受信ルールに一致したフレームの格納先（最大 2 つ）を指定。

CAN_STORE_RM	受信バッファ
CAN_STORE_RF0	受信 FIFO0
CAN_STORE_RF1	受信 FIFO1
CAN_STORE_CF	送受信 FIFO
CAN_STORE_RM_AND_RF0	受信バッファおよび受信 FIFO0
CAN_STORE_RM_AND_RF1	受信バッファおよび受信 FIFO1
CAN_STORE_RM_AND_CF	受信バッファおよび送受信 FIFO
CAN_STORE_RF0_AND_RF1	受信 FIFO0 および受信 FIFO1
CAN_STORE_RF0_AND_CF	受信 FIFO0 および送受信 FIFO
CAN_STORE_RF1_AND_CF	受信 FIFO1 および送受信 FIFO

rxbuf_idx

受信バッファに格納する場合の受信バッファ番号を指定。

storage が CAN_STORE_RM など受信バッファに格納する設定の場合に有効。

p_opt

受信ルールオプションを格納する構造体のアドレス。詳細は「2.10.2 構造体、共用体」を参照。

オプションが不要な場合は NULL を指定

Return Values

CAN_SUCCESS	正常終了
CAN_ERR_INVALID_ARG	指定パラメータに不正がある
CAN_ERR_INVALID_MODE	チャネル・リセット・モードではない
CAN_ERR_BUF_FULL	受信ルールが上限に達している

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

CAN モジュールの受信ルールを追加します。

関数 R_CAN_SetConfig 呼び出し後受信ルール数は 0 になり、本関数の正常呼び出しにより受信ルールは 1 つずつ追加されます。最大 16 のルールが設定可能です。

Reentrant

再入不可です。

Example

```
e_can_err rtn;
st_can_filter_t filter;
st_can_filter_opt_t opt;
filter.IDE_TYPE = CAN_IDE_STD_FORMAT;
filter.RTR_TYPE = CAN_RTR_ANY_FRAME;
filter.ID = 0x700u;
filter.ID_MASK = 0x700u;
opt.DLC = 0u;
opt.LB  = 0u;
opt.LBL = 0xA5A5u;
opt.IFL = 2u;
rtn = R_CAN_AddRxRule(&filter, CAN_STORE_RM, rxbuf_idx, &opt);
```

Special Notes:

本関数は、関数 R_CAN_SetConfig 呼び出し後に呼び出してください。

3.5 R_CAN_StartComm

CAN モジュールを動作させ、チャネル 0 を通信可能なモードに遷移させます。

Format

e_can_err_t R_CAN_StartComm(void);

Parameters

なし。

Return Values

CAN_SUCCESS	正常終了
CAN_ERR_INVALID_MODE	呼び出し時のグローバル・モードまたはチャネル・モードが不正
CAN_ERR_ILLEGAL_STS	グローバル・モードまたはチャネル・モードの遷移失敗
CAN_ERR_WAITING	グローバル・モードまたはチャネル・モードの遷移待ち中

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

グローバル・モードをグローバル動作モードに遷移させる処理を行い、成功したらチャネル 0 のチャネル・モードをチャネル動作モードに遷移させる処理を行います。

モード遷移が正常に行われたことを確認した後、以下の使用許可を設定します。

- 使用する受信 FIFO バッファ
- 使用する送受信 FIFO バッファ

割り込み許可対象レジスタについては、グローバル動作モードへの遷移前に、割り込みレジスタの許可設定を行います。（INTRCAN0WUP 除く）

- CAN に関連する割り込み要求フラグをクリアする
  RCANGRVCIF, RCAN0ERIF, RCAN0CFRIF, RCAN0TRMIF, RCANGRFRIF, RCANGERRIF
- CAN に関連する割り込みマスク・フラグの許可設定を行う
  RCANGRVCMK, RCAN0ERMK, RCAN0CFRMK, RCAN0TRMMK, RCANGRFRMK, RCANGERRMK

Reentrant

再入不可です。

Example

```
e_can_err rtn;
rtn = CAN_ERR_WAITING;
while (CAN_ERR_WAITING == rtn)
{
    rtn = R_CAN_StartComm();
}

if (CAN_SUCCESS == rtn)
{
    while (0x0080u != R_CAN_GetChStatus()); /* Wait until communication is ready */
}
else
{
    /* error process */
}
```

Special Notes:

1. 本関数は、関数 R_CAN_SetConfig 呼び出し後に呼び出してください。
2. 受信 FIFO バッファ、送受信 FIFO バッファの使用許可処理は、コンフィギュレーションで使用設定をしている場合にのみ行います。
3. 正常終了後、関数 R_CAN_GetChStatus で通信可能な状態になることを確認してください。

3.6 R_CAN_StopComm

チャネル 0 をリセットモードに遷移させ、CAN モジュールを停止させます。

Format

e_can_err_t R_CAN_StopComm(void);

Parameters

なし。

Return Values

CAN_SUCCESS	正常終了
CAN_ERR_WAITING	グローバル・モードまたはチャネル・モードの遷移待ち中

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

チャネル 0 のチャネル・モードをチャネル・リセット・モードに遷移させ、成功したらグローバル・モードをグローバル・リセット・モードに遷移させる処理を行います。

チャネル・リセット・モードに遷移することにより、チャネルの CAN 通信が停止します。

チャネル・リセット・モードへの遷移前にチャネル HALT モードに遷移させ、送受信の終了を待ちます。

CAN に関連する割り込みレジスタは、すべてチャネル・モード遷移処理の前に禁止設定を行います。

- CAN に関連する割り込みマスク・フラグの禁止設定を行う
  RCANGRVCMK, RCAN0ERMK, RCAN0WUPMK, RCAN0CFRMK, RCAN0TRMMK, RCANGRFRMK, RCANGERRMK
- CAN に関連する割り込み要求フラグをクリアする
  RCANGRVCIF, RCAN0ERIF, RCAN0WUPIF, RCAN0CFRIF, RCAN0TRMIF, RCANGRFRIF, RCANGERRIF

Reentrant

再入不可です。

Example

```
e_can_err rtn;
rtn = R_CAN_StopComm();
```

Special Notes:

なし。

3.7 R_CAN_Sleep

CAN モジュールをグローバルスリープモードに遷移させます。

Format

e_can_err_t R_CAN_Sleep(const uint8_t wup);

Parameters

wup

CAN ウェイクアップの使用有無（CAN_WUP_UNUSE～CAN_WUP_USE（0～1）を指定）

Return Values

CAN_SUCCESS	正常終了
CAN_ERR_INVALID_MODE	呼び出し時のグローバル・モードまたはチャネル・モードが不正
CAN_ERR_ILLEGAL_STS	グローバル・モードまたはチャネル・モードの遷移失敗

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

CAN モジュールをグローバルスリープモードに遷移させます。

このモードにより、モジュール全体のクロックを停止させ、低消費電力を実現します。

グローバルスリープモードへの遷移後、CAN モジュールへのクロックを停止します。

- CAN の入力クロック供給を停止する CAN0EN=0 (CAN_WUP_UNUSE の場合)
- CAN への X1 クロック供給を停止する CAN0MCKE=0

CAN ウェイクアップの検出には CAN0EN による CRXD0 端子へのクロック供給が必要なため、引数 wup で CAN0EN を停止するかどうかを指定します。CAN_WUP_USE が指定されている場合は、CAN0EN=0 の設定を行いません。

コンフィギュレーションで INTRCAN0WUP が許可に設定されている場合は、割り込みレジスタの許可設定を行います。

- INTRCAN0WUP の割り込み要求フラグをクリアする（RCAN0WUPIF）
- INTRCAN0WUP の割り込みマスク・フラグの許可設定を行う（RCAN0WUPMK）

Reentrant

再入不可です。

Example

```
e_can_err rtn;
rtn = R_CAN_Sleep(CAN_WUP_UNUSE);
```

Special Notes:

なし。

3.8 R_CAN_SendByTXMB

チャネル 0 の送信バッファからデータを送信します。

Format

e_can_err_t R_CAN_SendByTXMB(const can_txbuf_t txbuf_idx, const st_can_tx_frame_t * p_frame);

Parameters

txbuf_idx

送信バッファ番号（CAN_TXBUF0～CAN_TXBUF3（0～3）を指定）

p_frame

送信メッセージを格納する構造体のアドレス。詳細は「2.10.2 構造体、共用体」を参照。

Return Values

CAN_SUCCESS	正常終了
CAN_ERR_BUF_BUSY	送信バッファが BUSY 状態
CAN_ERR_INVALID_ARG	パラメータ・エラー（送信バッファ番号が 0～3 ではない）

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

指定された送信バッファにメッセージデータを格納し、送信要求を発行します。

データ格納前に、送信バッファ送信結果フラグのクリア処理を行います。

すでに送信要求がある場合や送信バッファ送信結果フラグのクリアが失敗するなど、送信バッファのステータスが 0 でない場合は、送信処理を行わずに CAN_ERR_BUF_BUSY を返却します。

Reentrant

送信バッファ番号が異なる場合、再入可能です。

Example

```
s_tx_frame.Head.Bits.IDE     = CAN_IDE_STD_FORMAT;
s_tx_frame.Head.Bits.ID      = 0x7FEuL;
s_tx_frame.Head.Bits.RTR     = CAN_RTR_DATA_FRAME;
s_tx_frame.Head.Bits.FDCTR = CAN_FDCTR_CLASSICAL;
s_tx_frame.Head.Bits.DLC     = CAN_DLC_LEN8;
s_tx_frame.Head.Bits.THLEN = CAN_THL_DISABLE;
s_tx_frame.Head.Bits.LBL   = 0u;
s_tx_frame.Head.Bits.IFL     = 0u;
for (i = 0u; i < CAN_DLC_LEN8; i++)   s_tx_frame.Data.DB[i] = 0x11u * (i + 1);
rtn = R_CAN_SendByTXMB(CAN_TXBUF0, &s_tx_frame);
```

Special Notes:

1. 本関数は、関数 R_CAN_StartComm の呼び出しが正常終了した後に呼び出してください。
2. 送受信 FIFO バッファにリンクされている送信バッファ番号は指定しないでください。

3.9 R_CAN_AbortTXMB

チャネル 0 の送信バッファのアボート要求を行います。

Format

e_can_err_t R_CAN_AbortTXMB(const can_txbuf_t txbuf_idx);

Parameters

txbuf_idx

送信バッファ番号（CAN_TXBUF0～CAN_TXBUF3（0～3）を指定）

Return Values

CAN_SUCCESS	正常終了
CAN_ERR_INVALID_ARG	パラメータ・エラー（送信バッファ番号が 0～3 ではない）

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

指定された送信バッファに対し、送信アボート要求を発行します。

Reentrant

送信バッファ番号が異なる場合、再入可能です。

Example

```
e_can_err rtn;
rtn = R_CAN_AbortTXMB(CAN_TXBUF0);
```

Special Notes:

送受信 FIFO バッファにリンクされている送信バッファ番号は指定しないでください。

3.10 R_CAN_ReadTxHistory

チャネル 0 の送信履歴リストからエントリを読み込みます。

Format

e_can_err_t R_CAN_ReadTxHistory(st_can_txhist_t *p_entry);

Parameters

p_entry

送信履歴エントリを格納する構造体のアドレス。詳細は「2.10.2 構造体、共用体」を参照。

Return Values

CAN_SUCCESS	エントリ読み出し成功
CAN_SUCCESS_WITH_LOST	エントリ読み出し成功（オーバフローあり）
CAN_ERR_BUF_EMPTY	未読エントリなし（バッファが空）
CAN_ERR_INVALID_ARG	パラメータ・エラー（p_entry が NULL）

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

格納されている送信履歴を 1 エントリ分読み込みます。

送信履歴リストのステータスを確認し、オーバフローが発生していれば、オーバフロー・フラグをクリアします。このとき、エントリの読み込みは行い、CAN_SUCCESS_WITH_LOST を返します。

バッファが空であれば CAN_ERR_BUF_EMPTY を返します。

Reentrant

再入不可です。

Example

```
e_can_err_t          rtn;
st_can_txhist_t entry;

rtn = R_CAN_ReadTxHistory(&entry);
if (rtn == CAN_SUCCESS)
{
    if (entry.txbuf_idx == CAN_TXBUF_NOT)
    {
        /* process for TX history of CFIFO */
    }
    else
    {
        /* process for TX history of TXMB */
    }
}
```

Special Notes:

1. 本関数は、RS-CANFD lite が持つ割り込み要求フラグ（THLSTS レジスタの THLIF フラグ）のクリア処理を行いません。
2. 本関数の呼び出しにより、THLSTS レジスタの THLELT フラグがクリアされます。
3. コンフィギュレーションで送信履歴を使用設定していない場合、常に CAN_ERR_BUF_EMPTY を返します。

3.11 R_CAN_GetTXMBResult

チャネル 0 の送信バッファから送信結果を取得します。

Format

```
e_can_txb_result_t R_CAN_GetTXMBResult(const can_txbuf_t txbuf_idx);
```

Parameters

txbuf_idx

送信バッファ番号（CAN_TXBUF0～CAN_TXBUF3（0～3）を指定）

Return Values

CAN_TXB_TRANSMITTING	送信中、または送信要求なし
CAN_TXB_ABORT_OVER	送信アボート完了
CAN_TXB_END	送信完了（送信アボート要求なし）
CAN_TXB_END_WITH_ABORT	送信完了（送信アボート要求あり）
CAN_TXB_ARG_ERROR	パラメータ・エラー（送信バッファ番号が 0～3 ではない）

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

指定された送信バッファのステータス・レジスタを読み出し、送信結果を返却します。

送信結果の読み出し後はステータス・レジスタのクリア処理を行い、送信結果をクリアします。

Reentrant

送信バッファ番号が異なる場合、再入可能です。

Example

```
e_can_txb_result_t rtn;
rtn = R_CAN_GetTXMBResult(CAN_TXBUF0);
```

Special Notes:

本関数の呼び出しにより、CANi 送信完了／CANi 送信アボートの割り込み要求フラグ（TMSTSm レジスタの TMTRF[1:0]フラグ）がクリアされます。

3.12 R_CAN_SendByCFIFO

チャネル 0 の送受信 FIFO バッファからデータを送信します。

Format

e_can_err_t R_CAN_SendByCFIFO(const st_can_tx_frame_t * p_frame);

Parameters

p_frame

送信メッセージを格納する構造体のアドレス。詳細は「2.10.2 構造体、共用体」を参照。

Return Values

CAN_SUCCESS	正常終了
CAN_ERR_BUF_FULL	送受信 FIFO バッファ・フル
CAN_ERR_INVALID_ARG	パラメータ・エラー（送受信 FIFO バッファが送信モードでない）

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

送受信 FIFO バッファにメッセージデータを格納します。

メッセージデータを格納する前に、送受信 FIFO バッファのステータスを確認し、送受信 FIFO バッファ・フルの場合は格納処理を行わずに CAN_ERR_BUF_FULL を返します。

Reentrant

再入不可です。

Example

```
s_tx_frame.Head.Bits.IDE     = CAN_IDE_STD_FORMAT;
s_tx_frame.Head.Bits.ID      = 0x7FEuL;
s_tx_frame.Head.Bits.RTR     = CAN_RTR_DATA_FRAME;
s_tx_frame.Head.Bits.FDCTR = CAN_FDCTR_CLASSICAL;
s_tx_frame.Head.Bits.DLC     = CAN_DLC_LEN8;
s_tx_frame.Head.Bits.THLEN = CAN_THL_DISABLE;
s_tx_frame.Head.Bits.LBL   = 0u;
s_tx_frame.Head.Bits.IFL     = 0u;
for (i = 0u; i < CAN_DLC_LEN8; i++)   s_tx_frame.Data.DB[i] = 0x11u * (i + 1);
rtn = R_CAN_SendByCFIFO(&s_tx_frame);
```

Special Notes:

1. 本関数は、関数 R_CAN_StartComm の呼び出しが正常終了した後に呼び出してください。
2. 本関数は送受信 FIFO バッファの送信完了割り込みについて、RS-CANFD lite が持つ割り込み要求フラグ（CFSTS レジスタの CFTXIF フラグ）のクリア処理を行いません。
3. コンフィギュレーションで送受信 FIFO バッファを送信モードで使用設定していない場合、本関数のコンパイル時にパラメータ・エラー返却処理のみが有効になります。

3.13 R_CAN_AbortCFIFO

チャネル 0 の送受信 FIFO のアボート処理を行います。

Format

e_can_err_t R_CAN_ AbortCFIFO(void);

Parameters

なし。

Return Values

CAN_SUCCESS	正常終了
CAN_ERR_BUF_BUSY	送受信 FIFO が空にならなかった
CAN_ERR_INVALID_ARG	パラメータ・エラー（送受信 FIFO バッファが使用されていない）

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

送受信 FIFO を一旦使用禁止にすることで、送受信 FIFO 内のメッセージをアボートします。

送受信 FIFO を使用禁止にした後、送受信 FIFO が空になったのを確認してから、送受信 FIFO を使用許可に戻します。

Reentrant

再入不可です。

Example

```
e_can_err_t    rtn;
rtn = R_CAN_AbortCFIFO();
```

Special Notes:

1. 本関数は送受信 FIFO バッファが空になるまで一定時間ウェイトを行います。もし、送受信 FIFO バッファのメッセージが送信中、または次の送信に決定している場合は、送信完了、CAN バス・エラーの検出、またはアービトレーション・ロストの後に空になります。そのため、1 フレーム分の送信が終わる前にウェイトがタイムアウトし、CAN_ERR_BUF_BUSY が返る可能性があります。
2. コンフィギュレーションで送受信 FIFO バッファを使用設定していない場合、本関数のコンパイル時にパラメータ・エラー返却処理のみが有効になります。

3.14 R_CAN_ReadRXMB

R_CAN_ReadRXMB 受信バッファから受信データを読み込みます。

R_CAN_ReadRXMBInHandler 受信バッファから受信データを読み込みます。（割り込みハンドラ内）

Format

```
e_can_err_t R_CAN_ReadRXMB(const can_rxbuf_t rxbuf_idx,
                           st_can_rx_frame_t * p_frame,
                           can_length_t * p_length);
e_can_err_t R_CAN_ReadRXMBInHandler(const can_rxbuf_t rxbuf_idx,
                                    st_can_rx_frame_t * p_frame,
                                    can_length_t * p_length);
```

Parameters

rxbuf_idx

受信バッファ番号（CAN_RXBUF0～CAN_RXBUF15（0～15））

p_frame

受信メッセージを格納する構造体のアドレス。詳細は「2.10.2 構造体、共用体」を参照。

p_length

受信 CAN データ長

Return Values

CAN_SUCCESS	メッセージ読み出し成功
CAN_ERR_BUF_EMPTY	未読メッセージなし（バッファが空）
CAN_ERR_BUF_BUSY	受信完了フラグのクリアが失敗した
CAN_ERR_OVERWRITE	読み出し途中でバッファが上書きされた
CAN_ERR_INVALID_ARG	パラメータ・エラー（受信バッファ番号が 0～[受信バッファ使用数-1]ではない）

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

指定された受信バッファに格納されている受信メッセージデータを読み込みます。

関数 R_CAN_ReadRXMBInHandler は、メッセージ読み出し前に受信完了フラグを確認し、フラグが 1 の場合はクリア処理を行います。受信完了フラグのクリア処理が失敗した場合は CAN_ERR_BUF_BUSY を返し、メッセージ読み出し途中で次の受信メッセージによってバッファが上書きされた場合は CAN_ERR_OVERWRITE を返します。フラグが 0 の場合は CAN_ERR_BUF_EMPTY を返します。

関数 R_CAN_ReadRXMBInHandler は、呼び出し前に受信完了フラグがクリアされていることを前提とし、メッセージ読み出し前に受信完了フラグの確認およびクリア処理を行いません。受信格納されている前提で読み出すため、CAN_ERR_BUF_EMPTY を返しません。

Reentrant

再入不可です。

Example

```
e_can_err_t            rtn;
st_can_rx_frame_t rx_frame;
can_length_t           length;

rtn = R_CAN_ReadRXMB(CAN_RXBUF0, &rx_frame, &length);
if (rtn == CAN_SUCCESS)
{
      /* process for receive message */
}
```

Special Notes:

コンフィギュレーションで受信バッファ使用数に 0 を設定している場合、本関数のコンパイル時にパラメータ・エラー返却処理のみが有効になります。

3.15 R_CAN_ReadRXFIFO

受信 FIFO バッファから受信データを読み込みます。

Format

```
e_can_err_t R_CAN_ReadRXFIFO(const can_rxfifo_t rxfifo_idx,
                             st_can_rx_frame_t * p_frame,
                             can_length_t * p_length);
```

Parameters

rxfifo_idx

受信 FIFO バッファ番号（CAN_RXFIFO0～CAN_RXFIFO1（0～1））

p_frame

受信メッセージを格納する構造体のアドレス。詳細は「2.10.2 構造体、共用体」を参照。

p_length

受信 CAN データ長

Return Values

CAN_SUCCESS	メッセージ読み出し成功
CAN_SUCCESS_WITH_LOST	メッセージ読み出し成功（メッセージ・ロストあり）
CAN_ERR_BUF_EMPTY	未読メッセージなし（バッファが空）
CAN_ERR_INVALID_ARG	パラメータ・エラー （受信 FIFO バッファ番号が範囲外、または指定受信 FIFO バッファが未使用設定）

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

指定された受信 FIFO バッファに格納されている受信データを 1 メッセージ分読み込みます。

受信 FIFO バッファのステータスを確認し、メッセージ・ロストが発生していれば、メッセージ・ロスト・フラグをクリアします。このとき、メッセージの読み込みは行い、CAN_SUCCESS_WITH_LOST を返します。

バッファが空であれば CAN_ERR_BUF_EMPTY を返します。

Reentrant

受信 FIFO バッファ番号が異なる場合、再入可能です。

Example

```
e_can_err_t          rtn;
st_can_rx_frame_t rx_frame;
can_length_t         length;
```

```
rtn = R_CAN_ReadRXFIFO(CAN_ RXFIFO0, &rx_frame, &length);
if (rtn == CAN_SUCCESS)
{
    /* process for receive message */
}
```

Special Notes:

1. 本関数は受信 FIFO バッファの受信完了割り込みについて、RS-CANFD lite が持つ割り込み要求フラグ（RFSTSk レジスタの RFIF フラグ）のクリア処理を行いません。
2. 本関数の呼び出しにより、RFSTSk レジスタの RFMLT フラグがクリアされます。他に FIFO メッセージ・ロストが発生している受信 FIFO バッファまたは送受信 FIFO バッファがない場合、FIFO メッセージ・ロスト割り込みの割り込み要求フラグ（GERFLL レジスタの MES フラグ）が 0 になります。
3. コンフィギュレーションで使用設定している受信 FIFO バッファがない場合、本関数のコンパイル時にパラメータ・エラー返却処理のみが有効になります。

3.16 R_CAN_ReadCFIFO

チャネル 0 の送受信 FIFO バッファから受信データを読み込みます。

Format

e_can_err_t R_CAN_ReadCFIFO(st_can_rx_frame_t * p_frame, can_length_t * p_length);

Parameters

p_frame

受信メッセージを格納する構造体のアドレス。詳細は「2.10.2 構造体、共用体」を参照。

p_length

受信 CAN データ長

Return Values

CAN_SUCCESS	メッセージ読み出し成功
CAN_SUCCESS_WITH_LOST	メッセージ読み出し成功（メッセージ・ロストあり）
CAN_ERR_BUF_EMPTY	未読メッセージなし（バッファが空）
CAN_ERR_INVALID_ARG	パラメータ・エラー（送受信 FIFO バッファが受信モードでない）

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

送受信 FIFO バッファが受信モードであるとき、格納されている受信データを 1 メッセージ分読み込みます。

送受信 FIFO バッファのステータスを確認し、メッセージ・ロストが発生していれば、メッセージ・ロスト・フラグをクリアします。このとき、メッセージの読み込みは行い、CAN_SUCCESS_WITH_LOST を返します。

バッファが空であれば CAN_ERR_BUF_EMPTY を返します。

Reentrant

再入不可です。

Example

```
e_can_err_t          rtn;
st_can_rx_frame_t rx_frame;
can_length_t         length;

rtn = R_CAN_ReadCFIFO(&rx_frame, &length);
if (rtn == CAN_SUCCESS)
{
    /* process for receive message */
}
```

Special Notes:

4. 本関数は送受信 FIFO バッファの受信完了割り込みについて、RS-CANFD lite が持つ割り込み要求フラグ（CFSTSm レジスタの CFIF フラグ）のクリア処理を行いません。
5. 本関数の呼び出しにより、CFSTS レジスタの CFMLT フラグがクリアされます。他に FIFO メッセージ・ロストが発生している受信 FIFO バッファまたは送受信 FIFO バッファがない場合、FIFO メッセージ・ロスト割り込みの割り込み要求フラグ（GERFLL レジスタの MES フラグ）が 0 になります。
6. コンフィギュレーションで送受信 FIFO バッファを受信モードで使用設定していない場合、本関数のコンパイル時にパラメータ・エラー返却処理のみが有効になります。

3.17 r_can_glb_xxxx_isr

r_can_glb_rxfifo_isr　CAN 受信 FIFO 割り込みハンドラです。

r_can_glb_rxmb_isr　CAN 受信バッファ割り込みハンドラです。

r_can_glb_error_isr　CAN グローバルエラー割り込みハンドラです。

Format

R_BSP_PRAGMA_STATIC_INTERRUPT(r_can_glb_rxfifo_isr, CAN_INTRCANGRFR_VECT);

R_BSP_PRAGMA_STATIC_INTERRUPT(r_can_glb_rxmb_isr, CAN_INTRCANGRVC_VECT);

R_BSP_PRAGMA_STATIC_INTERRUPT(r_can_glb_error_isr, CAN_INTRCANGERR_VECT);

Parameters

なし。

Return Values

なし。

Properties

r_rscanfd_rl78.c にプロトタイプ宣言されています。

Description

CAN グローバル割り込みの割り込みハンドラです。

割り込み要因となる CAN モジュールの割り込み要求フラグをクリアし、ユーザコールバック関数を呼び出します。このとき、コンフィギュレーションで割り込み許可設定されていないものはクリアの対象外とします。

例. RMIEC のコンフィギュレーション値が 1, RMND=3 → クリア処理 → RMND=2

bit0 のみクリアする。 bit1 は割り込み許可が設定されてないためクリアしない。

RL78/F2x の CAN グローバル割り込みは、割り込み要求がクリアされるまで、次の割り込みは発生しません。また、複数の割り込み要因の OR 値が 1 つの割り込みの要因に割り当てられています。そのため、本関数はすべての割り込み要因が 0 になったことが確認できるまで、繰り返しクリア処理を行います。

クリアする割り込み要求フラグと参照する割り込み許可（コンフィギュレーション値）は以下の通りです。

r_can_glb_rxfifo_isr

要求フラグ：RFSTSk レジスタの RFIF フラグ　※読み出しは RFISTS レジスタに対して行います

割り込み許可：RFCCk レジスタの RFIE フラグ

r_can_glb_rxmb_isr

要求フラグ：RMND レジスタ

割り込み許可：RMIEC レジスタ

r_can_glb_err_isr

以下のうち GCTRL レジスタで割り込み許可設定であるビット

要求フラグ：

GERFL レジスタの DEF ビット

GERFL レジスタの MSE ビットを 1 にするビット

（RFSTSk レジスタの RFMLT ビット、CFSTS レジスタの CFMLT ビット）

GERFL レジスタの CMPOF ビット

GERFL レジスタの THLES ビット

割り込み許可：

GCTR レジスタの DEIE, MEIE, CMPOFIE ビット

Reentrant

再入不可です。

Example

ユーザからの呼び出しは行いません。

Special Notes:

1. 割り込みハンドラ内では、フラグクリアまでを行います。
2. CAN 受信 FIFO、CAN 受信バッファの読み出しは、ユーザコールバック内で関数 R_CAN_ReadRXFIFO, R_CAN_ReadRXMBInHandler を呼び出す必要があります。

3.18 r_can_ch0_xxxx_isr

r_can_ch0_transmit_isr	CAN0 送信割り込みハンドラです。
r_can_ch0_cfifo_rx_isr	CAN0 送受信 FIFO 受信完了割り込みハンドラです。
r_can_ch0_error_isr	CAN0 エラー割り込みハンドラです。
r_can_ch0_wakeup_isr	CAN0 ウェイクアップ割り込みハンドラです。

Format

```
R_BSP_PRAGMA_STATIC_INTERRUPT(r_can_ch0_transmit_isr, CAN_INTRCAN0TRM_VECT);
R_BSP_PRAGMA_STATIC_INTERRUPT(r_can_ch0_cfifo_rx_isr, CAN_INTRCAN0CFR_VECT);
R_BSP_PRAGMA_STATIC_INTERRUPT(r_can_ch0_error_isr, CAN_INTRCAN0ERR_VECT);
R_BSP_PRAGMA_STATIC_INTERRUPT(r_can_ch0_wakeup_isr, CAN_INTRCAN0WUP_VECT);
```

Parameters

なし。

Return Values

なし。

Properties

r_rscanfd_rl78.c にプロトタイプ宣言されています。

Description

CAN チャネル割り込みの割り込みハンドラです。

割り込み要因となる CAN モジュールの割り込み要求フラグをクリアし、ユーザコールバック関数を呼び出します。このとき、コンフィギュレーションで割り込み許可設定されていないものはクリアの対象外とします。（INTRCAN0WUP は対象となる割り込み要求フラグはないため、クリア処理なし。）

RL78/F2x の CAN チャネル割り込みは、割り込み要求がクリアされるまで、次の割り込みは発生しません。また、INTRCAN0TRM と INTRCAN0ERR は複数の割り込み要因の OR 値が 1 つの割り込みの要因に割り当てられています。そのため、INTRCAN0TRM と INTRCAN0ERR の割り込みハンドラはすべての割り込み要因が 0 になったことが確認できるまで、繰り返しクリア処理を行います。

クリアする割り込み要求フラグと参照する割り込み許可（コンフィギュレーション値）は以下の通りです。

r_can_ch0_transmit_isr

要求フラグ：

TMSTS レジスタの TMTRF フラグ

※読み出しは TMTCSTS, TMTASTS レジスタに対して行います。

CFSTS レジスタの CFTXIF フラグ (TFIE フラグ=1 の場合)

THLSTS レジスタの THLIF フラグ

割り込み許可：

TMIEC レジスタ, C0CTRH レジスタの TAIE ビット

CFCC レジスタの CFTXIE ビット

r_can_ch0_cfifo_rx_isr

要求フラグ：CFSTS レジスタの CFRFIF フラグ

割り込み許可：参照しない　※対象となる要求フラグが CFRFIF のみのため

r_can_ch0_error_isr

要求フラグ：C0ERFL レジスタの下位 8 ビット、C0FDSTS レジスタの TDCVF フラグ

※C0FDSTS レジスタの EOCO, SOCO ビットには対応しません

割り込み許可：C0CTRL レジスタの上位 8 ビット、C0CTRH レジスタの TDCVFIE ビット

r_can_ch0_wakeup_isr

なし

Reentrant

再入不可です。

Example

ユーザからの呼び出しは行いません。

Special Notes:

なし。

3.19 CAN_CFG_CALLBACK_XXXX

割り込みハンドラからのコールバックです。user_callback 関数名は任意です。

Format

```
#define CAN_CFG_CALLBACK_GLB_RXFIFO          user_callback
#define CAN_CFG_CALLBACK_GLB_RXMB            user_callback
#define CAN_CFG_CALLBACK_GLB_ERROR           user_callback
#define CAN_CFG_CALLBACK_CH0_TRANSMIT        user_callback
#define CAN_CFG_CALLBACK_CH0_CFIFO_RX        user_callback
#define CAN_CFG_CALLBACK_CH0_ERROR           user_callback
#define CAN_CFG_CALLBACK_CH0_WAKEUP          user_callback
void CAN_CFG_CALLBACK_XXXX(uint16_t arg);
```

Parameters

arg

割り込み要因フラグ

Return Values

なし。

Properties

r_rscanfd_rl78_config.h にプロトタイプ宣言されています。

Description

CAN グローバル割り込み、CAN チャネル割り込みの割り込みハンドラから呼び出されるユーザコールバック関数です。コンフィギュレーションにより、任意の関数名を指定します。

各コールバックの呼び出し元と引き渡される割り込み要因フラグは以下の通りです。

CAN_CFG_CALLBACK_GLB_RXFIFO

r_can_glb_rxfifo_isr から呼び出し

引数 bit0: 受信 FIFO0 で受信あり、bit1: 受信 FIFO1 で受信あり、bit2-15: 0 固定

CAN_CFG_CALLBACK_GLB_RXMB

r_can_glb_rxmb_isr から呼び出し

引数 bit0: 受信バッファ 0 で受信あり、bit1: 受信バッファ 1 で受信あり、...、
bit15: 受信バッファ 15 で受信あり

CAN_CFG_CALLBACK_GLB_ERROR

r_can_glb_error_isr から呼び出し

引数 bit0: DLC エラー、bit1: メッセージ・ロスト、bit2: 送信履歴オーバフロー、
bit3: ペイロードオーバフロー
bit4-5,8: メッセージ・ロスト詳細
bit4: 受信 FIFO0 でメッセージ・ロスト、bit5: 受信 FIFO1 でメッセージ・ロスト、

bit8: 送受信 FIFO でメッセージ・ロスト

その他：0 固定

CAN_CFG_CALLBACK_CH0_TRANSMIT

r_can_ch0_transmit_isr から呼び出し

引数 bit0-3: 送信バッファ 0-3 で送信完了 (例. bit0=1 ならば送信バッファ 0 で送信完了)

bit4-7: 送信バッファ 0-3 で送信アボート完了

(例. bit4=1 ならば送信バッファ 0 でアボート完了)

bit8: 送受信 FIFO で送信割り込み、

bit12: 送信履歴割り込み、

bit9-11, 13-15: 0 固定

CAN_CFG_CALLBACK_CH0_CFIFO_RX

r_can_ch0_cfifo_rx_isr から呼び出し

引数 bit0: 送受信 FIFO で受信あり、bit1-15: 0 固定

CAN_CFG_CALLBACK_CH0_ERROR

r_can_ch0_error_isr から呼び出し

引数 bit0: バス・エラー検出、bit1: エラーワーニング検出、bit2: エラー・パッシブ検出、

bit3: バスオフ開始、bit4: バスオフ復帰、bit5: オーバロード検出、

bit6: バスロック検出、bit7: アービトレーション・ロスト検出

bit8: TDC バイオレーション検出、bit9-15: 0 固定

CAN_CFG_CALLBACK_CH0_WAKEUP

r_can_ch0_wakeup_isr から呼び出し

引数 bit0: ウェイクアップ、bit1-15: 0 固定

Reentrant

再入不可です。

Example

ユーザからの呼び出しは行いません。

Special Notes:

なし。

3.20 R_CAN_GetChStatus

チャネル 0 のステータスを取得します。

Format

uint16_t R_CAN_GetChStatus(void);

Parameters

なし。

Return Values

対象チャネルのステータス（下位 9 ビット）

bit 8	正常に受信したメッセージでレセシブの ESI を検出したとき 1
bit 7	通信可能な状態のとき 1
bit 6	受信中のとき 1、バスアイドルまたは送信中またはバスオフ状態のとき 0
bit 5	送信中またはバスオフ状態のとき 1、バスアイドルまたは受信中のとき 0
bit 4	バスオフ状態のとき 1
bit 3	エラー・パッシブ状態のとき 1
bit 2	チャネル・スリープ・モードのとき 1
bit 1	チャネル HALT モードのとき 1
bit 0	チャネル・リセット・モードのとき 1

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

対象チャネルのステータスを読み出し、返却します。

ステータスの bit8 が 1 の場合、読み出し後はステータス・レジスタのクリア処理を行い、bit8 をクリアします。

Reentrant

再入不可です。

Example

```
uint16_t sts;
sts = R_CAN_GetChStatus();
```

Special Notes:

なし。

3.21 R_CAN_GetChBusErrFlag

チャネル 0 のバスエラーフラグを取得します。

Format

uint8_t R_CAN_GetChBusErrFlag (void);

Parameters

なし。

Return Values

対象チャネルのバスエラーフラグ（下位 7 ビット）

bit 6	ACK デリミタエラー検出時 1
bit 5	ドミナントビットエラー検出時 1
bit 4	レセシブビットエラー検出時 1
bit 3	CRC エラー検出時 1
bit 2	ACK エラー検出時 1
bit 1	フォームエラー検出時 1
bit 0	スタッフエラー検出時 1

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

対象チャネルのバスエラーフラグを読み出し、返却します。

バスエラーフラグのビットが 1 の場合、読み出し後はビットをクリアします。

Reentrant

再入不可です。

Example

```
uint8_t errflag;
errflag = R_CAN_ReadChBusErrFlag();
```

Special Notes:

なし。

3.22 R_CAN_GetTDCResult

チャネル 0 の送信遅延補正（TDC）結果を取得します。

Format

uint8_t R_CAN_GetTDCResult (void);

Parameters

なし。

Return Values

送信遅延補正結果

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

対象チャネルの送信遅延補正結果を読み出し、返却します。

Reentrant

再入不可です。

Example

```
uint8_t result;
result = R_CAN_GetTDCResult();
```

Special Notes:

なし。

3.23 R_CAN_GetTSCounter

タイムスタンプカウンタ値を取得します。

Format

uint16_t R_CAN_GetTSCounter(void);

Parameters

なし。

Return Values

タイムスタンプカウンタ値

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

タイムスタンプカウンタの現在値を読み出し、返却します。

Reentrant

再入不可です。

Example

```
uint16_t value;
value = R_CAN_GetTSCounter();
```

Special Notes:

なし。

3.24 R_CAN_GetVersion

本ドライバのバージョンを取得します。

Format

uint16_t R_CAN_GetVersion(void);

Parameters

なし。

Return Values

バージョン番号（上位 8 ビット：メジャーバージョン、下位 8 ビット：マイナーバージョン）

Properties

r_rscanfd_rl78_if.h にプロトタイプ宣言されています。

Description

本ドライバのバージョンを読み出し、返却します。

Reentrant

再入不可です。

Example

```
uint16_t result;
result = R_CAN_GetVersion();
```

Special Notes:

なし。

4. 付録

4.1 動作確認環境

本モジュールの動作確認環境を以下に示します。

表 4.1 動作確認環境 (Rev.1.00)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 CS+ for CC V8.07.00 IAR システムズ製 IAR Embedded Workbench for Renesas RL78 4.21.3
C コンパイラ	ルネサスエレクトロニクス製 C/C++ compiler for R78 family V.1.11.00 IAR システムズ製 IAR C/C++ Compiler for Renesas RL78 4.21.3.2447
モジュールのリビジョン	Rev.1.00
使用ボード	RL78/F24 Target Board(型名：RTK7F124FPC01000BJ)

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2022.4.20	ー	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. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   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. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「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。
5. クロックについて
   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。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   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、将来の拡張機能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。
8. 製品間の相違について
   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、フラッシュメモリ、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、ノイズ幅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。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、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または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またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング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。
6. 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）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、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（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合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。
9. 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、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他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。
10. 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。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。
11. 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。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 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。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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直接的、間接的に支配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。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。

© 2022 Renesas Electronics Corporation. All rights reserved.

© 2022 Renesas Electronics Corporation. All rights reserved.