RX ファミリ

メモリアクセス用ドライバインタフェースモジュール

Firmware Integration Technology

要旨

本アプリケーションノートは、Firmware Integration Technology (FIT)を使用した Memory アクセス用ドライバインタフェースモジュールについて説明します。以降、本モジュールを MEMDRV FIT モジュールと称します。

なお、本モジュールはシリアル NOR/NAND Flash のコマンド制御ミドルウェア等の上位層と RSPI/QSPI/QSPIX/SCI（SPI モード）/RSCI（SPI モード）デバイスドライバ等の下位層の間のアダプターとなります。NOR/NAND Flash のコマンド制御ミドルウェアと RSPI/QSPI/QSPIX デバイスドライバは含まないため、別途入手してください。

対象コンパイラ

- Renesas Electronics C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

各コンパイラの動作確認内容については「5.1 動作確認環境」を参照してください。

関連ドキュメント

- RX ファミリ ボードサポートパッケージモジュール Firmware Integration Technology （R01AN1685）
- RX ファミリ RSPI モジュール Firmware Integration Technology（R01AN1827）
- RX ファミリ QSPI クロック同期式シングルマスタ制御モジュール Firmware Integration Technology （R01AN1940）
- RX ファミリ SCI モジュール Firmware Integration Technology（R01AN1815）
- RX ファミリ QSPIX モジュール Firmware Integration Technology（R01AN5685）
- RX ファミリ RSCI モジュール Firmware Integration Technology（R01AN5759）
- RX ファミリ DMA コントローラ DMACA 制御モジュール Firmware Integration Technology（R01AN2063）
- RX ファミリ DTC モジュール Firmware Integration Technology（R01AN1819）
- RX ファミリコンペアマッチタイマ（CMT）モジュール Firmware Integration Technology（R01AN1856）
- RX ファミリロングワード型キューバッファ（LONGQ）モジュール Firmware Integration Technology （R01AN1889）

目次

1. 概要

1.1 MEMDRV FIT モジュールとは

本モジュールは API として、プロジェクトに組み込んで使用します。本モジュールの組み込み方については、「2.12 FIT モジュールの追加方法」を参照してください。

1.2 MEMDRV FIT モジュールの概要

表 1-1 に MEMDRV FIT モジュールの概要を示します。

表 1-1　概要

項目		内容
制御デバイス数		最大 2 個
エンディアン		リトルエンディアン／ビッグエンディアンに対応
Firmware Integration Technology（FIT）		準拠
組み合せて使用する FIT モジュール	基本設定	・BSP FIT モジュール
	シリアル通信（クロック同期式シングルマスタ制御）	・RSPI FIT モジュール（Single-SPI）（注２） ・QSPI SMstr FIT モジュール（Single/Dual/Quad-SPI）（注３） ・SCI FIT モジュール（SPI mode）（注４） ・QSPIX FIT モジュール（Single-SPI）（注５） ・RSCI FIT モジュール（SPI mode）（注 6）
	I/O 設定	・GPIO FIT モジュール
	端子設定	・MPC FIT モジュール
	データ転送（注１）	・DMACA FIT モジュール ・DTC FIT モジュール
	タイマ（注１）	・CMT FIT モジュール
	データ管理	・LONGQ FIT モジュール

注１：DMAC 転送もしくは DTC 転送を使用する場合のみ
注２：RX ファミリ RSPI モジュール Firmware Integration Technology（R01AN1827）
注３：RX ファミリ QSPI クロック同期式シングルマスタ制御モジュール Firmware Integration Technology（R01AN1940）
注４：RX ファミリ SCI モジュール Firmware Integration Technology(R01AN1815)
注５：RX ファミリ QSPIX モジュール Firmware Integration Technology（R01AN5685）
注 6：RX ファミリ RSCI モジュール Firmware Integration Technology（R01AN5759）

1.3　MEMDRV FIT モジュールの使用方法

1.3.1　MEMDRV FIT モジュールを C++プロジェクト内で使用する方法

C++プロジェクトでは、MEMDRV FIT モジュールのインタフェースヘッダファイルを extern “C”の宣言に追加してください。

```
extern “C”
{
    #include “r_smc_entry.h”
    #include “r_memdrv_rx_if.h”
}
```

1.4　API の概要

表 1-2 に本モジュールに含まれる API 関数を示します。

表 1-2　API 関数一覧

関数	関数説明
R_MEMDRV_Open()	メモリドライバのオープン処理
R_MEMDRV_Close()	メモリドライバのクローズ処理
R_MEMDRV_Enable()	メモリドライバの有効化処理
R_MEMDRV_Disable()	メモリドライバの無効化処理
R_MEMDRV_EnableTxData()	データ送信の有効化処理
R_MEMDRV_DisableTxData()	データ送信の無効化処理
R_MEMDRV_EnableRxData()	データ受信の有効化処理
R_MEMDRV_DisableRxData()	データ受信の無効化処理
R_MEMDRV_Tx()	コマンド送信処理
R_MEMDRV_TxData()	データ送信処理
R_MEMDRV_Rx()	状態または ID の受信処理
R_MEMDRV_RxData()	データ受信処理
R_MEMDRV_ClearDMACFlagTx()	DMAC の送信関連割込みフラグクリア
R_MEMDRV_ClearDMACFlagRx()	DMAC の受信関連割込みフラグクリア
R_MEMDRV_1msInterval()	インターバルタイマカウント処理
R_MEMDRV_SetLogHdlAddress()	LONGQ FIT モジュールのハンドラアドレス設定処理
R_MEMDRV_Log()	LONGQ FIT モジュールを使ったエラーログ取得処理
R_MEMDRV_GetVersion()	メモリドライバのバージョン情報を取得

1.5　処理例

1.5.1　ソフトウェア構成（シリアル NOR ドライバの場合）

図 1-1 にソフトウェア構成を示します。また、表 1-3 に各レイヤの概要を示します。

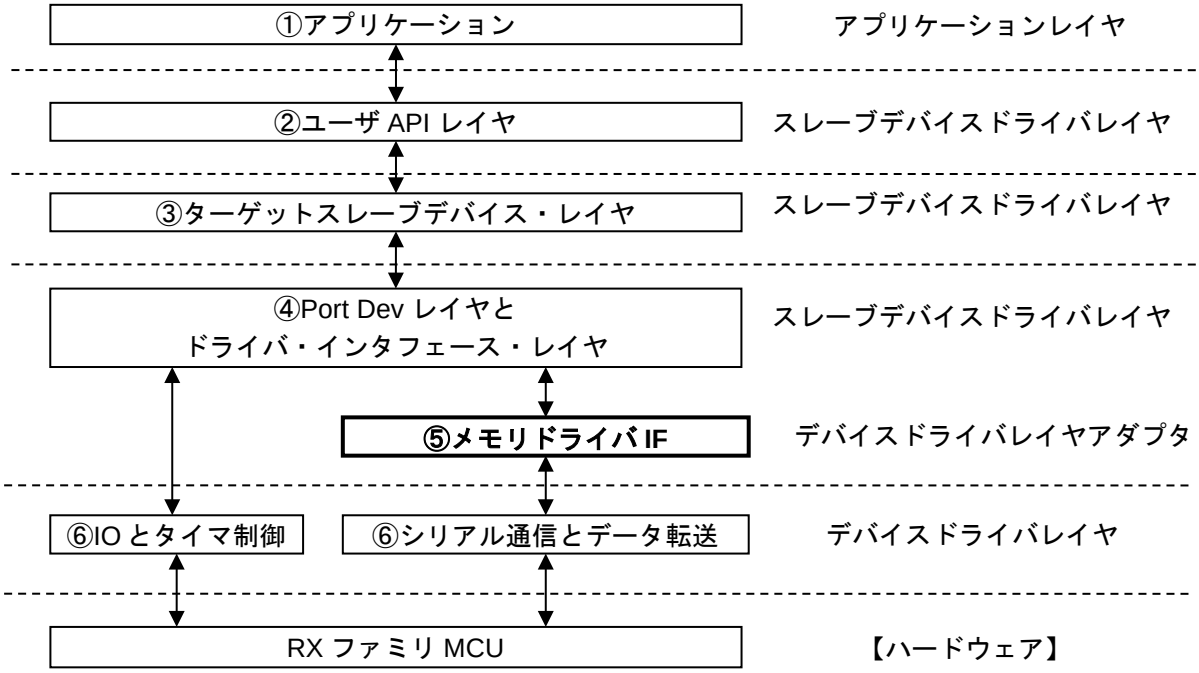

図 1-1　ソフトウェア構成

表 1-3 ソフトウェアブロック概要

ブロック名称	概要
①アプリケーション	ユーザアプリケーション
②ユーザ API レイヤ	ユーザインタフェース部分
③ターゲットスレーブデバイス・レイヤ	Serial Flash memory 制御部分
④Port Dev レイヤと ドライバ・インタフェース・レイヤ	スレーブデバイスセレクト信号をポートで制御するための制御部分 下位層のデバイスドライバとの接続部分
⑤メモリドライバ IF	**デバイスドライバレアダプター**
⑥周辺 IP 制御（シリアル通信、ポート等）	以下の制御を実行するデバイスドライバレイヤ。各 FIT モジュールを用いて制御が可能。 ・シリアル通信（クロック同期式シングルマスタ制御） ・I/O 制御 ・端子設定 ・データ転送 ・タイマ

1.5.2　API コール手順

図 1-2 に CPU 送信、受信の API コール手順を示します。

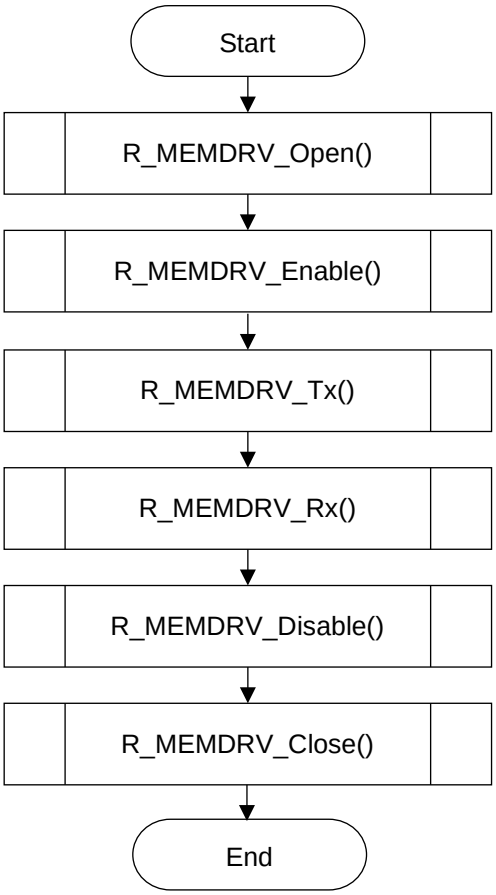

図 1-2　CPU 送信、受信の API コール手順

図 1-3 に DMAC/DTC 利用の API コール手順を示します

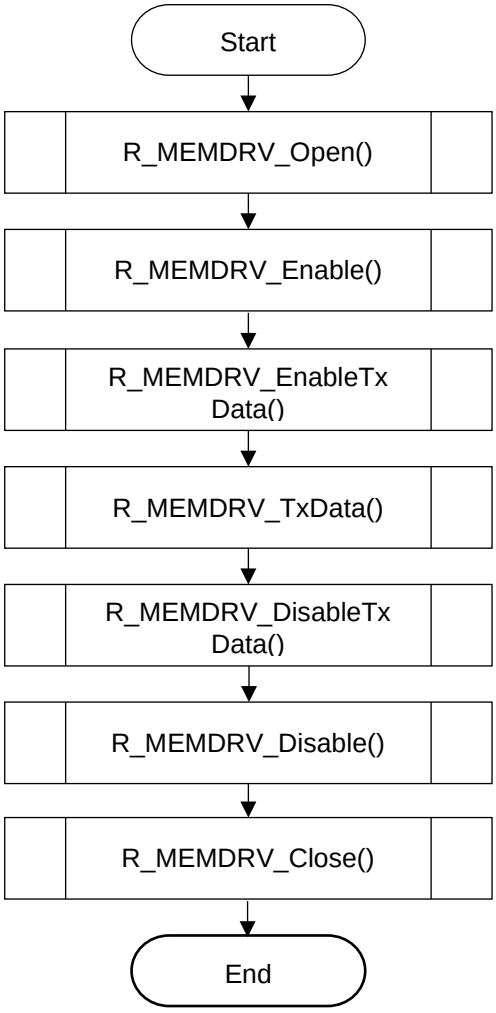

図 1-3 DMAC/DTC 利用の API コール手順

2. API 情報

本 FIT 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。

- シリアル通信（RSPI/QSPI/QSPIX/SCI の簡易 SPI モード/RSCI の簡易 SPI モード）
- I/O ポート
- DMAC/DTC データ転送（DMAC/DTC を使用する場合のみ）
- タイマ（DMAC/DTC を使用する場合のみ）

2.2 ソフトウェアの要求

このドライバは以下の FIT モジュールに依存しています。

- r_bsp Rev.5.20 以上
- r_rspi_rx（RSPI FIT モジュールを使用する場合）
- r_qspix_rx（QSPIX FIT モジュールを使用する場合）
- r_qspi_smstr_rx（クロック同期式シングルマスタ制御に QSPI SMstr FIT モジュールを使用する場合）
- r_sci_rx（SCI FIT モジュールを使用する場合、CPU 転送のみ）
- r_rsci_rx（RSCI FIT モジュールを使用する場合、CPU 転送のみ）
- r_dtc_rx（データ転送に DTC FIT モジュールを使用する場合のみ）
- r_dmaca_rx（データ転送に DMACA FIT モジュールを使用する場合のみ）
- 他タイマやソフトウェアタイマで代用できます。
- r_longq（エラーログ取得機能を有効にし、LONGQ FIT モジュールを使用する場合のみ）

2.3 サポートされているツールチェーン

本 FIT モジュールは「5.1 動作確認環境」に示すツールチェーンで動作確認を行っています。

2.4 使用する割り込みベクタ

DMAC 転送または DTC 転送を行う場合、割り込みが有効になります。割り込みの詳細については、MCU

のシリアル通信機能（クロック同期式シングルマスタ）を制御するデバイスドライバのアプリケーション

ノートをご参照ください。

2.5 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フェース定義は r_memdrv_rx_if.h に記載しています。

2.6 整数型

このドライバは 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。

2.7　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_memdrv_rx_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。

表 2-1 Configuration options(config.h)

Configuration options in r_memdrv_rx_config.h	
MEMDRV_CFG_DEVx_INCLUDED ※デバイス 0 のデフォルト値は「1」です。 デバイス 1 のデフォルト値は「0」です。 DEVx の「x」はデバイス番号を表します（x = 0 または 1）。	この定義はデバイス x に関連しています。 （1：有効、0：無効） 少なくとも 1 つのデバイスでは、このオプションを「有効」に設定する必要があります。
MEMDRV_CFG_PARAM_CHECKING_ENABLE ※デフォルト値は “BSP_CFG_PARAM_CHECKING_ENABLE”	パラメータチェック処理をコードに含めるか選択できます。 “0”を選択すると、パラメータチェック処理をコードから省略できるため、コードサイズが削減できます。 “0”の場合、パラメータチェック処理をコードから省略します。 “1”の場合、パラメータチェック処理をコードに含めます。
MEMDRV_CFG_DEVx_MODE_TRNS ※DEVx のデフォルト値は以下のとおりです。 MEMDRV_TRNS_CPU	MCU 内蔵 RAM へのデータ転送方式を定義します。 以下から 1 つ設定してください。 MEMDRV_TRNS_CPU：CPU 転送（Software 転送） MEMDRV_TRNS_DMAC：DMAC 転送 MEMDRV_TRNS_DTC：DTC 転送
MEMDRV_CFG_DEVx_MODE_DRVR ※DEVx のデフォルト値は以下のとおりです。 MEMDRV_DRVR_RX_FIT_RSPI	使用するデバイスドライバを定義します。 以下から 1 つ設定してください。 MEMDRV_DRVR_RX_FIT_RSPI： RSPI FIT モジュール MEMDRV_DRVR_RX_FIT_QSPI_SMSTR： QSPI クロック同期式シングルマスタ制御 FIT モジュール MEMDRV_DRVR_RX_FIT_SCI_SPI： SCI クロック同期制御 FIT モジュール（SPI モードで動作） MEMDRV_DRVR_RX_FIT_QSPIX_MMM： QSPIX FIT モジュール（メモリマップモード） MEMDRV_DRVR_RX_FIT_RSCI_SPI： RSCI クロック同期制御 FIT モジュール（SPI モードで動作）
MEMDRV_CFG_DEVx_MODE_DRVR_CH ※DEVx のデフォルト値は以下のとおりです。 MEMDRV_DRVR_CH0	使用するデバイスドライバのチャネル番号を定義します。 以下から 1 つ選択してください（CH0 - CH15）。 MEMDRV_DRVR_CH0 MEMDRV_DRVR_CH1 ・・・ MEMDRV_DRVR_CH15
MEMDRV_CFG_DEVx_TYPE デフォルト値は 0	デバイスタイプを定義します。 0 : NOR FLASH もしくは EEPROM. 1 : NAND FLASH.
MEMDRV_CFG_DEVx_BR ※DEVx のデフォルト値は“(uint32_t)(1000000)”	コマンド発行する際のビットレート設定です。 シリアル通信機能（RSPI or QSPI or QSPIX）のビットレートレジスタに書き込む値を設定してください。
MEMDRV_CFG_DEVx_BR_WRITE_DATA ※DEVx のデフォルト値は“(uint32_t)(1000000)”	データ出力する際のビットレート設定です。 シリアル通信機能（RSPI or QSPI or QSPIX）のビットレートレジスタに書き込む値を設定してください。

Configuration options in r_memdrv_rx_config.h	
MEMDRV_CFG_DEVx_BR_READ_DATA ※DEVx のデフォルト値は“(uint32_t)(1000000)”	データ入力する際のビットレート設定です。 シリアル通信機能（RSPI or QSPI or QSPIX）のビットレートレジスタに書き込む値を設定してください。
MEMDRV_CFG_DEVx_DMAC_CH_NO_Tx ※DEV0 のデフォルト値は“0” DEV1 のデフォルト値は“2”	DAMC FIT モジュールを使用する場合、DMAC チャネル番号を設定してください。 この DMAC チャネルは、MCU の内蔵 RAM からシリアル通信機能のデータバッファへデータを転送する時に使用します。
MEMDRV_CFG_DEVx_DMAC_CH_NO_Rx ※DEV0 のデフォルト値は“1” DEV1 のデフォルト値は“3”	DAMC FIT モジュールを使用する場合、DMAC チャネル番号を設定してください。 この DMAC チャネルは、シリアル通信機能のデータバッファから MCU の内蔵 RAM へデータを転送する時に使用します。
MEMDRV_CFG_DEVx_DMAC_INT_PRL_Tx ※DEVx のデフォルト値は“10”	DAMC FIT モジュールを使用する場合、DMAC 割り込み優先レベルを設定してください。 この DMAC チャネルは、MCU の内蔵 RAM からシリアル通信機能のデータバッファへデータを転送する時に使用します。
MEMDRV_CFG_DEVx_DMAC_INT_PRL_Rx ※DEVx のデフォルト値は“10”	DAMC FIT モジュールを使用する場合、DMAC 割り込み優先レベルを設定してください。 この DMAC チャネルは、シリアル通信機能のデータバッファから MCU の内蔵 RAM へデータを転送する時に使用します。
MEMDRV_CFG_LONGQ_ENABLE ※デフォルトは“0（無効）”	FIT モジュールの BSP 環境で使用する場合、デバッグ用のエラーログ取得処理を使用するか選択できます。（1：有効、0：無効） 無効にした場合、処理をコードから省略します。 有効にした場合、処理をコードに含めます。 使用するためには、別途 LONGQ FIT モジュールが必要です。 また、デバイスドライバの#define xxx_LONGQ_ENABLE を有効にしてください。

2.8　コードサイズ

本モジュールのコードサイズを表 2-2 に示します。

ROM (コードおよび定数) と RAM (グローバルデータ) のサイズは、ビルド時の「2.7 コンパイル時の設定」のコンフィギュレーションオプションによって決まります。掲載した値は、「2.3 サポートされているツールチェーン」の C コンパイラでコンパイルオプションがデフォルト時の参考値です。コンパイルオプションのデフォルトは最適化レベル：2、最適化のタイプ：サイズ優先、データ・エンディアン：リトルエンディアンです。コードサイズは C コンパイラのバージョンやコンパイルオプションにより異なります。

下表の値は下記条件で確認しています。

モジュールリビジョン: r_memdrv_rx rev1.05

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.04.00
(統合開発環境のデフォルト設定に”-lang = c99”オプションを追加)
GCC for Renesas RX 8.03.00.202202
(統合開発環境のデフォルト設定に” -std = gnu99”オプションを追加)
IAR C/C++ Compiler for Renesas RX version 4.20.3
(統合開発環境のデフォルト設定)

コンフィグレーションオプション: デフォルト設定

表 2-2 コードサイズ

ROM、RAM およびスタックのコードサイズ（注１）									
デバイス	分類		使用メモリ						
			Renesas Compiler		GCC		IAR Compiler		
			パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし
RX130	ROM	1 チャネル使用	2307 バイト	2132 バイト	5020 バイト	4756 バイト	3237 バイト	3025 バイト
		2 チャネル使用	2307 バイト	2132 バイト	5052 バイト	4780 バイト	3265 バイト	3053 バイト
	RAM	1 チャネル使用	25 バイト		28 バイト		18 バイト	
		2 チャネル使用	25 バイト		28 バイト		18 バイト	
	最大使用スタックサイズ		16 バイト		-		196 バイト	
RX231	ROM	1 チャネル使用	2307 バイト	2132 バイト	5044 バイト	4780 バイト	3235 バイト	2858 バイト
		2 チャネル使用	2307 バイト	2132 バイト	5076 バイト	4804 バイト	3265 バイト	2888 バイト
	RAM	1 チャネル使用	25 バイト		28 バイト		18 バイト	
		2 チャネル使用	25 バイト		28 バイト		18 バイト	
	最大使用スタックサイズ		16 バイト		-		196 バイト	
RX64M	ROM	1 チャネル使用	2307 バイト	2132 バイト	5044 バイト	4780 バイト	3235 バイト	2839 バイト
		2 チャネル使用	2307 バイト	2132 バイト	5076 バイト	4804 バイト	3261 バイト	2865 バイト
	RAM	1 チャネル使用	25 バイト		28 バイト		18 バイト	
		2 チャネル使用	25 バイト		28 バイト		18 バイト	
	最大使用スタックサイズ		16 バイト		-		228 バイト	

ROM、RAM およびスタックのコードサイズ（注 1）

デバイス	分類		使用メモリ					
			Renesas Compiler		GCC		IAR Compiler	
			パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし
RX71M	ROM	1 チャネル使用	2315 バイト	2140 バイト	5044 バイト	4780 バイト	3239 バイト	2843 バイト
		2 チャネル使用	2315 バイト	2140 バイト	5076 バイト	4804 バイト	3265 バイト	2869 バイト
	RAM	1 チャネル使用	37 バイト		40 バイト		30 バイト	
		2 チャネル使用	37 バイト		40 バイト		30 バイト	
	最大使用スタックサイズ		16 バイト		-		228 バイト	

注 1：下記条件で確認しています。
エンディアン: リトルエンディアン
クロック同期式シングルマスタ制御ソフトウェア: RSPI
データ転送モード: Software

2.9　引数

API 関数の引数である構造体を示します。この構造体は、API 関数のプロトタイプ宣言とともに r_memdrv_rx_if.h に記載されています。

2.10 戻り値

API 関数の戻り値を示します。この列挙型は API 関数のプロトタイプ宣言とともに r_memdrv_rx_if.h で記載されています。

```
typedef enum e_memdrv_err
{
    MEMDRV_BUSY          = 1,  /* Successful operation (device is busy) */
    MEMDRV_SUCCESS       = 0,  /* Successful operation */
    MEMDRV_ERR_PARAM     = -1, /* Parameter error */
    MEMDRV_ERR_HARD      = -2, /* Hardware error */
    MEMDRV_ERR_WP        = -4, /* Write-protection error */
    MEMDRV_ERR_TIMEOUT   = -6, /* Time out error */
    MEMDRV_ERR_OTHER     = -7  /* Other error */
} memdrv_err_t;
```

2.11 コールバック関数

本モジュールでは、コールバック関数がありません。

2.12 FIT モジュールの追加方法

本モジュールは、使用するプロジェクトごとに追加する必要があります。ルネサスでは、スマート・コンフィグレータを使用した(1)、(3)、(5)の追加方法を推奨しています。ただし、スマート・コンフィグレータは、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

(1) e² studio 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
e² studio のスマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: e² studio 編 (R20AN0451)」を参照してください。

(2) e² studio 上で FIT コンフィグレータを使用して FIT モジュールを追加する場合
e² studio の FIT コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加することができます。詳細は、アプリケーションノート「RX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)」を参照してください。

(3) CS+上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版スマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: CS+編 (R20AN0470)」を参照してください。

(4) CS+上で FIT モジュールを追加する場合
CS+上で、手動でユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」を参照してください。

(5) IAREW 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
スタンドアロン版スマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: IAREW 編 (R20AN0535)」を参照してください。

2.13 for 文、while 文、do while 文について

本モジュールでは、レジスタの反映待ち処理等で for 文、while 文、do while 文（ループ処理）を使用しています。これらループ処理には、「WAIT_LOOP」をキーワードとしたコメントを記述しています。そのため、ループ処理にユーザがフェイルセーフの処理を組み込む場合は、「WAIT_LOOP」で該当の処理を検索できます。

以下に記述例を示します。

```
while 文の例：
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
   /* The delay period needed is to make sure that the PLL has stabilized. */
}

for 文の例：
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
   g_protect_counters[i] = 0;
}

do while 文の例：
/* Reset completion waiting */
do
{
   reg = phy_read(ether_channel, PHY_REG_CONTROL);
   count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

2.14 制限事項

2.14.1 RAM の配置に関する制限事項

FIT では、API 関数のポインタ引数に NULL と同じ値を設定すると、パラメータチェックにより戻り値がエラーとなる場合があります。そのため、API 関数に渡すポインタ引数の値は NULL と同じ値にしないでください。

ライブラリ関数の仕様で NULL の値は 0 と定義されています。そのため、API 関数のポインタ引数に渡す変数や関数が RAM の先頭番地(0x0 番地)に配置されていると上記現象が発生します。この場合、セクションの設定変更をするか、API 関数のポインタ引数に渡す変数や関数が 0x0 番地に配置されないように RAM の先頭にダミーの変数を用意してください。

なお、CCRX プロジェクト(e^2 studio V7.5.0)の場合、変数が 0x0 番地に配置されることを防ぐために RAM の先頭番地が 0x4 になっています。GCC プロジェクト(e^2 studio V7.5.0)、IAR プロジェクト(EWRX V4.12.1)の場合は RAM の先頭番地が 0x0 になっていますので、上記対策が必要となります。

IDE のバージョンアップによりセクションのデフォルト設定が変更されることがあります。最新の IDE を使用される際は、セクション設定をご確認の上、ご対応ください。

3. API 関数

R_MEMDRV_Open()

この関数は、メモリドライバをオープンする関数です。この関数は他の API 関数を使用する前に実行される必要があります。

Format

```
memdrv_err_t R_MEMDRV_Open (
        uint8_t             devno,
        st_memdrv_info_t        *p_memdrv_info
)
```

Parameters

uint8_t　devno

デバイス番号

st_memdrv_info_t　*p_memdrv_info

メモリドライバ情報構造

uint32_t　cnt;

データ長

uint8_t　* p_data;

データ格納場所へのポインタ

uint8_t　io_mode;

QSPI 転送モード (SINGLE/DUAL/QUAD) SPI バスサイクルを閉じるか否か

bool　read_after_write;

SPI バスサイクルを閉じるか否か

Return Values

MEMDRV_SUCCESS	*/* 正常終了 */*
MEMDRV_ERR_PARAM	*/*パラメータエラー */*
MEMDRV_ERR_OTHER	*/* 他のエラー */*

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

SPI/QSPI 通信、データ転送（DTC/DMAC 選択時のみ）で使用する FIT モジュールを初期化します。

Example

```
memdrv_err_t ret_drv = MEMDRV_SUCCESS;
st_memdrv_info_t  memdrv_info;

memdrv_info.cnt = 0;
memdrv_info.p_data = NULL;
memdrv_info.io_mode = MEMDRV_MODE_SINGLE;
memdrv_info.read_after_write = true;

ret_drv = R_MEMDRV_Open(devno,&memdrv_info);
```

Special Notes:

なし

R_MEMDRV_Close()

メモリドライバをクローズする関数です。

Format

```
memdrv_err_t R_MEMDRV_Close (
        uint8_t             devno,
        st_memdrv_info_t        *p_memdrv_info
)
```

Parameters

uint8_t　devno

デバイス番号

st_memdrv_info_t　*p_memdrv_info

メモリドライバ情報構造

uint32_t　cnt;

データ長

uint8_t　* p_data;

データ格納場所へのポインタ

uint8_t　io_mode;

QSPI 転送モード (SINGLE/DUAL/QUAD)

bool　read_after_write;

SPI バスサイクルを閉じるか否か

Return Values

MEMDRV_SUCCESS	*/* 正常終了 */*
MEMDRV_ERR_PARAM	*/*パラメータエラー */*
MEMDRV_ERR_OTHER	*/* 他のエラー */*

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

SPI/QSPI 通信、データ転送（DTC/DMAC 選択時のみ）で使用する FIT モジュールをクローズします。

Example

```
memdrv_err_t ret_drv = MEMDRV_SUCCESS;
st_memdrv_info_t  memdrv_info;

memdrv_info.cnt = 0;
memdrv_info.p_data = NULL;
memdrv_info.io_mode = MEMDRV_MODE_SINGLE;
memdrv_info.read_after_write = true;

ret_drv = R_MEMDRV_Close(devno,&memdrv_info);
```

Special Notes:

なし

R_MEMDRV_Enable()

メモリドライバを有効化する関数です。

Format

```
memdrv_err_t R_MEMDRV_Enable (
        uint8_t             devno,
        st_memdrv_info_t        *p_memdrv_info
)
```

Parameters

uint8_t　devno

デバイス番号

st_memdrv_info_t　*p_memdrv_info

メモリドライバ情報構造

uint32_t　cnt;

データ長

uint8_t　* p_data;

データ格納場所へのポインタ

uint8_t　io_mode;

QSPI 転送モード (SINGLE/DUAL/QUAD)

bool　read_after_write;

SPI バスサイクルを閉じるか否か

Return Values

MEMDRV_SUCCESS	*/* 正常終了 */*
MEMDRV_ERR_PARAM	*/*パラメータエラー */*
MEMDRV_ERR_OTHER	*/* 他のエラー */*

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

メモリドライバを有効にする処理です。

Example

```
memdrv_err_t ret_drv = MEMDRV_SUCCESS;
st_memdrv_info_t  memdrv_info;

memdrv_info.cnt = 0;
memdrv_info.p_data = NULL;
memdrv_info.io_mode = MEMDRV_MODE_SINGLE;
memdrv_info.read_after_write = true;

ret_drv = R_MEMDRV_Enable(devno,&memdrv_info);
```

Special Notes:

なし

R_MEMDRV_Disable()

メモリドライバを無効化する関数です。

Format

```
memdrv_err_t R_MEMDRV_Disable (
        uint8_t             devno,
        st_memdrv_info_t        *p_memdrv_info
)
```

Parameters

uint8_t　devno

デバイス番号

st_memdrv_info_t　*p_memdrv_info

メモリドライバ情報構造

uint32_t　cnt;

データ長

uint8_t　* p_data;

データ格納場所へのポインタ

uint8_t　io_mode;

QSPI 転送モード (SINGLE/DUAL/QUAD)

bool　read_after_write;

SPI バスサイクルを閉じるか否か

Return Values

MEMDRV_SUCCESS　/* 正常終了 */

MEMDRV_ERR_PARAM　/*パラメータエラー */

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

メモリドライバの通信設定を無効にします。

Example

```
memdrv_err_t ret_drv = MEMDRV_SUCCESS;
st_memdrv_info_t  memdrv_info;

memdrv_info.cnt = 0;
memdrv_info.p_data = NULL;
memdrv_info.io_mode = MEMDRV_MODE_SINGLE;
memdrv_info.read_after_write = true;

ret_drv = R_MEMDRV_Disable(devno,&memdrv_info);
```

Special Notes:

なし

R_MEMDRV_EnableTxData()

データ送信を有効化する関数です。

Format

```
memdrv_err_t R_MEMDRV_EnableTxData (
      uint8_t      devno,
      st_memdrv_info_t   *p_memdrv_info
)
```

Parameters

uint8_t　devno

デバイス番号

st_memdrv_info_t　*p_memdrv_info

メモリドライバ情報構造

uint32_t　cnt;

データ長

uint8_t　* p_data;

データ格納場所へのポインタ

uint8_t　io_mode;

QSPI 転送モード (SINGLE/DUAL/QUAD)

bool　read_after_write;

SPI バスサイクルを閉じるか否か

Return Values

MEMDRV_SUCCESS	*/* 正常終了 */*
MEMDRV_ERR_PARAM	*/*パラメータエラー */*
MEMDRV_ERR_OTHER	*/* 他のエラー */*

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

データ送信時に使用する DMAC/DTC FIT モジュールの設定を有効にします。

Example

```
memdrv_err_t ret_drv = MEMDRV_SUCCESS;
st_memdrv_info_t  memdrv_info;

memdrv_info.cnt = 0;
memdrv_info.p_data = NULL;
memdrv_info.io_mode = MEMDRV_MODE_SINGLE;
memdrv_info.read_after_write = true;

ret_drv = R_MEMDRV_EnableTxData(devno,&memdrv_info);
```

Special Notes:

なし

R_MEMDRV_DisableTxData()

メモリドライバのデータ送信を無効化する関数です。

Format

```
memdrv_err_t R_MEMDRV_DisableTxData (
      uint8_t     devno,
      st_memdrv_info_t   *p_memdrv_info
)
```

Parameters

uint8_t　devno

デバイス番号

st_memdrv_info_t　*p_memdrv_info

メモリドライバ情報構造

uint32_t　cnt;

データ長

uint8_t　* p_data;

データ格納場所へのポインタ

uint8_t　io_mode;

QSPI 転送モード (SINGLE/DUAL/QUAD)

bool　read_after_write;

SPI バスサイクルを閉じるか否か

Return Values

MEMDRV_SUCCESS	*/* 正常終了 */*
MEMDRV_ERR_PARAM	*/*パラメータエラー */*
MEMDRV_ERR_OTHER	*/* 他のエラー */*

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

データ送信時に使用する DMAC/DTC FIT モジュールの設定を無効にします。

Example

```
memdrv_err_t ret_drv = MEMDRV_SUCCESS;
st_memdrv_info_t  memdrv_info;

memdrv_info.cnt = 0;
memdrv_info.p_data = NULL;
memdrv_info.io_mode = MEMDRV_MODE_SINGLE;
memdrv_info.read_after_write = true;

ret_drv = R_MEMDRV_DisableTxData(devno,&memdrv_info);
```

Special Notes:

なし

R_MEMDRV_EnableRxData()

データ受信を有効化する関数です。

Format

```
memdrv_err_t R_MEMDRV_EnableRxData (
      uint8_t     devno,
      st_memdrv_info_t   *p_memdrv_info
)
```

Parameters

uint8_t　devno

デバイス番号

st_memdrv_info_t　*p_memdrv_info

メモリドライバ情報構造

uint32_t　cnt;

データ長

uint8_t　* p_data;

データ格納場所へのポインタ

uint8_t　io_mode;

QSPI 転送モード (SINGLE/DUAL/QUAD)

bool　read_after_write;

SPI バスサイクルを閉じるか否か

Return Values

MEMDRV_SUCCESS　/* 正常終了 */

MEMDRV_ERR_PARAM　/*パラメータエラー */

MEMDRV_ERR_OTHER　/* 他のエラー */

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

データ受信時に使用する DMAC/DTC FIT モジュールの設定を有効にします。

Example

```
memdrv_err_t ret_drv = MEMDRV_SUCCESS;
st_memdrv_info_t  memdrv_info;

memdrv_info.cnt = 0;
memdrv_info.p_data = NULL;
memdrv_info.io_mode = MEMDRV_MODE_SINGLE;
memdrv_info.read_after_write = true;

ret_drv = R_MEMDRV_EnableRxData(devno,&memdrv_info);
```

Special Notes:

なし

R_MEMDRV_DisableRxData()

メモリドライバのデータ受信を無効化する関数です。

Format

```
memdrv_err_t R_MEMDRV_DisableRxData (
      uint8_t     devno,
      st_memdrv_info_t   *p_memdrv_info
)
```

Parameters

uint8_t　devno

デバイス番号

st_memdrv_info_t　*p_memdrv_info

メモリドライバ情報構造

uint32_t　cnt;

データ長

uint8_t　* p_data;

データ格納場所へのポインタ

uint8_t　io_mode;

QSPI 転送モード (SINGLE/DUAL/QUAD)

bool　read_after_write;

SPI バスサイクルを閉じるか否か

Return Values

MEMDRV_SUCCESS	*/* 正常終了 */*
MEMDRV_ERR_PARAM	*/*パラメータエラー */*
MEMDRV_ERR_OTHER	*/* 他のエラー */*

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

データ受信時に使用する DMAC/DTC FIT モジュールの設定を無効にします。

Example

```
memdrv_err_t ret_drv = MEMDRV_SUCCESS;
st_memdrv_info_t  memdrv_info;

memdrv_info.cnt = 0;
memdrv_info.p_data = NULL;
memdrv_info.io_mode = MEMDRV_MODE_SINGLE;
memdrv_info.read_after_write = true;

ret_drv = R_MEMDRV_DisableRxData(devno,&memdrv_info);
```

Special Notes:

なし

R_MEMDRV_Tx()

コマンド送信を処理する関数です。

Format

```
memdrv_err_t R_MEMDRV_Tx (
      uint8_t      devno,
      st_memdrv_info_t   *p_memdrv_info
)
```

Parameters

uint8_t　devno

デバイス番号

st_memdrv_info_t　*p_memdrv_info

メモリドライバ情報構造体

uint32_t　cnt;

データ長

uint8_t　* p_data;

データ格納箇所へのポインタ

uint8_t　io_mode;

QSPI 転送モード (SINGLE/DUAL/QUAD)

bool　read_after_write;

SPI バスサイクルを閉じるか否か

bool　read_in_memory_mapped;

メモリマップドモードでリードアクセスするか否か

Return Values

MEMDRV_SUCCESS	*/* 正常終了 */*
MEMDRV_ERR_PARAM	*/* パラメータエラー */*
MEMDRV_ERR_HARD	*/* ハードウエアエラー */*
MEMDRV_ERR_OTHER	*/* 他のエラー */*

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

メモリドライバ情報構造体で指定したデータを送信します。CPU 転送のみ対応します。

Example

```
memdrv_err_t ret_drv = MEMDRV_SUCCESS;
st_memdrv_info_t  memdrv_info;
uint8_t * p_data;

memdrv_info.cnt = 16;
memdrv_info.p_data = p_data;
memdrv_info.io_mode = MEMDRV_MODE_SINGLE;
memdrv_info.read_after_write = true;

ret_drv = R_MEMDRV_Tx(devno,&memdrv_info);
```

Special Notes:

なし

R_MEMDRV_TxData()

データ送信を処理する関数です。

Format

```
memdrv_err_t R_MEMDRV_TxData (
      uint8_t      devno,
      st_memdrv_info_t   *p_memdrv_info
)
```

Parameters

uint8_t　devno

デバイス番号

st_memdrv_info_t　*p_memdrv_info

メモリドライバ情報構造体

uint32_t　cnt;

データ長

uint8_t　* p_data;

データ格納箇所へのポインタ

uint8_t　io_mode;

QSPI 転送モード (SINGLE/DUAL/QUAD)

bool　read_after_write;

SPI バスサイクルを閉じるか否か

Return Values

MEMDRV_SUCCESS	*/* 正常終了 */*
MEMDRV_ERR_PARAM	*/* パラメータエラー */*
MEMDRV_ERR_HARD	*/* ハードウエアエラー */*
MEMDRV_ERR_OTHER	*/* 他のエラー */*

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

メモリドライバ情報構造体で指定したデータを送信します。CPU/DMAC/DTC 転送に対応します。

Example

```
memdrv_err_t ret_drv = MEMDRV_SUCCESS;
st_memdrv_info_t  memdrv_info;
uint8_t * p_data;

memdrv_info.cnt = 16;
memdrv_info.p_data = p_data;
memdrv_info.io_mode = MEMDRV_MODE_SINGLE;
memdrv_info.read_after_write = true;

ret_drv = R_MEMDRV_TxData(devno,&memdrv_info);
```

Special Notes:

なし

R_MEMDRV_Rx()

状態または ID の受信を処理する関数です。

Format

```
memdrv_err_t R_MEMDRV_Rx (
      uint8_t      devno,
      st_memdrv_info_t   *p_memdrv_info
)
```

Parameters

uint8_t　devno

デバイス番号

st_memdrv_info_t　*p_memdrv_info

メモリドライバ情報構造体

uint32_t　cnt;

データ長

uint8_t　* p_data;

データ格納箇所へのポインタ

uint8_t　io_mode;

QSPI 転送モード (SINGLE/DUAL/QUAD)

bool　read_after_write;

SPI バスサイクルを閉じるか否か

Return Values

MEMDRV_SUCCESS	*/* 正常終了 */*
MEMDRV_ERR_PARAM	*/* パラメータエラー */*
MEMDRV_ERR_HARD	*/* ハードウエアエラー */*
MEMDRV_ERR_OTHER	*/* 他のエラー */*

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

メモリドライバ情報構造体で指定したデータを受信します。CPU 転送のみ対応します。

Example

```
memdrv_err_t ret_drv = MEMDRV_SUCCESS;
st_memdrv_info_t  memdrv_info;
uint8_t * p_data;

memdrv_info.cnt = 16;
memdrv_info.p_data = p_data;
memdrv_info.io_mode = MEMDRV_MODE_SINGLE;
memdrv_info.read_after_write = true;

ret_drv = R_MEMDRV_Rx(devno,&memdrv_info);
```

Special Notes:

なし

R_MEMDRV_RxData()

データ受信を処理する関数です。

Format

```
memdrv_err_t R_MEMDRV_RxData (
      uint8_t      devno,
      st_memdrv_info_t   *p_memdrv_info
)
```

Parameters

uint8_t　devno

デバイス番号

st_memdrv_info_t　*p_memdrv_info

メモリドライバ情報構造体

uint32_t　cnt;

データ長

uint32_t　p_addr;

メモリマップドモード要求時に読み出し開始アドレスを指定

uint8_t　* p_data;

データ格納箇所へのポインタ

uint8_t　io_mode;

QSPI 転送モード (SINGLE/DUAL/QUAD)

uint8_t　addr_size;

アドレスサイズ

bool　read_after_write;

SPI バスサイクルを閉じるか否か

Return Values

MEMDRV_SUCCESS	*/* 正常終了 */*
MEMDRV_ERR_PARAM	*/* パラメータエラー */*
MEMDRV_ERR_HARD	*/* ハードウエアエラー */*
MEMDRV_ERR_OTHER	*/* 他のエラー */*

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

メモリドライバ情報構造体で指定したデータを受信します。CPU/DMAC/DTC 転送に対応します。

Example

```
memdrv_err_t ret_drv = MEMDRV_SUCCESS;
st_memdrv_info_t  memdrv_info;
uint8_t * p_data;

memdrv_info.cnt = 16;
memdrv_info.p_data = p_data;
memdrv_info.io_mode = MEMDRV_MODE_SINGLE;
memdrv_info.read_after_write = true;

ret_drv = R_MEMDRV_RxData(devno,&memdrv_info);
```

Special Notes:

なし

R_MEMDRV_ClearDMACFlagTx()

DMAC の転送完了時に発生する送信エンプティ割り込みフラグをクリアする関数です。

Format

```
void R_MEMDRV_ClearDMACFlagTx (
      uint8_t     channel
)
```

Parameters

uint8_t　　channel

　デバイス番号

Return Values

なし

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

送信エンプティ割り込みフラグをクリアします。DMAC の転送完了時に発生する割り込みで使用してください。

Example

```
uint8_t channel = 0;

R_MEMDRV_ClearDMACFlagTx(channel);
```

Special Notes:

なし

R_MEMDRV_ClearDMACFlagRx()

DMAC の転送完了時に発生する受信バッファフル割り込みフラグをクリアする関数です。

Format

```
void R_MEMDRV_ClearDMACFlagRx (
      uint8_t     channel
)
```

Parameters

uint8_t　　channel

デバイス番号

Return Values

なし

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

受信バッファフル割り込みフラグをクリアします。DMAC の転送完了時に発生する割り込みで使用してください。

Example

```
uint8_t channel = 0;

R_MEMDRV_ClearDMACFlagRx(channel);
```

Special Notes:

なし

R_MEMDRV_1msInterval()

インターバルタイマカウントを処理する関数です。

Format

```
void R_MEMDRV_1msInterval (
      void
)
```

Parameters

なし

Return Values

なし

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

DMAC もしくは DTC 転送完了待ち時にドライバソフトウェアの内部タイマカウンタをインクリメントします。

Example

```
R_MEMDRV_1msInterval();
```

Special Notes:

なし

R_MEMDRV_SetLogHdlAddress()

LONGQ FIT モジュールのハンドラアドレス設定を処理する関数です。

Format

```
memdrv_err_t R_MEMDRV_SetLogHdlAddress (
      uint32_t     user_long_que
)
```

Parameters

uint32_t　　user_long_que

LONGQ FIT モジュールのハンドラアドレス

Return Values

MEMDRV_SUCCESS	*/* 正常終了 */*
MEMDRV_ERR_PARAM	*/* パラメータエラー */*
MEMDRV_ERR_OTHER	*/* 他のエラー */*

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

LONGQ FIT モジュールのハンドラアドレスをメモリドライバに設定します。

Example

```
memdrv_err_t ret = MEMDRV_SUCCESS;
uint32_t long_que_hndl_address = 0;

ret = R_MEMDRV_SetLogHdlAddress(long_que_hndl_address);
```

Special Notes:

MEMDRV_CFG_LONGQ_ENABLE == 0 のときにこの関数が呼び出された場合、この関数はなにもしません。

R_MEMDRV_Log()

LONGQ FIT モジュールを使ったエラーログ取得を処理する関数です。

Format

```
uint32_t R_MEMDRV_Log (
      uint32_t flg,
      uint32_t fid,
      uint32_t line
)
```

Parameters

flg

0x00000001（固定値）

fid

0x0000003f（固定値）

line

0x00001fff（固定値）

Return Values

0	*/* 正常終了した場合 */*
1	*/* 異常終了した場合 */*

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

エラーログを取得します。

エラーログ取得を終了する場合、コールしてください。

Example

```
memdrv_err_t ret_drv = MEMDRV_SUCCESS;
st_memdrv_info_t  memdrv_info;
uint8_t * p_data;

memdrv_info.cnt = 16;
memdrv_info.p_data = p_data;
memdrv_info.io_mode = MEMDRV_MODE_SINGLE;
memdrv_info.read_after_write = true;

ret_drv = R_MEMDRV_Tx(devno,&memdrv_info);
if(MEMDRV_SUCCESS != ret_drv)
{
   R_MEMDRV_Log(0x00000001, 0x0000003f, 0x00001fff);
   R_MEMDRV_Close(devno,&memdrv_info);
}
```

Special Notes:

別途 LONGQ FIT モジュールを組み込んでください。

R_MEMDRV_GetVersion()

メモリドライバのバージョン情報を取得する関数です。

Format

```
uint32_t R_MEMDRV_GetVersion (
      void
)
```

Parameters

なし

Return Values

上位 2 バイト:	*メジャーバージョン（10 進表示）*
下位 2 バイト:	*マイナーバージョン（10 進表示）*

Properties

r_memdrv_rx_if.h にプロトタイプ宣言されています。

Description

ドライバのバージョン情報を返します。

Example

```
uint32_t version = 0;

version = R_MEMDRV_GetVersion();
```

Special Notes:

なし

4. 端子設定

MEMDRV FIT モジュールを使用するためには、マルチファンクションピンコントローラ（MPC）で周辺機能の入出力信号を端子に割り付ける（以下、端子設定と称す）必要があります。下位層のドライバ Config で設定してください。

5. 付録

5.1 動作確認環境

本 FIT モジュールの動作確認環境を以下に示します。

表 5-1　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 e^2 studio V7.3.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V.3.01.00
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Ver.1.00
使用ボード	Renesas Starter Kit+ for RX65N（型名：RTK500565Nxxxxxxxx）
	Renesas Starter Kit for RX72T (型名.: RTK5572Txxxxxxxxxx)

表 5-2　動作確認環境(Rev.1.01)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.3.0 IAR Embedded Workbench for Renesas RX 4.10.1
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.08.04.201803 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.10.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Ver.1.01
使用ボード	Renesas Starter Kit+ for RX65N（型名：RTK500565Nxxxxxx）

表 5-3　動作確認環境(Rev.1.02)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.7.0 IAR Embedded Workbench for Renesas RX 4.12.1
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.08.04.201902 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.12.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Ver.1.02
使用ボード	Renesas Starter Kit+ for RX72M（型名：RTK5572Mxxxxxxxxxx）

表 5-4　動作確認環境(Rev.1.03)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio 2020-07 IAR Embedded Workbench for Renesas RX 4.14.1
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.02.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 8.03.00.202002 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.14.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Ver.1.03
使用ボード	Renesas Starter Kit+ for RX72N（型名：RTK5572Nxxxxxxxxxx）

表 5-5　動作確認環境(Rev.1.04)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio 2021-07 IAR Embedded Workbench for Renesas RX 4.20.1
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.03.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 8.03.00.202102 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.20.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Ver.1.04
使用ボード	Renesas Starter Kit+ for RX671（型名：RTK55671xxxxxxxxxx）

表 5-6　動作確認環境(Rev.1.05)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio 2023-01 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.04.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 8.03.00.202202 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.20.03 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Ver.1.05
使用ボード	Evaluation+ for RX671（型名：RTK5EK671xxxxxxxxxx）

5.2　トラブルシューティング

(1) Q : 本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「Could not open source file "platform.h"」エラーが発生します。

A : FIT モジュールがプロジェクトに正しく追加されていない可能性があります。プロジェクトへの追加方法をご確認ください。

- CS+を使用している場合
  アプリケーションノート RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」
- e^2 studio を使用している場合
  アプリケーションノート RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」

また、本 FIT モジュールを使用する場合、ボードサポートパッケージ FIT モジュール(BSP モジュール)もプロジェクトに追加する必要があります。BSP モジュールの追加方法は、アプリケーションノート「ボードサポートパッケージモジュール(R01AN1685)」を参照してください。

(2) Q : 本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「This MCU is not supported by the current r_memdrv_rx module.」エラーが発生します。

A : 追加した FIT モジュールがユーザプロジェクトのターゲットデバイスに対応していない可能性があります。追加した FIT モジュールの対象デバイスを確認してください。

(3) Q : GCC プロジェクト(e^2 studio V7.5.0)、IAR プロジェクト(EWRX V4.12.1)で、R_MEMDRV_Open に失敗します。

A.2.14.1 章の制限事項に該当している可能性があります。2.14.1 章を参照してください。

6. 参考ドキュメント

ユーザーズマニュアル：ハードウェア
（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート／テクニカルニュース
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境
RX ファミリ CC-RX コンパイラ ユーザーズマニュアル（R20UT3248）
（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデートの対応について

本モジュールは以下のテクニカルアップデートの内容を反映しています。

- テクニカルアップデートはありません。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2019.02.20	-	初版発行
1.01	2019.05.20	-	以下のコンパイラに対応 ・GCC for Renesas RX ・IAR C/C++ Compiler for Renesas RX
		1	「対象コンパイラ」を追加
		2	「関連ドキュメント」に、R01AN1723、R01AN1826 と R20AN0451 を削除
		9	「2.2 ソフトウェアの要求」 依存する r_bsp モジュールのリビジョンを追加
		10	表 2 1 Configuration options(config.h) MEMDRV_CFG_DEVx_MODE_DRVR　誤記を修正
		12-13	「2.8 コードサイズ」を更新
		16	「2.13　for文、while文、do while文について」に、「WAIT_LOOP」を記述している対象デバイスを削除
		49	「5.1 動作確認環境」に、表 5-2 を追加
1.02	2019.11.22	-	ソフトウェア不具合対応
		1	「対象デバイス」を削除
		1	「関連ドキュメント」に、R01AN1833 を削除
		3	「1.2 MEMDRV FIT モジュールの概要」に、記載を修正
		5	「1.4.1 ソフトウェア構成（シリアル NOR ドライバの場合）」を修正
		9	「2.7 コンパイル時の設定」に、MEMDRV_CFG_DEVx_TYPE を追加
		11	「2.8 コードサイズ」を更新
		13	「2.10 戻り値」の記載を更新
		16	「2.14 制限事項」を追加
		17-40	「3. API 関数」に、各 API の Reentrant を削除 「3.16 R_MEMDRV_SetLogHdlAddress()」の「Special Notes」を修正
		42	表 5-3 動作確認環境 (Rev.1.02)を追加
		43	「5.2 トラブルシューティング」に、(3)を追加
		ﾌﾟﾛｸﾞﾗﾑ	ソフトウェア不具合のため、Memdrv FIT モジュールを改修 ■内容 シリアルフラッシュ FIT は、書き込みバイト数/読み出しバイト数の上限を、4,294,967,295 バイト（0xffffffff）としているが、1024 バイトを書き込もうとしたら、データ転送が行われていません。 ■対策 Memdrv FIT モジュール Rev1.02 以降をご使用ください。
1.03	2020.09.10	-	DMAC/DTC 経由のコールバック処理を修正
		4	表 1-2 の誤記を修正
		11	「2.8 コードサイズ」を更新
		14	「2.12 FIT モジュールの追加方法」を更新
		43	表 5-4 動作確認環境 (Rev.1.03)を追加

Rev.	発行日	改訂内容	
		ページ	ポイント
1.03	2020.09.10	ﾌﾟﾛｸﾞﾗﾑ	r_rspi_rx FIT モジュールが更新されたため、以下の処理を修正 R_MEMDRV_ClearDMACFlagTx R_MEMDRV_ClearDMACFlagRx r_memdrv_rspi_callback
		ﾌﾟﾛｸﾞﾗﾑ	ソフトウェア不具合のため、Memdrv FIT モジュールを改修 ■内容 IAR かつビッグエンディアンで、使用するデバイスドライバを RSPI にして、Software 転送でデータを転送するように設定します。R_MEMDRV_TxData()と R_MEMDRV_RxData()で 4 バイト以上のデータを転送すると、指定した転送サイズより多くのデータが転送されてしまいます。 ■対策 Memdrv FIT モジュール Rev1.03 以降をご使用ください。
1.04	2021.10.30	1	「要旨」に、QSPIX を追加
		1	「関連ドキュメント」に、R01AN5685 を追加
		3	「1.2 MEMDRV FIT モジュールの概要」に、記載を修正
		4	「1.3 MEMDRV FIT モジュールの使用方法」を追加
		8	「2.1 ハードウェアの要求」に、QSPIX を追加
		8	「2.2 ソフトウェアの要求」に、r_qspix_rx（QSPIX FIT モジュールを使用する場合）を追加
		9	「2.7 コンパイル時の設定」に、 「MEMDRV_DRVR_RX_FIT_QSPIX_IAM」と「シリアル通信機能（RSPI or QSPI or QSPIX）」を追加
		11-12	「2.8 コードサイズ」を更新
		17-46	「3. API 関数」の説明を追加
		50	表 5-5 動作確認環境 (Rev.1.04)を追加
		ﾌﾟﾛｸﾞﾗﾑ	R_QSPIX_RX FIT モジュールが追加されているので、ファイル"r_memdrv_rx.c"が修正しました。 そして、ファイル "r_memdrv_qspix.c" が追加しました。
1.05	2023.03.16	1	「要旨」に、RSCI を追加
		1	「関連ドキュメント」に、R01AN5759 を追加
		4	「1.2 MEMDRV FIT モジュールの概要」に、記載を修正
		9	「2.1 ハードウェアの要求」に。
		9	「2.2 ソフトウェアの要求」に。
		11	「2.7 コンパイル時の設定」に、 「MEMDRV_DRVR_RX_FIT_QSPIX_MMM」と 「MEMDRV_DRVR_RX_FIT_RSCI_SPI」を追加 通信関数に RSCI と QSPIX のメモリマップドモードを追加。
		13	2.8 FIT モジュールのバージョンおよびコンパイラのバージョンを更新。
		34, 40	「3. API 関数」の説明を追加
		51	表 5-6 動作確認環境 (Rev.1.05)を追加
		ﾌﾟﾛｸﾞﾗﾑ	r_rsci_rx FIT モジュールが追加されているので、ファイル"r_memdrv_rx.c"が修正しました。 そして、ファイル "r_memdrv_rsci.c" が追加しました。 RSCI および QSPIX メモリマップモードのサポートを追加。

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。