RX ファミリ

DTC 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 DTC モジュールについて説明します。本モジュールは DTC ソフトウェアモジュールを使用して、データ転送処理の制御を行います。以降、本モジュールを DTC FIT モジュールと称します。

対象デバイス

- RX110 グループ、RX111 グループ、RX113 グループ、RX130 グループ、RX13T グループ、RX140 グループ
- RX230 グループ、RX231 グループ、RX23T グループ、RX24T グループ、RX24U グループ, RX26T グループ
- RX23W グループ
- RX23E-A グループ
- RX23E-B グループ
- RX260 グループ, RX261 グループ
- RX64M グループ、RX65N グループ、RX651 グループ、RX66T グループ、RX66N グループ、RX660 グループ、RX671 グループ
- RX71M グループ
- RX72T グループ
- RX72M グループ、RX72N グループ
- RX671 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。

ターゲットコンパイラ

- ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX （RX13T はサポートされていません。）

各コンパイラの動作確認環境に関する詳細な内容は、セクション「6.1 動作確認環境」を参照してください。

目次

1. 概要 4
1.1 DTC FIT モジュールとは 4
1.2 DTC FIT モジュールの概要 4
1.3 DTC FIT モジュールを使用する 6
1.3.1 DTC FIT モジュールを C++プロジェクト内で使用する 6
1.4 API の概要 6
1.5 DTC IP バージョン 7

2. API 情報 8
2.1 ハードウェアの要求 8
2.2 ソフトウェアの要求 8
2.3 制限事項 8
2.3.1 RAM の配置に関する制限事項 8
2.4 サポートされているツールチェーン 8
2.5 使用する割り込みベクタ 9
2.6 ヘッダファイル 9
2.7 整数型 9
2.8 コンパイル時の設定 10
2.9 コードサイズ 11
2.10 引数 17
2.10.1 r_dtc_rx_if.h 17
2.10.2 r_dtc_rx_target_if.h 18
2.11 戻り値 19
2.12 コールバック関数 19
2.13 FIT モジュールの追加方法 19
2.14 for 文、while 文、do while 文について 20

3. API 関数 21
R_DTC_Open() 21
R_DTC_Close() 22
R_DTC_Create() 24
R_DTC_CreateSeq() 33
R_DTC_Control() 40
R_DTC_GetVersion() 47

4. 端子設定 48

5. デモプロジェクト 49
5.1 dtc_demo_rskrx231, dtc_demo_rskrx231_gcc 49
5.2 dtc_demo_rskrx65n_2m, dtc_demo_rskrx65n_2m_gcc 49
5.3 dtc_demo_rskrx130, dtc_demo_rskrx130_gcc 49
5.4 dtc_demo_rskrx72m, dtc_demo_rskrx72m_gcc 49
5.5 dtc_demo_rskrx671, dtc_demo_rskrx671_gcc 49
5.6 ワークスペースにデモを追加する 49
5.7 デモのダウンロード方法 50

6. 付録 51
6.1 動作確認環境 51
6.2 トラブルシューティング 64

7. 参考ドキュメント 65

テクニカルアップデートの対応について 65

改訂記録 66

1. 概要

1.1 DTC 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、「2.13 FIT モジュールの追加方法」を参照してください。

1.2 DTC FIT モジュールの概要

DTC FIT モジュールは、以下の 3 つの転送モードをサポートしています。

- ノーマル転送モード
- リピート転送モード
- ブロック転送モード

各モードでチェーン転送機能、および、シーケンス転送の許可／禁止を設定できます。詳細はユーザーズマニュアル ハードウェア編の「データトランスファコントローラ」章をご覧ください。

DTC は割り込み要因の割り込み要求によって起動されます。ユーザは各起動要因に対する 1 個の転送情報、またはチェーン転送機能を使用する場合、連続する複数の転送情報を作成する必要があります。

転送情報には転送元と転送先の先頭アドレスと、DTC がデータを転送元から転送先にどのように転送するかを指定する設定情報が含まれます。DTC が起動すると、該当の割り込みに対応する転送情報を読み込み、その情報に従ってデータ転送を開始します。

DTC は指定された割り込み要因に対応する転送情報の先頭アドレスを DTC ベクタテーブルから読み込みます。ベクタテーブルは 4 バイトアドレスの配列で、各割り込み要因に対応する転送情報（n）の先頭アドレスが、ベクタ番号（n）に従ってテーブルのアドレス（4×n）の位置に配列要素として格納されています。

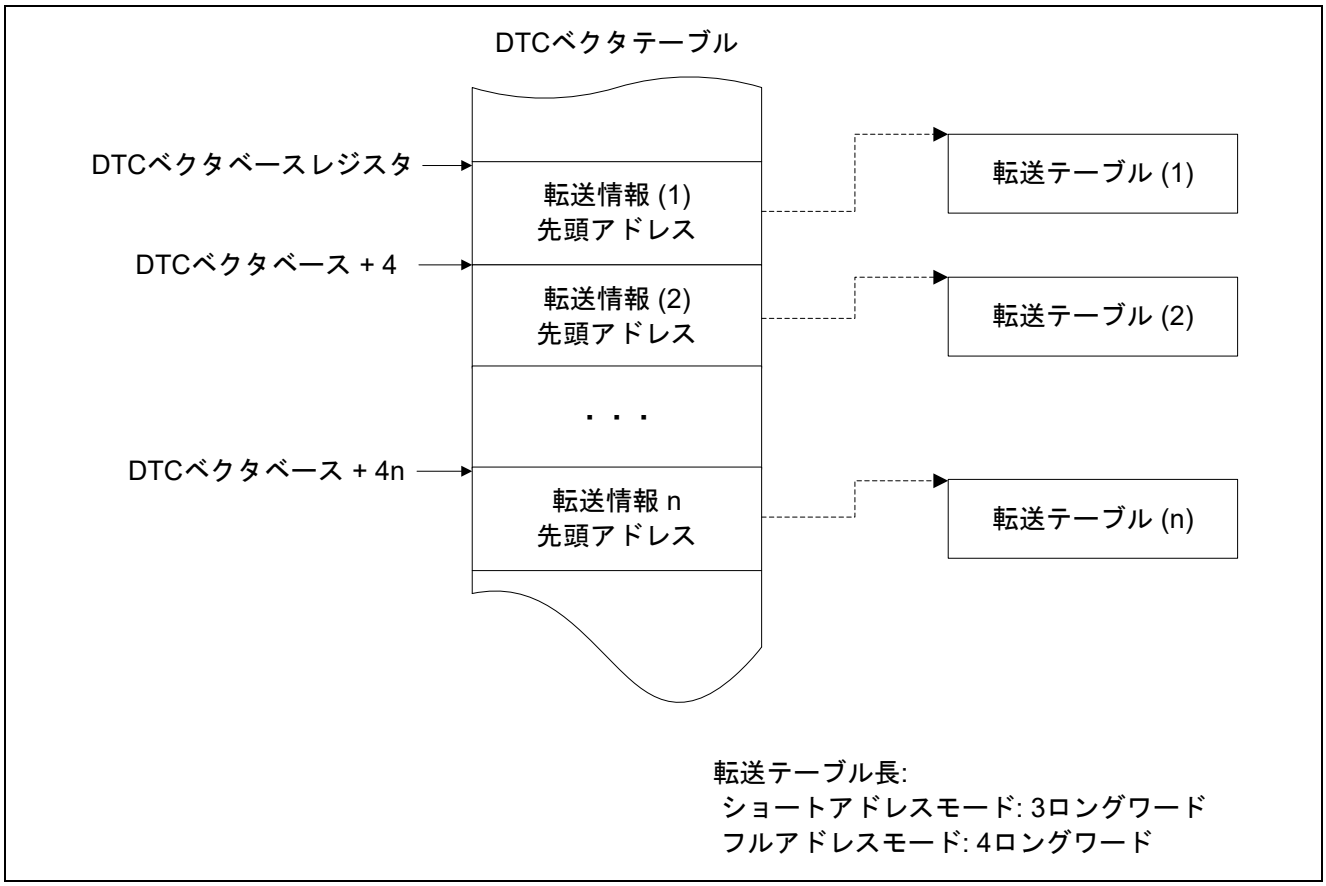

図 1.1　DTC ベクタと転送情報

DTC を使用する前に、RAM 領域に DTC ベクタテーブル用にメモリ領域を割り当てる必要があります。メモリはバイト単位で割り当てられ、メモリサイズは DTC で対応可能な割り込み要因の最大ベクタ番号に依存します。メモリサイズは、targets フォルダの各 MCU フォルダにある r_dtc_rx_target.h ファイルで定義される DTC_VECTOR_TABLE_SIZE_BYTES の値によって指定されます。メモリサイズのデフォルト値は割り込みベクタテーブルで定義できるすべての起動要因に対応可能な値となります。例えば、RX111 の場合、デフォルト値は 0x3E4 (0x3E4 = 249×4)で、RX64M の場合は 0x400 (0x400 = 256×4)となります。DTC ベクタテーブルの先頭アドレスは 1K バイト単位であることが必要です。また、ベクタテーブルはコンパイル時にリンカを使って配置することもできます。

DTC モジュールはショートアドレスモードとフルアドレスモードの 2 つのアドレスモードで動作することができます。ショートアドレスモードでは、1 つの転送情報のサイズは 3 ロングワード（12 バイト）で、DTC は 0x00000000～0x007FFFFF と 0xFF800000～0xFFFFFFFF の 16M バイトのメモリ空間にアクセスできます。フルアドレスモードでは、転送情報のサイズは 4 ロングワード（16 バイト）で、DTC は 0x00000000～0xFFFFFFFF の 4G バイトのメモリ空間にアクセスできます。

デフォルトでは、DTC は起動割り込みが発生するたびに転送情報をリードします。 1 つの起動要因から 2 回もしくは連続して何回もの起動が発生する場合、前の起動動作で転送情報が既に DTC 内に存在するため、2 回目以降のリードをスキップして DTC の転送効率を向上させることができます。転送情報リードスキップを許可するには、初期化時に R_DTC_Open()で設定するか、または R_DTC_Control()で DTC_CMD_ENABLE_READ_SKIP コマンドを使用します。

DTC モジュールを初期化するには、R_DTC_Open()を呼び出します。この関数は、DTC にクロックの供給を開始し、DTC ベクタテーブルの先頭アドレスを DTC ベクタベースレジスタ（DTCVBR）に書き込みます。シーケンス転送を使用する場合は、DTC インデックステーブの先頭アドレスを DTC インデックステーブルベースレジスタ（DTCIBR）に書き込みます。また、r_dtc_rx_config.h のユーザ設定に従って転送情報リードスキップ、DTC アドレスモード、および DTCER レジスタの設定を初期化します。

R_DTC_Create()関数にユーザが選択した設定内容を渡して、各割り込み要因に対応する転送情報を作成します。転送情報には転送元と転送先の先頭アドレス、および DTC がどのようにデータを転送元から転送先に転送するかを指示する情報が含まれます。R_DTC_Create()では、転送情報の先頭アドレスを DTC ベクタテーブルの指定されたベクタ番号の位置に格納します。

R_DTC_CreateSeq()関数はシーケンス転送を行うための転送情報を作成し、転送情報の先頭アドレスを DTC インデックステーブルの指定されたシーケンス番号の位置に格納します。

R_DTC_Control()を使って、DTC 起動因となる割り込みの選択と解除、DTC に供給するクロックの起動と停止、転送情報リードスキップ機能の許可／禁止、処理中のチェーン転送の中断、シーケンス転送の許可／禁止／中断を行います。

起動要因により割り込みが発生すると、DTC が起動されます。DTC は起動割り込みのベクタ番号に対応する転送情報を読み込んで設定を行い、データを転送します。R_DTC_Control()を使用して、DTC の動作状態や現処理の起動割り込みのベクタ番号などの DTC のステータスを取得できます。また、R_DTC_Control()関数を使用して実行中のチェーン転送処理を中断する機能やシーケンス転送処理を中断する機能もサポートしています。

DTC FIT モジュールの使用条件

以下に本モジュールの使用条件を示します。

- r_bsp でデフォルトのロック関数を使用する必要があります。
- DMAC と DTC のモジュールストップ設定ビットには共通のビットを使用する必要があります。

1.3　DTC FIT モジュールを使用する

1.3.1　DTC FIT モジュールを C++プロジェクト内で使用する

C++プロジェクトでは、FIT DTC 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_dtc_rx_if.h”
}
```

1.4　API の概要

表 1.1 に本モジュールに含まれる API 関数を示します。

表 1.1　API 関数一覧

関数	関数説明
R_DTC_Open()	初期化処理
R_DTC_Close()	終了処理
R_DTC_Create()	レジスタおよび起動要因設定処理
R_DTC_CreateSeq()	シーケンス転送用のレジスタおよび起動要因設定処理
R_DTC_Control()	動作設定処理
R_DTC_GetVersion()	バージョン情報取得処理

1.5　DTC IP バージョン

表 1.2 に DTC IP バージョンと対象デバイスの関係について示します。

DTC IP バージョンの違いにより、R_DTC_Create()関数と R_DTC_CreateSeq()関数の引数仕様が異なります。詳細は「0

API 関数」を参照してください。

表 1.2　DTC IP バージョン一覧

DTC IP バージョン	対象デバイス
DTCa	• RX110 グループ、RX111 グループ、RX113 グループ、RX130 グループ • RX230 グループ、RX231 グループ、RX23T グループ、RX23W グループ、RX23E-A グループ、RX23E-B グループ、RX24T グループ、RX24U グループ • RX64M グループ、RX66T グループ • RX71M グループ、RX72T グループ
DTCb	• RX65N グループ、RX66N グループ、RX660 グループ、RX72M グループ、RX72N グループ、RX13T グループ、RX671 グループ、RX140 グループ、RX26T グループ、RX260 グループ、RX261 グループ

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。

- DTC（DTCa または DTCb）
- ICU

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。

- ボードサポートパッケージ (r_bsp) v5.20 以上

2.3 制限事項

2.3.1 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(e2 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(e2 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。

2.4 サポートされているツールチェーン

本モジュールは「6.1 動作確認環境」で示すツールチェーンで動作確認を行っています。

2.5　使用する割り込みベクタ

DTC 割り込みは R_DTC_Create()関数または R_DTC_CreateSeq() 関数を実行することで有効になります。

表 2.1 使用する割り込みベクター覧

関数名	引数	設定値	割り込み発生タイミング
R_DTC_Create() R_DTC_CreateSeq()	p_data_cfg->response_interrupt	DTC_INTERRUPT_AFTER_ALL_COMPLETE	指定した回数のデータ転送が終了したとき、CPU へ割り込み要求が発生
		DTC_INTERRUPT_PER_SINGLE_TRANSFER	データ転送のたびに、CPU への割り込み要求が発生

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フェース定義は r_dtc_rx_if.h に記載しています。

“DTC_VECTOR_TABLE_SIZE_BYTES” 定義を使って RAM 領域に DTC ベクタテーブル用のメモリを割り当てるときは、r_dtc_rx_target.h ファイルも、同様にインクルードされなければなりません。

2.7　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。

2.8　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_dtc_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。

Configuration options in r_dtc_rx_config.h	
#define DTC_CFG_PARAM_CHECKING_ENABLE ※デフォルト値は r_bsp_config.h ファイルで定義される “BSP_CFG_PARAM_CHECKING_ENABLE”の値となります。	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：パラメータチェック処理をコードから省略します。 • 1：パラメータチェック処理をコードに含めます。 システムのデフォルト設定を再使用するために、デフォルト値を“BSP_CFG_PARAM_CHECKING_ENABLE”に設定します。
#define DTC_CFG_DISABLE_ALL_ACT_SOURCE ※デフォルト値は“DTC_ENABLE”	R_DTC_OPEN()で DTCER レジスタをクリアするかどうかを設定します。 • DTC_DISABLE：処理なし。 • DTC_ENABLE：R_DTC_OPEN()ですべての DTCER レジスタをクリアします。
#define DTC_CFG_SHORT_ADDRESS_MODE ※デフォルト値は“DTC_DISABLE”	DTC でサポートするアドレスモードを設定します。 • DTC_DISABLE：フルアドレスモードを選択します。 • DTC_ENABLE：ショートアドレスモードを選択します。
#define DTC_CFG_TRANSFER_DATA_READ_SKIP_EN ※デフォルト値は“DTC_ENABLE”	転送情報リードスキップを許可するかどうかを設定します。 • DTC_DISABLE：転送情報リードスキップを禁止します。 • DTC_ENABLE：転送情報リードスキップを許可します。
#define DTC_CFG_USE_DMAC_FIT_MODULE ※デフォルト値は“DTC_ENABLE”	DTC FIT モジュールと一緒に DMAC FIT モジュールを使用するかどうかを設定します。 • DTC_DISABLE：DMAC FIT モジュールを使用しない。 • DTC_ENABLE：DMAC FIT モジュールを使用する。 DMAC FIT モジュールを使用しないときに“DTC_ENABLE”を設定すると、コンパイルエラーが発生します。
#define DTC_CFG_USE_SEQUENCE_TRANSFER ※デフォルト値は“DTC_DISABLE”	シーケンス転送を使用するかどうか設定します。 • DTC_DISABLE：シーケンス転送を使用しない。 • DTC_ENABLE：シーケンス転送を使用する。 本定義を“DTC_ENABLE”とした場合、DTC_CFG_SHORT_ADDRESS_MODE は“DTC_DISABLE”に設定してください。本定義と DTC_CFG_SHORT_ADDRESS_MODE の定義を共に“DTC_ENABLE”にした場合、コンパイルエラーが発生します。また、シーケンス転送未対応の MCU に対して本定義を“DTC_ENABLE”にした場合、コンパイルエラーが発生します。

2.9　コードサイズ

本モジュールのコードサイズを下表に示します。

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「2.8 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、「2.4 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。

ROM, RAM, and Stack Code Sizes

デバイス	分類	使用メモリ					
		ルネサス製コンパイラ		GCC		IAR コンパイラ	
RX111	ROM	1207 バイト		2680 バイト		2235 バイト	
	RAM	9 バイト ＋2,024 バイト （注 5、注 6）		8 バイト ＋2,024 バイト （注 5、注 6）		1044 バイト ＋2,024 バイト （注 5、注 6）	
	最大使用ユーザスタック	60 バイト		-			
	最大使用割り込みスタック	-		-			
RX231	ROM	1445 バイト		2948 バイト		2223 バイト	
	RAM	9 バイト ＋2,024 バイト （注 5、注 6）		8 バイト ＋2,024 バイト （注 5、注 6）		1044 バイト ＋2,024 バイト （注 5、注 6）	
	最大使用ユーザスタック	60 バイト		-		24 バイト	
	最大使用割り込みスタック	-		-		-	
RX23W	ROM	1413 バイト		-	-	-	-
	RAM	9 バイト ＋2,024 バイト （注 5、注 6）					
	最大ユーザスタック	60 バイト					
	最大割り込みスタック	-					
RX23E-A	ROM	1365 バイト		6476 バイト		2137 バイト	
	RAM	9 バイト ＋2,024 バイト （注 5、注 6）		2168 バイト ＋2,024 バイト （注 5、注 6）		1045 バイト ＋2,024 バイト （注 5、注 6）	
	最大ユーザスタック	64 バイト		-		56 バイト	
	最大割り込みスタック	-		-			
RX65N	ROM	1966 バイト（注 6）	2159 バイト（注 7）	3540 バイト（注 6）	3892 バイト（注 7）	2672 バイト（注 6）	2892 バイト（注 7）
	RAM	9 バイト	9 バイト	12 バイト	12 バイト	1045 バイト	1045 バイト

ROM, RAM, and Stack Code Sizes							
デバイス	分類	使用メモリ					
		ルネサス製コンパイラ		GCC		IAR コンパイラ	
		+2,048 バイト（注 5、注 6）	+3,072 バイト（注 5、注 7）	+2,048 バイト（注 5、注 6）	+3,072 バイト（注 5、注 7）	+2,048 バイト（注 5、注 6）	+3,072 バイト（注 5、注 7）
	最大使用ユーザスタック	64 バイト	64 バイト	-	-	172 バイト	176 バイト
	最大使用割り込みスタック	-	-	-	-	-	-
RX66T	ROM	1515 バイト (注 6)		3576 バイト (注 6)		2359 バイト (注 6)	
	RAM	9 バイト +2,048 バイト (注 5、6)		12 バイト +2,048 バイト (注 5、6)		1045 バイト +2,048 バイト (注 5、6)	
	最大ユーザスタック	60 バイト		-		24 バイト	
	最大割り込みスタック	-		-		-	
RX66N	ROM	1988 バイト（注 6）	2171 バイト（注 7）	7220 バイト（注 6）	7604 バイト（注 7）	2409 バイト（注 6）	2633 バイト（注 7）
	RAM	9 バイト +2,048 バイト（注 5、注 6）	9 バイト +3,072 バイト（注 5、注 7）	2208 バイト +2,048 バイト（注 5、注 7）	2208 バイト +3,072 バイト（注 5、注 7）	1045 バイト +2,048 バイト（注 5、注 6）	1045 バイト +3,072 バイト（注 5、注 7）
	最大使用ユーザスタック	52 バイト	52 バイト	-	-	60 バイト（注 6）	60 バイト（注 7）
	最大使用割り込みスタック	-	-	-	-	-	-
RX71M	ROM	1873 バイト		4392 バイト		2430 バイト	
	RAM	9 バイト +2,048 バイト（注 5、注 6）		12 バイト +2,048 バイト（注 5、注 6）		1045 バイト +2,048 バイト（注 5、注 6）	
	最大使用ユーザスタック	60 バイト		-		24 バイト	
	最大使用割り込みスタック	-		-		-	
RX72T	ROM	1,515 バイト		3076 バイト		2363 バイト	
	RAM	9 バイト +2,048 バイト（注 5、注 6）		12 バイト +2,048 バイト（注 5、注 6）		1045 バイト +2,048 バイト（注 5、注 6）	
	最大使用ユーザスタック	60 バイト		-		24 バイト	

ROM, RAM, and Stack Code Sizes							
デバイス	分類	使用メモリ					
		ルネサス製コンパイラ		GCC		IAR コンパイラ	
RX72M	ROM	1932 バイト（注 6）	2115 バイト（注 7）	7204 バイト（注 6）	7588 バイト（注 7）	2557 バイト（注 6）	2781 バイト（注 7）
	RAM	9 バイト＋2,048 バイト（注 5、注 6）	9 バイト＋3,072 バイト（注 5、注 7）	68 バイト＋2,048 バイト（注 5、注 6）	68 バイト＋3,072 バイト（注 5、注 7）	1045 バイト＋2,048 バイト（注 5、注 6）	1045 バイト＋3,072 バイト（注 5、注 7）
	最大使用ユーザスタック	64 バイト	64 バイト	-	-	180 バイト（注 6）	176 バイト（注 7）
	最大使用割り込みスタック	-	-	-	-	-	-
RX72N	ROM	1988 バイト（注 6）	2172 バイト（注 7）	7220 バイト（注 6）	7602 バイト（注 7）	2407 バイト（注 6）	2631 バイト（注 7）
	RAM	9 バイト＋2,048 バイト（注 5、注 6）	9 バイト＋3,072 バイト（注 5、注 7）	2172 バイト＋2,048 バイト（注 5、注 6）	2172 バイト＋3,072 バイト（注 5、注 7）	1045 バイト＋2,048 バイト（注 5、注 6）	1045 バイト＋3,072 バイト（注 5、注 7）
	最大使用ユーザスタック	52 バイト	52 バイト	-	-	60 バイト（注 6）	60 バイト（注 7）
	最大使用割り込みスタック	-	-	-	-	-	-
RX13T	ROM	1356 バイト（注 6）	1550 バイト（注 7）	6552 バイト（注 6）	6936 バイト（注 7）	-	-
	RAM	9 バイト＋2,048 バイト（注 5、注 6）	9 バイト＋3,072 バイト（注 5、注 7）	2172 バイト＋2,048 バイト（注 5、注 6）	2172 バイト＋3,072 バイト（注 5、注 7）	-	-
	最大使用ユーザスタック	64 バイト	64 バイト	-	-	-	-
	最大使用割り込みスタック	-	-	-	-	-	-
RX671	ROM	1982 バイト（注 6）	2157 バイト（注 7）	7280 バイト（注 6）	7664 バイト（注 7）	2407 バイト（注 6）	2631 バイト（注 7）
	RAM	9 バイト＋2,048 バイト	9 バイト＋3,072 バイト	68 バイト＋2,048 バイト	68 バイト＋3,072 バイト	2069 バイト＋2,048 バイト	2069 バイト＋3,072 バイト

ROM, RAM, and Stack Code Sizes							
デバイス	分類	使用メモリ					
		ルネサス製コンパイラ		GCC		IAR コンパイラ	
		（注 5、注 6）	（注 5、注 7）	（注 5、注 6）	（注 5、注 7）	（注 5、注 6）	（注 5、注 7）
	最大使用ユーザスタック	52 バイト	52 バイト	-	-	60 バイト	60 バイト
	最大使用割り込みスタック	-	-	-		-	
RX140	ROM	1409 バイト（注 6）	1589 バイト（注 7）	6644 バイト（注 6）	7028 バイト（注 7）	2223 バイト（注 6）	2447 バイト（注 7）
	RAM	9 バイト＋2,048 バイト（注 5、注 6）	9 バイト＋3,072 バイト（注 5、注 7）	64 バイト＋2,048 バイト（注 5、注 6）	64 バイト＋3,072 バイト（注 5、注 7）	1045 バイト＋2,048 バイト（注 5、注 6）	1045 バイト＋3,072 バイト（注 5、注 7）
	最大使用ユーザスタック	52 バイト	52 バイト	-	-	56 バイト	56 バイト
	最大使用割り込みスタック	-	-	-		-	
RX660	ROM	1871 バイト（注 6）	2045 バイト（注 7）	5044 バイト（注 6）	5100 バイト（注 7）	2399 バイト（注 6）	2455 バイト（注 7）
	RAM	9 バイト＋2,048 バイト（注 5、注 6）	9 バイト＋3,072 バイト（注 5、注 7）	2176 バイト＋2,048 バイト（注 5、注 6）	2176 バイト＋3,072 バイト（注 5、注 7）	1045 バイト＋2,048 バイト（注 5、注 6）	1045 バイト＋3,072 バイト（注 5、注 7）
	最大使用ユーザスタック	52 バイト	52 バイト	-	-	60 バイト	60 バイト
	最大使用割り込みスタック	-	-	-		-	
RX26T	ROM	1741 バイト（注 6）	1915 バイト（注 7）	6136 バイト（注 6）	6352 バイト（注 7）	2401 バイト（注 6）	2626 バイト（注 7）
	RAM	9 バイト＋2,048 バイト（注 5、注 6）	9 バイト＋3,072 バイト（注 5、注 7）	2176 バイト＋2,048 バイト（注 5、注 6）	2176 バイト＋3,072 バイト（注 5、注 7）	1045 バイト＋2,048 バイト（注 5、注 6）	1045 バイト＋3,072 バイト（注 5、注 7）
	最大使用ユーザスタック	52 バイト	52 バイト	-	-	60 バイト	60 バイト
	最大使用割り込みスタック	-	-	-		-	

ROM, RAM, and Stack Code Sizes							
デバイス	分類	使用メモリ					
		ルネサス製コンパイラ		GCC		IAR コンパイラ	
RX23E-B	ROM	1368 バイト（注 6）	1289 バイト（注 7）	5672 バイト（注 6）	5560 バイト（注 7）	1044 バイト（注 6）	1044 バイト（注 7）
	RAM	9 バイト +2,048 バイト（注 5、注 6）	9 バイト +3,072 バイト（注 5、注 7）	2168 バイト +2,048 バイト（注 5、注 6）	2168 バイト +3,072 バイト（注 5、注 7）	2190 バイト +2,048 バイト（注 5、注 6）	2086 バイト +3,072 バイト（注 5、注 7）
	最大使用ユーザスタック	48 バイト	48 バイト	-	-	56 バイト	56 バイト
	最大使用割り込みスタック	-	-	-		-	
RX260	ROM	1646 バイト（注 6）	1820 バイト（注 7）	3952 バイト（注 6）	4168 バイト（注 7）	2408 バイト（注 6）	2640 バイト（注 7）
	RAM	9 バイト +2,048 バイト（注 5、注 6）	9 バイト +3,072 バイト（注 5、注 7）	2168 バイト +2,048 バイト（注 5、注 6）	2168 バイト +3,072 バイト（注 5、注 7）	2068 バイト +2,048 バイト（注 5、注 6）	2068 バイト +3,072 バイト（注 5、注 7）
	最大使用ユーザスタック	52 バイト	52 バイト	-	-	56 バイト（注 6）	56 バイト（注 7）
	最大使用割り込みスタック	-	-	-		-	
RX261	ROM	1646 バイト（注 6）	1820 バイト（注 7）	3944 バイト（注 6）	4160 バイト（注 7）	2295 バイト（注 6）	2527 バイト（注 7）
	RAM	9 バイト +2,048 バイト（注 5、注 6）	9 バイト +3,072 バイト（注 5、注 7）	2168 バイト +2,048 バイト（注 5、注 6）	2168 バイト +3,072 バイト（注 5、注 7）	2068 バイト +2,048 バイト（注 5、注 6）	2068 バイト +3,072 バイト（注 5、注 7）
	最大使用ユーザスタック	52 バイト	52 バイト	-	-	56 バイト（注 6）	56 バイト（注 7）
	最大使用割り込みスタック	-	-	-		-	

注 1：「2.8 コンパイル時の設定」のデフォルト設定を選択した場合の値です。選択する定義により、コードサイズは異なります。

注 2：動作条件は以下のとおりです。

- r_dtc_rx.c
- r_dtc_rx_target.c

注 3：必要メモリサイズは、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。
注 4：リトルエンディアン時の値です。エンディアンにより、上記のメモリサイズは、異なります。
注 5：DTC FIT モジュールは、malloc()関数を使用し、DTC ベクタテーブル、および、DTC インデックステーブルに必要なメモリを確保します。このメモリサイズは、ターゲット MCU の r_dtc_rx_target.h にある“#define DTC_VECTOR_TABLE_SIZE_BYTES”により決まります。
注 6：DTC_CFG_USE_SEQUENCE_TRANSFER が DTC_DISABLE の場合
注 7：DTC_CFG_USE_SEQUENCE_TRANSFER が DTC_ENABLE の場合

2.10 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_dtc_rx_if.h に記載されています。

2.10.1 r_dtc_rx_if.h

```
/* Short-address mode */
typedef struct st_transfer_data { /* 3 long words */
    uint32_t lw1;
    uint32_t lw2;
    uint32_t lw3;
} dtc_transfer_data_t;

/* Full-address mode */
typedef struct st_transfer_data { /* 4 long words */
    uint32_t lw1;
    uint32_t lw2;
    uint32_t lw3;
    uint32_t lw4;
} dtc_transfer_data_t;

/* Transfer data configuration */
/* Moved struct dtc_transfer_data_cfg_t to r_dtc_rx_target_if.h */

typedef enum e_dtc_command {
    DTC_CMD_DTC_START,     /* DTC will accept activation requests.    */
    DTC_CMD_DTC_STOP,   /* DTC will not accept new activation request.   */
    DTC_CMD_ACT_SRC_ENABLE,
      /* Enable an activation source specified by vector number.   */
    DTC_CMD_ACT_SRC_DISABLE,
      /* Disable an activation source specified by vector number.  */
    DTC_CMD_DATA_READ_SKIP_ENABLE,  /* Enable Transfer Data Read Skip.      */
    DTC CMD DATA READ SKIP DISABLE, /* Disable Transfer Data Read Skip.     */
    DTC CMD STATUS GET,       /* Get the current status of DTC.    */
    DTC CMD CHAIN TRANSFER ABORT
            /* Abort the current Chain transfer process. */
    DTC CMD SEQUENCE TRANSFER ENABLE/* Enable sequence transfer    */
    DTC CMD SEQUENCE TRANSFER DISABLE  /* Disable Sequence transfer      */
    DTC CMD SEQUENCE TRANSFER ABORT    /* Abort sequence transfer    */
} dtc_command_t;
```

2.10.2 r_dtc_rx_target_if.h

dtc_transfer_data_cfg_t は DTC の IP Version により定義が異なります。

2.10.1.1 DTCa の場合

```
 typedef struct st_dtc_transfer_data_cfg {
     dtc_transfer_mode_t     transfer_mode;    /* DTC transfer mode          */
     dtc_data_size_t         data_size;        /* Size of data               */
     dtc_src_addr_mode_t     src_addr_mode;    /* Address mode of source     */
     dtc_chain_transfer_t    chain_transfer_enable;
                                       /* Chain transfer is enabled or not  */
     dtc_chain_transfer_mode_t chain_transfer_mode;
                                     /* How chain transfer is performed    */
     dtc_interrupt_t         response_interrupt;
                                     /* How response interrupt is raised   */
     dtc_repeat_block_side_t repeat_block_side;/* Side being repeat or block */
     dtc_dest_addr_mode_t   dest_addr_mode;    /* Address mode of destination*/
     uint32_t                source_addr;      /* Start address of source    */
     uint32_t                dest_addr;  /* Start address of destination     */
     uint32_t                transfer_count;   /* Transfer count             */
     uint16_t                block_size;
                                   /* Size of a block in block transfer mode */
     uint16_t                rsv;              /* Reserve bit                */
} dtc_transfer_data_cfg_t;
```

2.10.1.2 DTCb の場合

```
 typedef struct st_dtc_transfer_data_cfg {
     dtc_transfer_mode_t     transfer_mode;    /* DTC transfer mode          */
     dtc_data_size_t         data_size;        /* Size of data               */
     dtc_src_addr_mode_t     src_addr_mode;    /* Address mode of source     */
     dtc_chain_transfer_t    chain_transfer_enable;
                                       /* Chain transfer is enabled or not  */
     dtc_chain_transfer_mode_t chain_transfer_mode;
                                       /* How chain transfer is performed   */
     dtc_interrupt_t         response_interrupt;
                                       /* How response interrupt is raised  */
     dtc_repeat_block_side_t repeat_block_side;/* Side being repeat or block */
     dtc_dest_addr_mode_t    dest_addr_mode;   /* Address mode of destination*/
     uint32_t                source_addr;      /* Start address of source    */
     uint32_t                dest_addr;  /* Start address of destination     */
     uint32_t                transfer_count;   /* Transfer count             */
     uint16_t                block_size;
                                   /* Size of a block in block transfer mode */
     uint16_t                rsv;              /* Reserve bit                */
     dtc_write_back_t        writeback_disable;
                        /* Transfer information writeback is enabled or not  */
     dtc_sequence_end_t      sequence_end;
                          /* Sequence transfer is continued or end          */
     dtc_refer_index_table_t refer_index_table_enable;
                          /* Index table reference is enabled or not        */
     dtc_disp_add_t          disp_add_enable;
                 /* Displacement value is added to the source address or not */
} dtc_transfer_data_cfg_t;
```

2.11 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_dtc_rx_if.h で記載されています。

```
typedef enum e_dtc_err       /* DTC API error codes */
{
    DTC_SUCCESS_DMAC_BUSY = 0,
   /* One or some DMAC resources are locked by another process. */
    DTC_SUCCESS,
    DTC_ERR_OPENED,         /* DTC was initialized already.      */
    DTC_ERR_NOT_OPEN,       /* DTC module is not initialized yet.   */
    DTC_ERR_INVALID_ARG,    /* Arguments are invalid.         */
    DTC_ERR_INVALID_COMMAND,      /* Command parameters are invalid.      */
    DTC_ERR_NULL_PTR,       /* Argument pointers are NULL.    */
    DTC_ERR_BUSY     /* The DTC resources are locked by another process. */
    DTC_ERR_ACT          /* Data transfer is in progress      */
} dtc_err_t;
```

2.12 コールバック関数

DTC FIT モジュールではコールバック関数を使用しません。

2.13 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、Smart Configurator を使用した(1)、(2)の追加方法を推奨しています。ただし、Smart Configurator 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(3)の方法を使用してください。

(1) e² studio 上で Smart Configurator を使用して FIT モジュールを追加する場合
e² studio の Smart Configurator 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「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(2) CS+上で Smart Configurator 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 Smart Configurator 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「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(3) 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。

2.14 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 */
```

3. API 関数

R_DTC_Open()

DTC FIT モジュールの API 使用時に、最初に実行される関数です。

Format

```
dtc_err_t       R_DTC_Open ( void )
```

Parameters

なし

Return Values

```
[DTC_SUCCESS]       /* 正常終了 */
[DTC_ERR_OPENED]    /* DTC は既に初期化されています。 */
[DTC_ERR_BUSY]      /* リソースは他のプロセスによってロックされています。 */
```

Properties

ファイル r_dtc_rx_if.h にプロトタイプ宣言されています。

Description

この関数は、DTC をロックし（注 1）、DTC への電源供給を開始し、DTC ベクタテーブル、アドレスモード、転送情報リードスキップ機能の設定を初期化します。また、r_dtc_rx_config.h 内で DTC_CFG_DISABLE_ALL_ACT_SOURCE を DTC_ENABLE に設定した場合、すべての DTCER レジスタをクリアします。DTC_CFG_USE_SEQUENCE_TRANSFER を DTC_ENABLE に設定した場合、DTC インデックステーブルで使用する領域を確保します。

注1. DTC FIT モジュールは r_bsp のデフォルトのロック機能を使用しています。そのため、処理が正常に終了すると、DTC はロックされた状態になります。

Example

```
dtc_err_t ret;
/* Call R_DTC_Open() */
ret = R_DTC_Open();
```

Special Notes:

r_bsp_config.h の#define BSP_CFG_HEAP_BYTES には、r_dtc_rx_target.h の#define DTC_VECTOR_TABLE_SIZE_BYTES より大きい値を設定してください。これは、DTC FIT モジュールで malloc()関数を使用して、DTC ベクタテーブルの領域を確実に確保するためです。

R_DTC_Close()

この関数は、DTC のリソースを開放します。

Format

dtc_err_t R_DTC_Close (void)

Parameters

なし

Return Values

[DTC_SUCCESS]	*/* 正常終了 */*
[DTC_SUCCESS_DMAC_BUSY]	*/* 正常終了。 1 つ以上の DMAC のリソースがロックされています。 */*

Properties

ファイル r_dtc_rx_if.h にプロトタイプ宣言されています。

Description

この関数は、DTC のロックを解除し（注 1）、DTC 起動許可レジスタ（DTCERn）をクリアして、すべての DTC 起動要因を禁止にします。DTC へのクロック供給を停止し、DTC はモジュールストップ状態へ遷移します。

さらに、DMAC のすべてのチャネルのロックが解除されていた場合、本関数は DMAC と DTC をモジュールストップ状態に遷移させます（注 2）。

注1. DTC FIT モジュールは r_bsp のデフォルトのロック機能を使用しています。そのため、処理が正常に終了すると、DTC はロック解除された状態になります。

注2. DMAC および DTC のモジュールストップ設定ビットとして共用ビットが使用されるため、本関数では、モジュールストップ状態を設定する前に、すべての DMAC チャネルのロックが解除されていることを確認します。詳細はユーザーズマニュアル ハードウェア編の「消費電力低減機能」章をご覧ください。

以下を参照し、使用するモジュールの組み合わせに応じて処理方法を変更してください。

DMAC コントロール	DTC コントロール	処理方法
DMACA FIT モジュール （ロック機能コントロール関数および DTC ロック状態確認関数がある）	DTC FIT モジュール （ロック機能コントロール関数および DMAC ロック状態確認関数がある）	処理 1
上記以外		処理 2

処理 1: r_bsp のデフォルトのロック関数を使用し、DMAC FIT モジュール（注 1）で DMAC を制御する

関数は、r_bsp のデフォルトのロック関数を使用して、DMAC の全チャネルおよび DTC のロックが解除されていることを確認し、DTC をモジュールストップ状態に遷移させます。

注1. DMAC FIT モジュールがモジュールストップコントロール関数（DTC がロック状態であることを確認する関数）を備えていることが、この処理の必要条件となります。

処理 2: 上記以外の方法による制御

ユーザは、DMAC の全チャネルのロックが解除されていること、および DTC のロックが解除されている（使用中でない）ことを確認するためのコードを提供する必要があります。DTC FIT モジュールには、この処理用に空関数が用意されています。

r_bsp のデフォルトのロック機能を使用しない場合、r_dtc_rx_target.c ファイルの r_dtc_check_DMAC_locking_byUSER()関数で“/* do something */”とコメントが入っている行の後に、DMAC の全チャネルおよび DTC のロック／ロック解除を確認するためのプログラムコードを挿入してください。

なお、r_dtc_check_DMAC_locking_byUSER()関数の戻り値には、以下に示すブール型を使用してください。

r_dtc_check_DMAC_locking_byUSER()の戻り値

[true]	*/* DMAC の全チャネルのロックが解除されています。*/*
[false]	*/* 1 つ以上の DMAC のチャネルがロックされています。*/*

Example

```
dtc_err_t ret;
ret = R_DTC_Close();
```

Special Notes:

DMAC FIT モジュールを使用せずに DMAC を制御する場合は、本関数の呼び出しによって DMAC がモジュールストップ状態に遷移されないように、DMAC の使用状態を監視し、DMAC のロック／ロック解除を制御してください。DMAC 転送設定を行わない時は、DMAC が動作中でなくても、DMAC はロックされている必要があります。

R_DTC_Create()

この関数は、DTC レジスタの設定と起動要因の設定を行います。

Format

```
dtc_err_t           R_DTC_Create (
        dtc_activation_source_t     act_source,
        dtc_transfer_data_t         *p_transfer_data,
        dtc_transfer_data_cfg_t     *p_data_cfg,
        uint32_t                    chain_transfer_nr
)
```

Parameters

dtc_activation_source_t act_source
起動要因

*dtc_transfer_data_t *p_transfer_data*
RAM の転送情報領域の開始アドレスへのポインタ

*dtc_transfer_data_cfg_t *p_data_cfg*
転送情報設定へのポインタ

DTCb の場合、以下の構造体メンバへの設定は無効であり、本関数内で以下の値を設定します。

```
p_data_cfg->writeback_disable = DTC_WRITEBACK_ENABLE;
p_data_cfg->sequence_end = DTC_SEQUENCE_TRANSFER_CONTINUE;
p_data_cfg->refer_index_table_enable = DTC_REFER_INDEX_TABLE_DISABLE;
p_data_cfg->disp_add_enable = DTC_SRC_ADDR_DISP_ADD_DISABLE;
```

uint32_t chain_transfer_nr
チェーン転送数

転送情報数とそれに対応する設定情報は“チェーン転送数 + 1”になります。例えば、chain_transfer_nr = 1 のとき、連続する転送情報が 2 つ、それに対応する設定情報が 2 つあることになり、最初の設定情報でチェーン転送が有効になります。

転送情報（*p_transfer_data*）の型定義はアドレスモードに依存します（詳細は以下参照）。ユーザはこのデータ型を使って転送情報を正しくメモリに配置します。

```
#if (1 == DTC_CFG_SHORT_ADDRESS_MODE) /* Short address mode */
typedef struct st_transfer_data { /* 3 long words */
    uint32_t lw1;
    uint32_t lw2;
    uint32_t lw3;
} dtc_transfer_data_t;
#else /* Full-address mode */
typedef struct st_transfer_data { /* 4 long words */
    uint32_t lw1;
    uint32_t lw2;
    uint32_t lw3;
    uint32_t lw4;
} dtc_transfer_data_t;
#endif
```

「転送情報設定へのポインタ（* p_data_cf）」の型は、DTC IP バージョンにより異なります。設定情報のデータ構造体を以下に示します。

1. DTCa の場合

```
typedef struct st_dtc_transfer_data_cfg {
  dtc_transfer_mode_t transfer_mode; /* DTC transfer mode     */
  dtc_data_size_t     data_size;     /* Size of data       */
  dtc_src_addr_mode_t src_addr_mode;    /* Address mode of source   */
  dtc_chain_transfer_t   chain_transfer_enable;
                                          /* Chain transfer is enabled or not */
  dtc_chain_transfer_mode_t chain_transfer_mode;
/* How chain transfer is performed   */
  dtc_interrupt_t     response_interrupt;
/* How response interrupt is raised */
  dtc_repeat_block_side_t repeat_block_side;/* Side being repeat or block */
  dtc_dest_addr_mode_t   dest_addr_mode;   /* Address mode of destination*/
  uint32_t         source_addr;/* Start address of source */
  uint32_t         dest_addr;  /* Start address of destination   */
  uint32_t         transfer_count;/* Transfer count        */
  uint16_t         block_size; /* Size of a block in block transfer mode  */
  uint16_t         rsv;        /* Reserve bit        */
} dtc_transfer_data_cfg_t;
```

2. DTCb の場合

```
typedef struct st_dtc_transfer_data_cfg {
  dtc_transfer_mode_t transfer_mode; /* DTC transfer mode     */
  dtc_data_size_t     data_size;     /* Size of data       */
  dtc_src_addr_mode_t src_addr_mode; /* Address mode of source   */
  dtc_chain_transfer_t   chain_transfer_enable;
                        /* Chain transfer is enabled or not   */
  dtc_chain_transfer_mode_t chain_transfer_mode;
                     /* How chain transfer is performed */ dtc_interrupt_t
  response_interrupt;
                     /* How response interrupt is raised   */
  dtc_repeat_block_side_t repeat_block_side;/* Side being repeat or block   */
  dtc_dest_addr_mode_t   dest_addr_mode;/* Address mode of destination*/
  uint32_t         source_addr;/* Start address of source */
  uint32_t         dest_addr;  /* Start address of destination   */
  uint32_t         transfer_count;/* Transfer count        */
  uint16_t         block_size; /* Size of a block in block transfer mode */
  uint16_t         rsv;        /* Reserve bit        */
  dtc_write_back_t    writeback_disable;
          /* Transfer information writeback is enabled or not   */
  dtc_sequence_end_t  sequence_end;
              /* Sequence transfer is continued or end */
  dtc_refer_index_table_t   refer_index_table_enable;
              /* Index table reference is enabled or not  */ dtc_disp_add_t
  disp_add_enable;
      /* Displacement value is added to the source address or not */
} dtc_transfer_data_cfg_t;
```

以下の列挙型の定義で、上記構造体の設定可能なオプションを示します。

```
/* Configurable options for DTC Transfer mode */
typedef enum e_dtc_transfer_mode
{
    DTC_TRANSFER_MODE_NORMAL = (0),         /* = (0 << 6):Normal mode */
    DTC_TRANSFER_MODE_REPEAT = (1 << 6),    /* Repeat mode */
    DTC_TRANSFER_MODE_BLOCK  = (2 << 6),    /* Block mode */
} dtc_transfer_mode_t;

/* Configurable options for DTC Data transfer size */
typedef enum e_dtc_data_size
{
    DTC_DATA_SIZE_BYTE  = (0),          /* = (0 << 4):8-bit (byte) data */
    DTC_DATA_SIZE_WORD  = (1 << 4),     /* 16-bit (word) data */
    DTC_DATA_SIZE_LWORD = (2 << 4),     /* 32-bit (long word) data */
} dtc_data_size_t;

/* Configurable options for Source address addressing mode */
typedef enum e_dtc_src_addr_mode
{
    DTC_SRC_ADDR_FIXED =  (0),/* = (0 << 2):Source address is fixed.*/
    DTC_SRC_ADDR_INCR  =  (2 << 2),
/* Source address is incremented after each transfer.*/
    DTC_SRC_ADDR_DECR  =  (3 << 2),
/* Source address is decremented after each transfer.*/
} dtc_src_addr_mode_t;

/* Configurable options for Chain transfer */
typedef enum e_dtc_chain_transfer
{
    DTC_CHAIN_TRANSFER_DISABLE      = (0),       /* Disable Chain transfer.*/
    DTC_CHAIN_TRANSFER_ENABLE       = (1 << 7),  /* Enable Chain transfer.*/
} dtc_chain_transfer_t;

/* Configurable options for how chain transfer is performed */
typedef enum e_dtc_chain_transfer_mode
{
    DTC_CHAIN_TRANSFER_CONTINUOUSLY = (0),
      /* = (0 << 6):Chain transfer is performed continuously.*/
    DTC_CHAIN_TRANSFER_NORMAL       = (1 << 6)
/* Chain transfer is performed only when the counter is changed to 0 or
CRAH.*/
} dtc_chain_transfer_mode_t;

/* Configurable options for Interrupt */
typedef enum e_dtc_interrupt
{
    DTC_INTERRUPT_AFTER_ALL_COMPLETE  = (0),
/* Interrupt is generated when specified data transfer is completed.*/
    DTC_INTERRUPT_PER_SINGLE_TRANSFER = (1 << 5)
/* Interrupt is generated when each transfer time is completed.*/
} dtc_interrupt_t;
```

```
/* Configurable options for Side to be repeat or block */
typedef enum e_dtc_repeat_block_side
{
    DTC_REPEAT_BLOCK_DESTINATION = (0),
/* = (0 << 4):Destination is repeat or block area.*/
    DTC_REPEAT_BLOCK_SOURCE  = (1 << 4)
/* Source is repeat or block area.*/
} dtc_repeat_block_side_t;


/* Configurable options for Destination address addressing mode */
typedef enum e_dtc_dest_addr_mode
{
    DTC_DES_ADDR_FIXED = (1 << 2),  /* Destination address is fixed.*/
    DTC_DES_ADDR_INCR  = (2 << 2),
/* Destination address is incremented after each transfer.*/
    DTC_DES_ADDR_DECR  = (3 << 2)
/* Destination address is decremented after each transfer.*/
} dtc_dest_addr_mode_t;

/* Configurable options to write back transfer information */
typedef enum e_dtc_write_back
{
  DTC_WRITEBACK_ENABLE  = (0), /* Writeback is enabled */
  DTC_WRITEBACK_DISABLE = (1)  /* Writeback is disabled */
} dtc write back t;

/* Configurable option to continue/end sequence transfer */
typedef enum e dtc sequence end
{
  DTC SEQUENCE TRANSFER CONTINUE = (0), /* Sequence transfer is continued */
  DTC SEQUENCE TRANSFER END      = (1)  /* Sequence transfer is ended */
} dtc sequence end t;

/* Configurable options for index table reference */
typedef enum e dtc refer index table
{
  DTC REFER INDEX TABLE DISABLE = (0),    /* Index table is not referred */
  DTC_REFER_INDEX_TABLE_ENABLE  = (1 << 1) /* Index table is referred */
} dtc_refer_index_table_t;

/* Configurable options to add/not to add Displacement value to the
destination address */
typedef enum e_dtc_disp_add
{
  DTC_SRC_ADDR_DISP_ADD_DISABLE = (0),
       /* Displacement value is not added to the source address */
  DTC_SRC_ADDR_DISP_ADD_ENABLE  = (1)
       /* Displacement value is added to the source address */
} dtc_disp_add_t;
```

p_data_cfg->transfer_count には、ノーマル転送モードとブロック転送モードでは 1～65536 の値が、リピート転送モードでは 1～256 の値が設定されます。

p_data_cfg->block_size には、ブロック転送モードで 1～256 の値が設定されます。

ショートアドレスモードでは、転送情報の開始アドレス（第 2 引数）、転送元領域、転送先領域は 0x00000000～0x007FFFFF および 0xFF800000～0xFFFFFFFF の範囲内で設定されます。

Return Values

[DTC_SUCCESS]	*/* 正常終了 */*
[DTC_ERR_NOT_OPEN]	*/* DTC は未初期化状態です。 */*
[DTC_ERR_INVALID_ARG]	*/* 引数は無効な値です。 */*
[DTC_ERR_NULL_PTR]	*/*引数のポインタが NULL です。 */*

Properties

ファイル r_dtc_rx_if.h にプロトタイプ宣言されています。

Description

転送情報に設定情報を書き込みます。

割り込み番号に対応する転送情報の先頭アドレスを DTC ベクタテーブルに書き込みます。

Example

処理 1: チェーン転送を行わない場合

```
dtc_transfer_data_cfg_t td_cfg;
dtc_activation_source_t act_src = DTCE_ICU_SWINT; /* activation source is
Software Interrupt */

dtc_transfer_data_t  transfer_data;  /* assume that DTC address mode is full
mode */

dtc_err_t ret;
uint32_t src = 1234;
uint32_t des[3];
uint8_t  ien_bk;

/* create the configuration - no chain transfer */
/* Source address addressing mode is FIXED
 * Data size is 32 bits (long word)
 * DTC transfer mode is Repeat mode & Source side is repeat area
 * Interrupt is raised after each single transfer
 * Chain transfer is disabled
*/
td_cfg.src_addr_mode         = DTC_SRC_ADDR_FIXED;
td cfg.data size             = DTC DATA SIZE LWORD;
td cfg.transfer mode         = DTC TRANSFER MODE REPEAT;
td cfg.dest addr mode        = DTC DES ADDR INCR;
td cfg.repeat block side     = DTC REPEAT BLOCK SOURCE;
td cfg.response interrupt    = DTC INTERRUPT PER SINGLE TRANSFER;
td cfg.chain transfer enable = DTC CHAIN TRANSFER DISABLE;
td cfg.chain transfer mode   = (dtc chain transfer mode t)0;

td cfg.source addr           = (uint32 t)&src;
td cfg.dest addr             = (uint32 t)des;
td cfg.transfer count        = 1;
td cfg.block size            = 3;

/* Disable Software interrupt request before calling R_DTC_Create() */
ien_bk = ICU.IER[3].BIT.IEN3 ; /* store old setting */
ICU.IER[3].BIT.IEN3 = 0;
```

```
/* Calling to R_DTC_Create() */

ret = R_DTC_Create(act_src, &transfer_data, &td_cfg, 0);

/* Restore the setting for Software interrupt request */
ICU.IER[3].BIT.IEN3 = ien_bk;
```

処理 2: チェーン転送を 1 回行う場合

```
dtc transfer data cfg t td cfg[2]; /* need 2 configuration sets */
dtc activation source t act src = DTCE ICU SWINT;
/* activation source is Software Interrupt */
uint32_t transfer_data[8];
/* for 2 Transfer data; assume that DTC address mode is full mode */
dtc_err_t ret;
uint32_t src = 1234;
uint32_t des[3];        /* The destination for first Transfer data */
uint32_t des2[3];       /* The destination for second Transfer data */
uint8_t  ien_bk;

/* create the configuration 1 - support chain transfer */
/* Source address addressing mode is FIXED
 * Destination address addressing mode is INCREMENTED
 * Data size is 32 bits (long word)
 * DTC transfer mode is Normal mode
 * Interrupt is raised after each single transfer
 * Chain transfer is enabled
 * Chain transfer is performed after when transfer counter is set to 0
*/
td_cfg[0].src_addr_mode         = DTC_SRC_ADDR_FIXED;
td_cfg[0].data_size             = DTC_DATA_SIZE_LWORD;
td_cfg[0].transfer_mode         = DTC_TRANSFER_MODE_NORMAL;
td_cfg[0].dest_addr_mode        = DTC_DES_ADDR_INCR;
td_cfg[0].repeat_block_side     = DTC_REPEAT_BLOCK_SOURCE;
td_cfg[0].response_interrupt    = DTC_INTERRUPT_PER_SINGLE_TRANSFER;
td_cfg[0].chain_transfer_enable = DTC_CHAIN_TRANSFER_ENABLE;
td_cfg[0].chain_transfer_mode   = DTC_CHAIN_TRANSFER_NORMAL;

td_cfg[0].source_addr           = (uint32_t)&src;
td_cfg[0].dest_addr       = (uint32_t)des; /* transfer from source to des 1 */
td_cfg[0].transfer_count        = 1;
td_cfg[0].block_size            = 3;

/* create the configuration 2 - no chain transfer */
/* Source address addressing mode is FIXED
 * Destination address addressing mode is INCREMENTED
 * Data size is 32 bits (long word)
 * DTC transfer mode is Normal mode
 * Interrupt is raised after each single transfer
 * Chain transfer is disabled*/
```

```
td_cfg[1].src_addr_mode          = DTC_SRC_ADDR_FIXED;
td_cfg[1].data_size              = DTC_DATA_SIZE_LWORD;
td_cfg[1].transfer_mode          = DTC_TRANSFER_MODE_NORMAL;
td_cfg[1].dest_addr_mode         = DTC_DES_ADDR_INCR;
td_cfg[1].repeat_block_side      = DTC_REPEAT_BLOCK_SOURCE;
td_cfg[1].response_interrupt     = DTC_INTERRUPT_PER_SINGLE_TRANSFER;
td_cfg[1].chain_transfer_enable = DTC_CHAIN_TRANSFER_DISABLE;
td_cfg[1].chain_transfer_mode    = (dtc_chain_transfer_mode_t)0;

td_cfg[1].source_addr            = (uint32_t)&src;
td_cfg[1].dest_addr       = (uint32_t)des2; /* transfer from source to des 2*/
td_cfg[1].transfer_count         = 1;
td_cfg[1].block_size             = 3;
/* Disable Software interrupt request before calling R_DTC_Create() */
ien_bk = ICU.IER[3].BIT.IEN3 ; /* store old setting */
ICU.IER[3].BIT.IEN3 = 0;

/* Call R_DTC_Create() */
ret = R_DTC_Create(act_src, transfer_data , td_cfg, 1); /* The fourth argument
indicates that there's one chain transfer enabled in first Transfer data */

/* Restore the setting for Software interrupt request */
ICU.IER[3].BIT.IEN3 = ien_bk;
```

処理 3: 複数要因の登録を行う場合

```
dtc transfer data cfg t td cfg sw;
dtc transfer data cfg t td cfg cmt;
dtc activation source t act src sw = DTCE ICU SWINT;
/* activation source is Software Interrupt */
dtc activation source t act src cmt = DTCE CMT0 CMI0;
/* activation source is CMT Interrupt */
dtc transfer data t transfer data sw;
/* assume that DTC address mode is full mode */
dtc_transfer_data_t transfer_data_cmt;
/* assume that DTC address mode is full mode */

dtc_err_t ret;
uint32_t src_sw = 1234;
uint32_t src_cmt = 5678;
uint32_t des_sw[3];
uint32_t des_cmt[3];
uint8_t  ien_bk;

/* create the configuration - no chain transfer */
/* Source address addressing mode is FIXED
* Data size is 32 bits (long word)
* DTC transfer mode is Repeat mode & Source side is repeat area
* Interrupt is raised after each single transfer
* Chain transfer is disabled
*/
```

```
td_cfg_sw.src_addr_mode = DTC_SRC_ADDR_FIXED;
td_cfg_sw.data_size = DTC_DATA_SIZE_LWORD;
td_cfg_sw.transfer_mode = DTC_TRANSFER_MODE_REPEAT;
td_cfg_sw.dest_addr_mode = DTC_DES_ADDR_INCR;
td_cfg_sw.repeat_block_side = DTC_REPEAT_BLOCK_SOURCE;
td_cfg_sw.response_interrupt = DTC_INTERRUPT_PER_SINGLE_TRANSFER;
td_cfg_sw.chain_transfer_enable = DTC_CHAIN_TRANSFER_DISABLE;
td_cfg_sw.chain_transfer_mode = (dtc_chain_transfer_mode_t)0;

td_cfg_sw.source_addr = (uint32_t)&src_sw;
td_cfg_sw.dest_addr = (uint32_t)des_sw;
td_cfg_sw.transfer_count = 1;
td_cfg_sw.block_size = 3;

/* Disable Software interrupt request before calling R_DTC_Create() */
ien_bk = ICU.IER[3].BIT.IEN3 ; /* store old setting */
ICU.IER[3].BIT.IEN3 = 0;

/* Calling to R_DTC_Create() */
ret = R_DTC_Create(act_src_sw, &transfer_data_sw, &td_cfg_sw, 0);
/* Restore the setting for Software interrupt request */
ICU.IER[3].BIT.IEN3 = ien_bk;

/* create the configuration - no chain transfer */
/* Source address addressing mode is FIXED
* Data size is 32 bits (long word)
* DTC transfer mode is Repeat mode & Source side is repeat area
* Interrupt is raised after each single transfer
* Chain transfer is disabled
*/
td_cfg_cmt.src_addr_mode = DTC_SRC_ADDR_FIXED;
td_cfg_cmt.data_size = DTC_DATA_SIZE_LWORD;
td_cfg_cmt.transfer_mode = DTC_TRANSFER_MODE_REPEAT;
td_cfg_cmt.dest_addr_mode = DTC_DES_ADDR_INCR;
td_cfg_cmt.repeat_block_side = DTC_REPEAT_BLOCK_SOURCE;
td_cfg_cmt.response_interrupt = DTC_INTERRUPT_PER_SINGLE_TRANSFER;
td_cfg_cmt.chain_transfer_enable = DTC_CHAIN_TRANSFER_DISABLE;
td_cfg_cmt.chain_transfer_mode = (dtc_chain_transfer_mode_t)0;

td_cfg_cmt.source_addr = (uint32_t)&src_cmt;
td_cfg_cmt.dest_addr = (uint32_t)des_cmt;
td_cfg_cmt.transfer_count = 1;
td_cfg_cmt.block_size = 3;

/* Calling to R_DTC_Create() */
ret = R_DTC_Create(act_src_cmt, &transfer_data_cmt, &td_cfg_cmt, 0);

R_CMT_CreateOneShot(10000, &cmt_callback, &cmt_channel);
```

Special Notes:

R_DTC_Create()を呼び出す前に、ユーザは割り込み要求許可ビット（IERm.IENj）をクリアし、処理対象の割り込み要求を禁止にする必要があります（割り込み要因は R_DTC_Create()に渡されます）。

```
ICU.IER[m].BIT.IENj = 0;
```

R_DTC_Create()の処理終了後に、禁止にした割り込み要求を許可します。

IERm.IENj ビットと割り込み要因の対応については、ユーザーズマニュアル ハードウェア編の割り込みコントローラ（ICU）章の「割り込みベクタテーブル」をご覧ください。

R_DTC_CreateSeq()

この関数は、シーケンス転送で使用する DTC レジスタと起動要因の設定を行います。

Format

```
dtc_err_t           R_DTC_CreateSeq(
                dtc_activation_source_t     act_source,
                dtc_transfer_data_t         *p_transfer_data,
                dtc_transfer_data_cfg_t     *p_data_cfg,
                uint32_t                    sequence_transfer_nr,
                uint8_t                     sequence_no
)
```

Parameters

dtc_activation_source_t　*act_source*

起動要因

dtc_transfer_data_t * *p_transfer_data*

RAM の転送情報領域の開始アドレスへのポインタ

dtc_transfer_data_cfg_t * *p_data_cfg*

転送情報設定へのポインタ

以下の構造体メンバも設定してください。

```
p_data_cfg->writeback_disable
p_data_cfg->sequence_end
p_data_cfg->refer_index_table_enable
p_data_cfg->disp_add_enable
```

uint32_t sequence_transfer_nr

1 シーケンス転送の転送情報数（0 - 4294967295）

sequence_transfer_nr	説明
0	指定したシーケンス番号（sequence_no）の転送要求が発生した場合、シーケンスを開始せずに CPU 割り込み要求を出力するための設定を行います。
1 - 4294967295	指定したシーケンス番号（sequence_no）の転送要求が発生した場合、シーケンス転送を行うための転送情報を設定します。 事前に sequence_transfer_nr に指定する数の転送情報を準備し、転送情報の先頭アドレスを*p_data_cfg に設定してください。

uint8_t *sequence_no*

シーケンス番号（0 - 255）

転送情報の型定義、データ構造体は R_DTC_Create()と同じです。全 256 とおりのシーケンス転送情報を設定することができます。

Return Values

```
DTC_SUCCESS             /* 正常終了 */
DTC_ERR_NOT_OPEN            /* DTC は未初期化状態です。*/
DTC_ERR_INVALID_ARG     /* 引数は無効な値です。*/
DTC_ERR_NULL_PTR            /*引数のポインタが NULL です。*/
```

Properties

ファイル r_dtc_rx_if.h にプロトタイプ宣言されています。

Description

転送情報に設定情報を書き込みます。

シーケンス番号に対応する転送情報の先頭アドレスを DTC インデックステーブルに書き込みます。

Example

受信 FIFO フル割り込み(以下、RXI)を DTC 起動要因として、シーケンス転送による調歩同期式シリアル受信を行う例を以下に説明します。使用する SCI はチャネル 10 です。外部通信デバイスから最初に受信した 1 バイトデータ（cmnd）に応じて自動的にシーケンス転送を開始します。

処理 1:

外部通信デバイスから cmnd=“00h”を受信後、SCI10 受信 FIFO しきい値を 4 バイトに変更し、外部通信デバイスから出力される 4 バイトのデータを受信し、DTC 転送によって RAM へ格納させる。

表 3-1 処理 1 で設定する転送情報

メンバ	転送情報 1	転送情報 2	転送情報 3
transfer_mode	ノーマル転送	ブロック転送	ノーマル転送
data_size	8 ビット	16 ビット	8 ビット
src_addr_mode	ソースアドレス固定	ソースアドレス固定	ソースアドレス固定
chain_transfer_enable	チェーン転送禁止	チェーン転送許可	チェーン転送禁止
chain_transfer_mode	チェーン転送を連続で実行（設定無効）	チェーン転送を連続で実行	チェーン転送を連続で実行（設定無効）
response_interrupt	指定したデータ転送が完了したら、割り込みを生成	指定したデータ転送が完了したら、割り込みを生成	指定したデータ転送が完了したら、割り込みを生成
repeat_block_side	転送先はリピートまたはブロック領域（設定無効）	転送先はリピートまたはブロック領域	転送先はリピートまたはブロック領域（設定無効）
dest_addr_mode	転送先アドレスは固定	転送ごとに、転送先アドレスをインクリメント	転送先アドレスは固定
source_addr	ROM の dtc_fcrh_data[0]のアドレス	SCI10.FRDR レジスタアドレス	ROM の g_dtc_fcrh_cmnd のアドレス
dest_addr	SCI10.FCR.H レジスタアドレス	RAM の g_dtc_rx_buf0[0]のアドレス	SCI10.FCR.H レジスタアドレス
transfer_count	1	1	1
block_size	（設定無効）	4	（設定無効）
writeback_disable	ライトバックしない	ライトバックしない	ライトバックしない
sequence_end	シーケンス転送を継続	シーケンス転送を継続	シーケンス転送を終了
refer_index_table_enable	インデックステーブルを参照しない	インデックステーブルを参照しない	インデックステーブルを参照しない
disp_add_enable	転送元アドレスにディスプレースメント値を加算しない	転送元アドレスにディスプレースメント値を加算しない	転送元アドレスにディスプレースメント値を加算しない

```
#include "platform.h"
#include "r_dtc_rx_if.h"

#define CMND0_RCV_NUM (4)
#define CMND0_RCV_FIFO_TRG (4)
#define CMND0_FCRH_DATA ((uint8_t)(0xF0 | CMND0_RCV_FIFO_TRG))
#define CMND0_INFO_NUM (3)

dtc_transfer_data_cfg_t g_dtc_pre_seqinfo_cmnd0[CMND0_INFO_NUM];
dtc_transfer_data_t g_dtc_seqinfo_cmnd0[CMND0_INFO_NUM];
uint16_t g_dtc_rx_buf0[CMND0_RCV_NUM];
const uint8_t g_dtc_fcrh_cmnd = 0xF1;
static const uint8_t dtc_fcrh_data[] =
{
CMND0_FCRH_DATA,
CMND1_FCRH_DATA,
CMND2_FCRH_DATA,
CMND3_FCRH_DATA
};

void dtc_pre_seqinfo_cmnd0_init(void);

void main(void)
{
dtc_err_t ret;
dtc activation source t act source;
uint32 t sequence transfer nr;
uint8 t  sequence no;
uint8 t ien bk;

…

/* ---- DTC sequence transfer information for Cmnd0 ---- */
dtc_pre_seqinfo_cmnd0_init();
act_source = DTCE_SCI10_RXI10;
sequence_transfer_nr = CMND0_INFO_NUM;
sequence_no = 0;
ien_bk  = IEN(SCI10,RXI10); /* IEN(x,x)->ICU.IER[z].BIT.IENz;*/
IEN(SCI10,RXI10) = 0;
ret = R_DTC_CreateSeq(act_source,
&g_dtc_seqinfo_cmnd0[0],
&g_dtc_pre_seqinfo_cmnd0[0],
sequence_transfer_nr,
sequence_no);
IEN(SCI10,RXI10)= ien_bk;

…
}

void dtc_pre_seqinfo_cmnd0_init(void)
{
/* [1st] Sequence transfer information –
Changing the SCI10 Receive FIFO trigger */
```

```
/* MRA */
g_dtc_pre_seqinfo_cmnd0[0].transfer_mode = DTC_TRANSFER_MODE_NORMAL;
g_dtc_pre_seqinfo_cmnd0[0].data_size = DTC_DATA_SIZE_BYTE;
g_dtc_pre_seqinfo_cmnd0[0].src_addr_mode = DTC_SRC_ADDR_FIXED;
g_dtc_pre_seqinfo_cmnd0[0].writeback_disable = DTC_WRITEBACK_DISABLE;
/* MRB */
g_dtc_pre_seqinfo_cmnd0[0].chain_transfer_enable =
DTC_CHAIN_TRANSFER_DISABLE;
g_dtc_pre_seqinfo_cmnd0[0].chain_transfer_mode =
DTC_CHAIN_TRANSFER_CONTINUOUSLY;
g_dtc_pre_seqinfo_cmnd0[0].response_interrupt =
DTC_INTERRUPT_AFTER_ALL_COMPLETE;
g_dtc_pre_seqinfo_cmnd0[0].repeat_block_side =
DTC_REPEAT_BLOCK_DESTINATION;
g_dtc_pre_seqinfo_cmnd0[0].dest_addr_mode = DTC_DES_ADDR_FIXED;
g_dtc_pre_seqinfo_cmnd0[0].refer_index_table_enable =
DTC_REFER_INDEX_TABLE_DISABLE;
g_dtc_pre_seqinfo_cmnd0[0].sequence_end =
DTC_SEQUENCE_TRANSFER_CONTINUE;
/* MRC */
g_dtc_pre_seqinfo_cmnd0[0].disp_add_enable =
DTC_SRC_ADDR_DISP_ADD_DISABLE;
/* SAR */
g_dtc_pre_seqinfo_cmnd0[0].source_addr = (uint32_t)&dtc_fcrh_data[0];
/* DAR */
g dtc pre seqinfo cmnd0[0].dest addr = (uint32 t)&SCI10.FCR.BYTE.H;
/* CRA, CRB */
g dtc pre seqinfo cmnd0[0].transfer count = 1;

/* [2nd] Sequence transfer information -
transfers the received data from SCI10.FRDR to RAM */
/* MRA */
g_dtc_pre_seqinfo_cmnd0[1].transfer_mode = DTC_TRANSFER_MODE_BLOCK;
g_dtc_pre_seqinfo_cmnd0[1].data_size = DTC_DATA_SIZE_WORD;
g_dtc_pre_seqinfo_cmnd0[1].src_addr_mode = DTC_SRC_ADDR_FIXED;
g_dtc_pre_seqinfo_cmnd0[1].writeback_disable = DTC_WRITEBACK_DISABLE;
/* MRB */
g_dtc_pre_seqinfo_cmnd0[1].chain_transfer_enable =
DTC_CHAIN_TRANSFER_ENABLE;
g_dtc_pre_seqinfo_cmnd0[1].chain_transfer_mode =
DTC_CHAIN_TRANSFER_CONTINUOUSLY;
g_dtc_pre_seqinfo_cmnd0[1].response_interrupt =
DTC_INTERRUPT_AFTER_ALL_COMPLETE;
g_dtc_pre_seqinfo_cmnd0[1].repeat_block_side =
DTC_REPEAT_BLOCK_DESTINATION;
g_dtc_pre_seqinfo_cmnd0[1].dest_addr_mode = DTC_DES_ADDR_INCR;
g_dtc_pre_seqinfo_cmnd0[1].refer_index_table_enable =
DTC_REFER_INDEX_TABLE_DISABLE;
g_dtc_pre_seqinfo_cmnd0[1].sequence_end =
DTC_SEQUENCE_TRANSFER_CONTINUE;
/* MRC */
g_dtc_pre_seqinfo_cmnd0[1].disp_add_enable = DTC_SRC_ADDR_DISP_ADD_DISABLE;
/* SAR */
g_dtc_pre_seqinfo_cmnd0[1].source_addr = (uint32_t)&SCI10.FRDR.WORD;
/* DAR */
g_dtc_pre_seqinfo_cmnd0[1].dest_addr = (uint32_t)&g_dtc_rx_buf0[0];
/* CRA, CRB */
g_dtc_pre_seqinfo_cmnd0[1].transfer_count = 1;
```

```
g_dtc_pre_seqinfo_cmnd0[1].block_size = CMND0_RCV_FIFO_TRG;

/* [3rd] Sequence transfer information -
Changing the SCI10 Receive FIFO trigger to 1 */
/* MRA */
g_dtc_pre_seqinfo_cmnd0[2].transfer_mode = DTC_TRANSFER_MODE_NORMAL;
g_dtc_pre_seqinfo_cmnd0[2].data_size = DTC_DATA_SIZE_BYTE;
g_dtc_pre_seqinfo_cmnd0[2].src_addr_mode = DTC_SRC_ADDR_FIXED;
g_dtc_pre_seqinfo_cmnd0[2].writeback_disable = DTC_WRITEBACK_DISABLE;
/* MRB */
g_dtc_pre_seqinfo_cmnd0[2].chain_transfer_enable =
DTC_CHAIN_TRANSFER_DISABLE;
g_dtc_pre_seqinfo_cmnd0[2].chain_transfer_mode =
DTC_CHAIN_TRANSFER_CONTINUOUSLY;
g_dtc_pre_seqinfo_cmnd0[2].response_interrupt =
DTC_INTERRUPT_AFTER_ALL_COMPLETE;
g_dtc_pre_seqinfo_cmnd0[2].repeat_block_side =
DTC_REPEAT_BLOCK_DESTINATION;
g dtc pre seqinfo cmnd0[2].dest addr mode = DTC DES ADDR FIXED;
g dtc pre seqinfo cmnd0[2].refer index table enable=
DTC REFER INDEX TABLE DISABLE;
g dtc pre seqinfo cmnd0[2].sequence end = DTC SEQUENCE TRANSFER END;
/* MRC */
g dtc pre seqinfo cmnd0[2].disp add enable = DTC SRC ADDR DISP ADD DISABLE;
/* SAR */
g dtc pre seqinfo cmnd0[2].source addr = (uint32 t)&g dtc fcrh cmnd;
/* DAR */
g dtc pre seqinfo cmnd0[2].dest addr = (uint32 t)&SCI10.FCR.BYTE.H;
/* CRA, CRB */
g dtc pre seqinfo cmnd0[2].transfer count = 1;
}
```

処理 2:

外部通信デバイスから cmnd≧“04h”を受信した場合、シーケンス転送せずに CPU への割り込み要求を発生させる。

```
#include "platform.h"
#include "r_dtc_rx_if.h"

void main(void)
{
dtc_err_t ret;
dtc_activation_source_t act_source;
uint32_t sequence_transfer_nr;
uint8_t  sequence_no;
uint8_t ien_bk;
uint16_t i;


…
```

```
/* ---- DTC sequence transfer information for Cmnd4-Cmnd255 ---- */
for (i = 4; i < 256; i++)
{
act_source = DTCE_SCI10_RXI10;
sequence_transfer_nr = 0;
sequence_no = i;
ien_bk  = IEN(SCI10,RXI10); /* IEN(x,x)->ICU.IER[z].BIT.IENz;*/
IEN(SCI10,RXI10) = 0;
ret = R_DTC_CreateSeq(act_source,
NULL,
NULL,
sequence_transfer_nr,
sequence_no);
IEN(SCI10,RXI10) = ien_bk;
}

…
```

Special Notes:

R_DTC_CreateSeq()を呼び出す前に、ユーザは割り込み要求許可ビット（IERm.IENj）をクリアし、処理対象の割り込み要求を禁止にする必要があります（割り込み要因は R_DTC_CreateSeq()に渡されます）。

```
ICU.IER[m].BIT.IENj = 0;
```

R_DTC_CreateSeq()の処理終了後に、禁止にした割り込み要求を許可します。

IERm.IENj ビットと割り込み要因の対応については、ユーザーズマニュアル ハードウェア編の割り込みコントローラ（ICU）章の「割り込みベクタテーブル」をご覧ください。

R_DTC_Control()

この関数は DTC の動作を制御します。

Format

```
dtc_err_t       R_DTC_Control (
                dtc_command_t       command,
                dtc_stat_t          * p_stat,
                dtc_cmd_arg_t       * p_args
)
```

Parameters

dtc_command_t command
DTC の制御コマンド。

*dtc_stat_t * p_stat*
コマンドが DTC_CMD_STATUS_GET の場合、ステータスのポインタ。

dtc_stat_t 構造体のメンバ

メンバ	内容	設定値	説明
vect_nr	DTC 起動ベクタ番号	ベクタ番号監視	この数値は DTC 転送が処理中のときにのみ有効です（DTC アクティブフラグ= 1）
in_progress	DTC アクティブフラグ	- false - true	- DTC 転送動作中ではない - DTC 転送動作中

*dtc_cmd_arg_t * p_args*
コマンドが *DTC_CMD_ACT_SRC_ENABLE*、DTC_CMD_ACT_SRC_DISABLE、DTC_CMD_CHAIN_TRANSFER_ABORT、DTC_CMD_SEQUENCE_TRANSFER_ENABLE、または DTC_CMD_CHANGING_DATA_FORCIBLY_SET の場合、引数の構造体のポインタ。

dtc_cmd_arg_t 構造体のメンバ

メンバ	内容	説明
act_src	DTC 起動ベクタ番号	この数値はコマンドが DTC_CMD_ACT_SRC_ENABLE または DTC_CMD_ACT_SRC_DISABLE または DTC_CMD_SEQUENCE_TRANSFER_ENABLE または DTC_CMD_CHANGING_DATA_FORCIBLY_SET の場合のみ有効
chain_transfer_nr	チェーン転送数（注 1）	この数値はコマンドが DTC_CMD_CHAIN_TRANSFER_ABORT または DTC_CMD_CHANGING_DATA_FORCIBLY_SET の場合のみ有効
*p_transfer_data	RAM の転送情報領域の開始アドレスへのポインタ	この数値はコマンドが DTC_CMD_CHANGING_DATA_FORCIBLY_SET の場合のみ有効
*p_data_cfg	転送情報設定へのポインタ	この数値はコマンドが DTC_CMD_CHANGING_DATA_FORCIBLY_SET の場合のみ有効

注1. ユーザが R_DTC_Create()を呼び出した時の引数“chain_transfer_nr”と同じ値を設定してください。

Return Values

[DTC_SUCCESS]	*/* 正常終了 */*
[DTC_ERR_NOT_OPEN]	*/* DTC は未初期化状態です。*/*
[DTC_ERR_INVALID_COMMAND]	*/* コマンドのパラメータが無効です。もしくは、DTC_CMD_CHANGING_DATA_FORCIBLY_SET コマンドのエラー。 */*
[DTC_ERR_NULL_PTR]	*/* 引数のポインタが NULL です。*/*
[DTC_ERR_ACT]	*/* データ転送実行中 */*

Properties

ファイル r_dtc_rx_if.h にプロトタイプ宣言されています。

Description

コマンドより異なる処理を実行します。

コマンド	引数 dtc_stat_t *	引数 dtc_cmd_arg_t *	説明
DTC_CMD_DTC_START	NULL	NULL	DTC モジュール起動ビット（DTCST）を使って DTC モジュールを起動します。
DTC_CMD_DTC_STOP	NULL	NULL	DTC モジュール起動ビット（DTCST）を使って DTC モジュールを停止します。
DTC_CMD_DATA_READ_SKIP_ENABLE	NULL	NULL	DTC 転送情報リードスキップ許可ビット（RRS）を使って、転送情報リードスキップを許可します。
DTC_CMD_DATA_READ_SKIP_DISABLE	NULL	NULL	DTC 転送情報リードスキップ許可ビット（RRS）を使って、転送情報リードスキップを禁止します。
DTC_CMD_ACT_SRC_ENABLE	NULL	p_args->act_src	DTC 起動許可ビット（DTCE）を 1 使って、DTC 起動要因をセットします。
DTC_CMD_ACT_SRC_DISABLE	NULL	p_args->act_src	DTC 起動許可ビット（DTCE）を使って、DTC 起動要因をクリアします。
DTC_CMD_STATUS_GET	p_stat->in_progress p_stat->vect_nr	NULL	DTC ステータスレジスタ（DTCSTS）を使って DTC のアクティブフラグ（ACT）とデータ転送実行中のベクタ番号（VECN[7:0]）を取得します。
DTC_CMD_CHAIN_TRANSFER_ABORT	NULL	p_args->chain_transfer_nr	処理中のチェーン転送を中止します。
DTC_CMD_SEQUENCE_TRANSFER_ENABLE	NULL	p_args->act_src	DTC シーケンス転送許可レジスタ（DTCSEQ）を使って、シーケンス転送ベクタ番号指定とシーケンス転送を許可します。
DTC_CMD_SEQUENCE_TRANSFER_DISABLE	NULL	NULL	DTC シーケンス転送許可レジスタ（DTCSEQ）を使って、シーケンス転送を禁止します。
DTC_CMD_SEQUENCE_TRANSFER_ABORT	NULL	NULL	シーケンス転送終了ビット（SQTFRL）を使って、シーケンス転送を強制的に終了します。
DTC_CMD_CHANGING_DATA_FORCIBLY_SET	NULL	p_args->act_src p_args->chain_transfer_nr p_args->p_transfer_data p_args->p_data_cfg	R_DTC_Create()によって設定された値を変更します。 R_DTC_Create()によって強制的に設定されたパラメータ（注 1）を変更するのに有効な処理です。

注 1：writeback_disable、sequence_end、refer_index_table_enable、および disp_add_enable

Example

処理 1: DTC モジュールを起動する。

```
dtc_err_t ret;

/* Start DTC module */
ret = R_DTC_Control(DTC_CMD_DTC_START, NULL, NULL);
```

処理 2 : DTC モジュールを停止する。

```
dtc_err_t ret;

/* Stop DTC module */
ret = R_DTC_Control(DTC_CMD_DTC_STOP, NULL, NULL);
```

処理 3 : 転送情報リードスキップを許可する。

```
dtc_err_t ret;

/* Enable transfer information read skip */
ret = R_DTC_Control(DTC_CMD_DATA_READ_SKIP_ENABLE, NULL, NULL);
```

処理 4 : 転送情報リードスキップを禁止する。

```
dtc_err_t ret;

/* Disable transfer information read skip */
ret = R_DTC_Control(DTC_CMD_DATA_READ_SKIP_DISABLE, NULL, NULL);
```

処理 5 : DTCE を使用し、DTC 起動要因をセットする。

```
dtc_err_t ret;
dtc_cmd_arg_t args;

/* Disable DTC transfer request by SCI10 receive data full interrupt */
IEN(SCI10, RXI10) = 0;

/* Set SCI10 receive data full interrupt as DTC activation source*/
args.act_src = DTCE_SCI10_RXI10;

/* Set the interrupt used for DTC activation source */
ret = R_DTC_Control(DTC_CMD_ACT_SRC_ENABLE, NULL, &args);
```

処理 6 : DTCE を使用し、DTC 起動要因をクリアする。

```
dtc_err_t ret;
dtc_cmd_arg_t args;

/* Disable DTC trasnfer request by SCI10 receive data full interrupt */
IEN(SCI10, RXI10) = 0;

/* Set SCI10 receive data full interrupt as DTC activation source */
args.act_src = DTCE_SCI10_RXI10;

/* Delete the interrupt used for DTC activation source */
ret = R_DTC_Control(DTC_CMD_ACT_SRC_DISABLE, NULL, &args);
```

処理 7：DTC のアクティブフラグ（ACT）とデータ転送実行中のベクタ番号（VECN[7:0]）を取得する。

```
dtc_err_t ret;
dtc_stat_t stat;
uint8_t interrupt_number;

/* Get DTC Active Flag (ACT) and Vector number(VECN[7:0])in progress */
ret = R_DTC_Control(DTC_CMD_STATUS_GET, stat, NULL);

if (true == stat.in_progress)
{
/* Vector number is valid */
interrupt_number = stat.vect_nr;
}
else
{
  /* Vector number is inbalid */
}
```

処理 8：処理中のチェーン転送を中止する。

```
dtc_err_t ret;
dtc_cmd_arg_t args;

/* No. Of chain transfer = 5 */
args. chain_transfer_nr = 5;

/* Abort the chain transfer in process */
ret = R_DTC_Control(DTC_CMD_STATUS_GET, NULL, &args);
```

処理 9：シーケンス転送を許可する。

```
dtc err t ret;
dtc cmd arg t args;

/* Set SCI10 receive data full interrupt as sequence transfger activation
source */
args.act src = DTCE SCI10 RXI10;

/* Enable sequence transfer */
ret = R_DTC_Control(DTC_CMD_SEQUENCE_TRANSFER_ENABLE, NULL, &args);
```

処理 10：シーケンス転送を禁止する。

```
dtc_err_t ret;

/* Disable sequence transfer */
ret = R_DTC_Control(DTC_CMD_SEQUENCE_TRANSFER_DISABLE, NULL, NULL);
```

処理 11：シーケンス転送を強制的に終了する。

```
dtc_err_t ret;

/* Disable DTC transfer request by SCI10 receive data full interrupt */
IEN(SCI10, RXI10) = 0;

/* Issue command repeatedly until sequence transfer can be aborted */
do
{
ret = R_DTC_Control(DTC_CMD_SEQUENCE_TRANSFER_ABORT, NULL, NULL);
} while (DTC_ERR_ACT == ret);
```

処理 12：R_DTC_Create()によって設定された値を変更する。

```
dtc_activation_source_t act_source;
uint32_t chain_transfer_nr;

act_source = DTCE_SCI10_RXI10;
chain_transfer_nr = 0;
if (R_DTC_Create(act_source,
                    &g_dtc_info_sqnum,
                    &g_dtc_pre_info_sqnum,
                    chain_transfer_nr) != DTC_SUCCESS)
{
    /* Error */
}

g_dtc_pre_info_sqnum.writeback_disable       = DTC_WRITEBACK_DISABLE;
g_dtc_pre_info_sqnum.sequence_end             =
DTC_SEQUENCE_TRANSFER_CONTINUE;
g_dtc_pre_info_sqnum.refer_index_table_enable = DTC_REFER_INDEX_TABLE_ENABLE;
g_dtc_pre_info_sqnum.disp_add_enable          = DTC_SRC_ADDR_DISP_ADD_DISABLE;
args.act_src = DTCE_SCI10_RXI10;
args.chain_transfer_nr = 0;
args.p_transfer_data = &g_dtc_info_sqnum;
args.p_data_cfg = &g_dtc_pre_info_sqnum;
if (R_DTC_Control(DTC_CMD_CHANGING_DATA_FORCIBLY_SET, NULL, &args) !=
DTC_SUCCESS)
{
    /* Error */
}
```

Special Notes:

コマンドが DTC_CMD_GET_STATUS の場合、DTC が処理中（p_stat->in_progress が true）の場合にのみベクタ番号は有効です。

コマンドが DTC_CMD_ENABLE_ACT_SRC、DTC_CMD_DISABLE_ACT_SRC もしくは DTC_CMD_SEQUENCE_TRANSFER_ABORT の場合、R_DTC_Control()関数を呼び出す前に、ユーザは割り込み要求許可ビット（IERm.IENj）をクリアし、処理対象の割り込み要求を禁止にする必要があります（割り込み要因は R_DTC_Control()に渡されます）。

```
ICU.IER[m].BIT.IENj = 0;
```

R_DTC_Control()の処理終了後に、禁止にした割り込み要求を許可します。

IERm.IENj ビットと割り込み要因の対応については、ユーザーズマニュアル ハードウェア編の割り込みコントローラ（ICU）章の「割り込みベクタテーブル」をご覧ください。

Abort 処理では元の転送情報は壊れてしまうため、転送中断後に再度チェーン転送情報を作成する必要があります。

DTC_CMD_CHANGING_DATA_FORCIBLY_SET で無効な値を設定しようとした場合、R_DTC_Control()は DTC_ERR_INVALID_COMMAND を返します。

無効な値を検出する前に、R_DTC_Control()は既にいくつかのレジスタを更新した可能性があります。この動作が発生するのは、ユーザが無効な値を指定して FORCIBLY DTC を変更しようとした場合のみです。

R_DTC_GetVersion()

この関数は、本モジュールのバージョン番号を返します。

Format
uint32_t R_DTC_GetVersion (void)

Parameters
なし

Return Values
バージョン番号
最上位の 2 バイトがメジャーバージョン番号、最下位の 2 バイトがマイナーバージョン番号

Properties
ファイル r_dtc_rx_if.h にプロトタイプ宣言されています。

Description
この関数は本モジュールのバージョンを返します。

Example
```
uint32_t version;
version = R_DTC_GetVersion();
```

Special Notes:
なし

4. 端子設定

DTC FIT モジュールはピン設定を使用しません。

5. デモプロジェクト

デモプロジェクトには、FIT モジュールとそのモジュールが依存するモジュール（例：r_bsp）を使用する main()関数が含まれます。本 FIT モジュールには以下のデモプロジェクトが含まれます。

注：BSP_CFG_HEAP_BYTES には、CCRX のプロジェクト（v3.04.00 以前）では 0x1000 を設定し、GCC（v8.3.0.202104 以前）のプロジェクトでは 0x1600 を設定することを推奨します。

5.1 dtc_demo_rskrx231, dtc_demo_rskrx231_gcc

プログラム dtc_demo_rskrx231, dtc_demo_rskrx231_gcc は、リピート転送モードに設定した DTC で AD 変換結果を転送します。プログラムを実行すると、DTC が AD 変換結果を 32 バイトのバッファに順次保存します。

5.2 dtc_demo_rskrx65n_2m, dtc_demo_rskrx65n_2m_gcc

プログラム dtc_demo_rskrx65n_2m, dtc_demo_rskrx65n_2m_gcc は、リピート転送モードに設定した DTC で AD 変換結果を転送します。プログラムを実行すると、DTC が AD 変換結果を 32 バイトのバッファに順次保存します。

5.3 dtc_demo_rskrx130, dtc_demo_rskrx130_gcc

プログラム dtc_demo_rskrx130, dtc_demo_rskrx130_gcc は、リピート転送モードに設定した DTC で AD 変換結果を転送します。プログラムを実行すると、DTC が AD 変換結果を 32 バイトのバッファに順次保存します。

5.4 dtc_demo_rskrx72m, dtc_demo_rskrx72m_gcc

プログラム dtc_demo_rskrx72m, dtc_demo_rskrx72m_gcc は、リピート転送モードに設定した DTC で AD 変換結果を転送します。プログラムを実行すると、DTC が AD 変換結果を 32 バイトのバッファに順次保存します。

5.5 dtc_demo_rskrx671, dtc_demo_rskrx671_gcc

プログラム dtc_demo_rskrx671, dtc_demo_rskrx671_gcc は、リピート転送モードに設定した DTC で AD 変換結果を転送します。プログラムを実行すると、DTC が AD 変換結果を 32 バイトのバッファに順次保存します。

5.6 ワークスペースにデモを追加する

デモプロジェクトは、本アプリケーションノートで提供されるファイルの FITDemos サブディレクトリにあります。ワークスペースにデモプロジェクトを追加するには、「ファイル」 >> 「インポート」を選択し、「インポート」ダイアログから「一般」の「既存プロジェクトをワークスペースへ」を選択して「次へ」ボタンをクリックします。「インポート」ダイアログで「アーカイブ・ファイルの選択」ラジオボタンを選択し、「参照」ボタンをクリックして FITDemos サブディレクトリを開き、使用するデモの zip ファイルを選択して「終了」をクリックします。

5.7 デモのダウンロード方法

デモプロジェクトは、RX Driver Package には同梱されていません。デモプロジェクトを使用する場合は、個別に各 FIT モジュールをダウンロードする必要があります。「スマートブラウザ」の「アプリケーションノート」タブから、本アプリケーションノートを右クリックして「サンプル・コード（ダウンロード）」を選択することにより、ダウンロードできます。

6. 付録

6.1 動作確認環境

DTC FIT モジュールの動作確認環境を以下に示します。

表 6-1 動作確認環境 (Rev.4.51)

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 V.2025-01 IAR Embedded Workbench for Renesas RX 5.10.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.07.00 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 8.3.0.202411 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 5.10.1 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	Rev.4.51
使用ボード	-

表 6-2 動作確認環境 (Rev.4.50)

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 V.2024-07 IAR Embedded Workbench for Renesas RX 5.10.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.06.00 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 8.3.0.202405 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 5.10.1 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	Rev.4.50
使用ボード	Evaluation Kit for RX261（型名：RTK5EK2610S00011BJ）

表 6-3 動作確認環境 (Rev.4.40)

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 V.2023-04 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 Package for RX Family V3.05.00 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 8.3.0.202204 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.4.40
使用ボード	Renesas Solution Starter Kit for RX23E-B（型名：RTK0ES1001C00001BJ）

表 6-4 動作確認環境 (Rev.4.30)

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 V.2022-10 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 Package for RX Family V3.05.00 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 8.3.0.202204 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.4.30
使用ボード	Renesas Flexible Motor Control Kit for RX26T(型名：RTK0EMXE70S00020BJ)

表 6-5 動作確認環境 (Rev.4.21)

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 V.2022-10 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 Package for RX Family V3.04.00 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 8.3.0.202202 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.4.21

表 6-6 動作確認環境 (Rev.4.20)

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 V.2022-07 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 Package for RX Family V3.04.00 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 8.3.0.202104 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.4.20
使用ボード	Renesas Starter Kit+ for RX65N-2MB (型名：RTK50565N2CxxxxxBR) Renesas Starter Kit+ for RX72M (型名：RTK5572MNDCxxxxxBJ) Renesas Starter Kit for RX130-512KB (型名：RTK5051308CxxxxxBR) Renesas Starter Kit for RX231 (型名：R0K505231SxxxBE) Renesas Starter Kit+ for RX671 (型名：RTK55671EDCxxxxxBJ)

表 6-7 動作確認環境 (Rev.4.10)

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 V.2022-04 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 Package for RX Family V3.04.00 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 8.3.0.202104 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.4.10
使用ボード	Renesas Starter Kit for RX660（型名：RTK556609HCxxxxxBJ）

表 6-8 動作確認環境 (Rev.4.00)

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 V.2022-01 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 Package for RX Family V3.04.00 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 8.3.0.202104 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.4.00
使用ボード	Target board for RX140（型名：RTK55140xxxxxxxxxx） Renesas Starter Kit for RX66T（型名：RTK50566T0SxxxxxBE）

表 6-9 動作確認環境 (Rev.3.90)

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 V.2021-07 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 Package for RX Family V3.03.00 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 8.3.0.202004 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.3.90
使用ボード	Renesas Starter Kit+ for RX671 （型名：RTK5055671xxxxxxxxx）

表 6-10 動作確認環境 (Rev.3.80)

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 V.2021-07 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 Package for RX Family V3.03.00 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 8.3.0.202004 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.3.80
使用ボード	Target board for RX140 （型名：RTK5RX140xxxxxxxxx）

表 6-11 動作確認環境 (Rev.3.70)

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 V.2021-07 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 Package for RX Family V3.03.00 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 8.3.0.202004 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.3.70
使用ボード	Renesas Starter Kit+ for RX671 （型名：RTK5055671xxxxxxxxx）

表 6-12 動作確認環境 (Rev.3.60)

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 V.7.8.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.02.00 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 8.3.0.201904 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
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	Rev.3.60
使用ボード	Renesas Starter Kit+ for RX72M （型名：RTK5572Mxxxxxxxxxx） Renesas Starter Kit+ for RX65N-2MB （型名： RTK50565N2CxxxxxBR） Renesas Starter Kit for RX130 （型名：RTK5005130SxxxxxBE） Renesas Starter Kit+ for RX231 （型名：RTK505231xxxxxxxx）

表 6-13 動作確認環境 (Rev.3.50)

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 V.7.7.0 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 Package for RX Family V3.02.00 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 8.3.0.201904 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.12.1 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	Rev.3.50
使用ボード	Renesas Solution Starter Kit+ for RX23E-A (product No.: RTK0ESXBxxxxxxxxxx)

表 6-14 動作確認環境 (Rev.3.40)

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 V.7.7.0 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 Package for RX Family V3.01.00 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 4.8.4.201902 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.12.1 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	Rev.3.40
使用ボード	Renesas Starter Kit+ for RX72N（型名：RTK5572Nxxxxxxxxxx）

表 6-15 動作確認環境 (Rev.3.30)

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 V.7.7.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.01.00 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 4.8.4.201902 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
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	Rev.3.30
使用ボード	RX13T CPU Card（型名：RTK0EMXA10C00000BJ）

表 6-16 動作確認環境 (Rev.3.21)

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 V.7.5.0 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 Package for RX Family V3.01.00 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 4.8.4.201902 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.12.1 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	Rev.3.21
使用ボード	Renesas Solution Starter Kit for RX23W（型名：RTK5523Wxxxxxxxxxx）

表 6-17 動作確認環境 (Rev.3.20)

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 V.7.5.0 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 Package for RX Family V3.01.00 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 4.8.4.201902 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.12.1 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	Rev.3.20
使用ボード	Renesas Starter Kit+ for RX72M（型名：RTK5572Mxxxxxxxxxx）

表 6-18 動作確認環境 (Rev.3.10)

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 V.7.5.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.01.00 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	Rev.3.10
使用ボード	Renesas Solution Starter Kit for RX23W（型名：RTK5523Wxxxxxxxxx）

表 6-19 動作確認環境 (Rev.3.01)

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 V.7.4.0 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 Package for RX Family V3.01.00 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 4.8.4.201803 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.10.1 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	Rev.3.01
使用ボード	Renesas Starter Kit+ for RX65N-2MB（型名：RTK50565Nxxxxxxxxx）

表 6-20 動作確認環境 (Rev.3.00)

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 V.7.4.0 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 Package for RX Family V3.01.00 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 4.8.4.201803 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.10.1 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	Rev.3.00
使用ボード	Renesas Starter Kit+ for RX65N-2MB（型名：RTK50565Nxxxxxxxxx）

表 6-21 動作確認環境 (Rev.2.20)

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 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を使用し、以下のオプションを追加。 -lang = c99
エンディアンの順序	ビッグエンディアン／リトルエンディアン
モジュールのバージョン	Ver.2.20
使用ボード	Renesas Starter Kit for RX72T（型名：RTK5572Txxxxxxxxx）

表 6-22 動作確認環境 (Rev.2.10)

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 V7.0.0
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.3.00.00
	コンパイルオプション：統合開発環境のデフォルト設定を使用し、以下のオプションを追加。 -lang = c99
エンディアンの順序	ビッグエンディアン／リトルエンディアン
モジュールのバージョン	Ver.2.10
使用ボード	Renesas Starter Kit for RX111 （型名：R0K505111SxxxBE） Renesas Starter Kit for RX113 （型名：R0K505113SxxxBE） Renesas Starter Kit for RX130 （型名：RTK5005130SxxxxxBE） Renesas Starter Kit for RX130-512KB （型名：RTK5051308SxxxxxBE） Renesas Starter Kit for RX231 （型名：R0K505231SxxxBE） Renesas Starter Kit for RX23T （型名：RTK500523TSxxxxxBE） Renesas Starter Kit for RX24T （型名：RTK500524TSxxxxxBE） Renesas Starter Kit for RX24U （型名：RTK500524USxxxxxBE） Renesas Starter Kit for RX64M （型名：R0K50564MSxxxBE） Renesas Starter Kit for RX71M （型名：R0K50571MSxxxBE） Renesas Starter Kit for RX65N （型名：RTK500565NSxxxxxBE） Renesas Starter Kit for RX65N-2MB （型名：RTK50565N2SxxxxxBE） Renesas Starter Kit for RX66T（型名：RTK50566T0SxxxxxBE）

表 6-23 動作確認環境 (Rev.2.08)

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 V6.0.0
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.2.07.00
	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
モジュールのバージョン	Ver.2.08
使用ボード	Renesas Starter Kit for RX111 （型名：R0K505111SxxxBE） Renesas Starter Kit for RX113 （型名：R0K505113SxxxBE） Renesas Starter Kit for RX130 （型名：RTK5005130SxxxxxBE） Renesas Starter Kit for RX130-512KB （型名：RTK5051308SxxxxxBE） Renesas Starter Kit for RX231 （型名：R0K505231SxxxBE） Renesas Starter Kit for RX23T （型名：RTK500523TSxxxxxBE） Renesas Starter Kit for RX24T （型名：RTK500524TSxxxxxBE） Renesas Starter Kit for RX24U （型名：RTK500524USxxxxxBE） Renesas Starter Kit for RX64M （型名：R0K50564MSxxxBE） Renesas Starter Kit for RX71M （型名：R0K50571MSxxxBE） Renesas Starter Kit for RX65N （型名：RTK500565NSxxxxxBE） Renesas Starter Kit for RX65N-2MB （型名：RTK50565N2SxxxxxBE）

6.2 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 e2 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_dtc_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。

7. 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。）

ユーザーズマニュアル：開発環境

（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

本モジュールは該当する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
2.02	2015.04.01	－	初版発行
2.03	2015.06.15	1 10 17 25	対象デバイスに、RX230 と RX231 を追加 1.2.2 動作環境とメモリサイズ　(5)RX231 の場合を追加。 3.2 R_DTC_Close() Description において、“DMAC のすべてのチャネルのロックが解除される場合”を“DMAC のすべてのチャネルのロックが解除されていた場合”に変更 3.3 R_DTC_Create() Example　処理 3: 複数要因の登録を行う場合を追加
2.04	2015.12.29	1 3 13 14 15 19 23 23 24 25 25 29	対象デバイスに、RX130、RX23T、RX24T を追加 1. 概要 以下の内容を変更した。 「DTC は割り込み要因の　～　を作成する必要があります。」 2.6 コンパイル時の設定　#define DTC_CFG_SHORT_ADDRESS_MODE 元は”ADDRRESS”であった。 2.7 引数 「/* Short-address mode */」、「/* Full-address mode */」を累加 2.9 FIT モジュールの追加方法　を更新した。 3.3 R_DTC_Create()　Parameters #if (1 == DTC_CFG_SHORT_ADDRESS_MODE) 元は”ADDRRESS”であった。 3.3 R_DTC_Create()　Example　処理 1 「uint8_t　ien_bk;」を追加 「des_addr」　元は「dest_addr」であった。 3.3 R_DTC_Create()　Example　処理 2 「uint32_t transfer_data[8]」　元は「uint32 transfer_data[8]」であった。 「uint8_t　ien_bk;」を追加 3.3 R_DTC_Create()　Example　処理 2 「des_addr」　元は「dest_addr」であった。（2 箇所） 3.3 R_DTC_Create()　Example　処理 3 「uint8_t　ien_bk;」を追加 「des_addr」　元は「dest_addr」であった。 3.3 R_DTC_Create()　Example　処理 3 「des_addr」　元は「dest_addr」であった。 3.4 R_DTC_Control()　Example 「uint8_t　interrupt_number;」を追加
2.05	2016.09.30	1 3-4 5 11 13 14 15 15-16	対象デバイスに、RX65N を追加 1. 概要　シーケンス転送の内容を追加 1.2.1 API の概要 表 1.1　「R_DTC_CreateSeq()関数」を追加 1.2.2 動作環境とメモリサイズ　(6)RX65N の場合を追加 2.1 ハードウェアの要求　「DTCb」を追加 2.6 コンパイル事の設定 表　「#define DTC_CFG_USE_SEQUENCE_TRANSFER」を追加 2.7 引数　「r_dtc_rx_target_if.h」を追加 2.7.1 r_dtc_rx_if.h、2.7.2 r_dtc_rx_target_if.h 元は 2.7 引数の内容であった。

		15	2.7.1 r_dtc_rx_if.h　構造体 dtc_command_t に以下を追加 DTC_CMD_SEQUENCE_TRANSFER_ENABLE DTC_CMD_SEQUENCE_TRANSFER_DISABLE DTC_CMD_SEQUENCE_TRANSFER_ABORT
		17 18 22 24 30 - 35 36 37 38 -40	2.8 戻り値　「DTC_ERR_ACT」を追加した 3.1 R_DTC_Open()　Description　DTC インデックステーブルの内容を追加 3.3 R_DTC_Create()　データ構造体 dtc_transfer_data_cfg_t に DTCb の内容を追加 3.3 R_DTC_Create()　以下のデータ構造体を追加 dtc_write_back_t、dtc_sequence_end_t、dtc_refer_index_table_t、dtc_disp_add_t 3.4 R_DTC_CreateSeq()　新規追加 3.5 R_DTC_Control() Return Values　DTC_ERR_ACT を追加 3.5 R_DTC_Control() Description　表を追加 3.5 R_DTC_Control() Example　内容を見直した
2.06	2017.01.31	11 21 - 22 30	1.2.2 動作環境とメモリサイズ　「(6)RX65N の場合」の表 1-12 と表 1-13 の情報を更新した。 3.3 R_DTC_Create() Parameters 説明を追加した。 3.4 R_DTC_CreateSeq() Parameters 説明を追加した。
2.07	2017.03.31	- 1 5 6 38	下記の章番号を変更した。 2.3 動作確認環境、2.8 コードサイズ、4.1 動作確認環境詳細：元は 1.2.2 章であった。 対象デバイスに、RX24U を追加 1.3 DTC IP バージョン　新規追加 1.4 関連アプリケーションノート　内容を見直した 4. 付録　新規追加
2.08	2017.07.31	– 1 7 32 - 36	下記の章のタイトルを変更した。 1.1 DTC FIT モジュールとは：元は 1.1 DTC FIT モジュールであった。 下記の章の本文を移動した。 1.2 DTC FIT モジュールの概要：元は 1. 概要であった。 下記の章番号を変更した。 5.1 動作確認環境：元は 2.3 動作確認環境であった。 5. 付録：元は 4. 付録であった。 6. 参考ドキュメント：元は 5.参考ドキュメントであった。 下記の章を追加した。 2.4 使用する割り込みベクタ。 2.12 FIT モジュールの追加方法。 4. 端子設定 5.2 トラブルシューティング 対象デバイスに、RX651 を追加 2.2 ソフトウェアの要求　「r_cgc_rx」を削除。 3.5 R_DTC_Control()　新規コマンド 「DTC_CMD_CHANGING_DATA_FORCIBLY_SET」を追加
2.10	2018.09.28	1、5 8 40	RX66T のサポートを追加。 RX66T に対応するコードサイズを追加。 「5.1 動作確認環境」：Rev.2.10 に対応する表を追加。
2.11	2018.11.16	40	Renesas Starter Kit for RX66T 製品番号変更

2.20	2019.02.01	－ 1	RX72T グループのサポートを追加。 対象デバイスに RX72T グループを追加。
		5 9 16-39 38 46 47 47	「DTC IP バージョン」セクションに RX72T グループを追加。 RX72T に対応するコードサイズを追加。 各 API 関数で「Reentrant」の説明を削除。 R_DTC_Control()関数の「Special Notes」を更新。 「5. デモプロジェクト」を追加。 Renesas Starter Kit+ for RX66T の型名を変更。。 「6.1 動作確認環境」Rev.2.20 に対応する表を追加。
3.00	2019.05.20	— 1 6 9 41 44	以下のコンパイラをサポート。 - GCC for Renesas RX - IAR C/C++ Compiler for Renesas RX 「ターゲットコンパイラ」のセクションを追加。 関連ドキュメントを削除。 「2.2 ソフトウェアの要求」r_bsp v5.20 以上が必要 「2.8 コードサイズ」セクションを更新。 表 5.1「動作確認環境」： Rev.3.00 に対応する表を追加。 「Web サイトおよびサポート」のセクションを削除。
3.01	2019.06.18	—	DTC_BIG_ENDIAN マクロ定義から "defined（__ BIG_ENDIAN__）" を削除しました。
3.10	2019.06.28	1、5 9 41 42	RX23W のサポートを追加。 RX23W に対応するコードサイズを追加。 「5. デモプロジェクト」を追加。 「6.1 動作確認環境」： Rev.3.10 に対応する表を追加。
3.20	2019.08.15	1、5 10 42 プログラム	RX72M のサポートを追加。 RX72M に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.3.20 に対応する表を追加。 表 6.2：RX23W ボード名変更。 RX72M のサポートを追加。
3.21	2019.11.12	42 プログラム	「6.1 動作確認環境」： Rev.3.21 に対応する表を追加。 RX23W の場合、r_dtc_rx_target.c からマクロ定義の一部を削除します。
3.30	2019.11.25	1、5 6 11 43 プログラム	RX13T のサポートを追加。 2.3 制限事項 制限事項を追加。 RX13T に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.3.30 に対応する表を追加。 RX13T のサポートを追加。 API 関数のコメントを Doxygen スタイルに変更。
3.40	2019.12.30	1、5 11	RX66N、RX72N のサポートを追加。 RX66N、RX72N に対応するコードサイズを追加。

		44	「6.1 動作確認環境」： Rev.3.40 に対応する表を追加。
		プログラム	RX66N、RX72N のサポートを追加。
3.50	2020.03.31	1、7 11 45 プログラム	RX23E-A のサポートを追加。 RX23E-A に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.3.50 に対応する表を追加。 RX23E-A のサポートを追加。
3.60	2020.06.30	47 48 プログラム	デモプロジェクトの更新と追加 「5. デモプロジェクト」に RSKRX72M を追加。 「6.1 動作確認環境」： Rev.3.60 に対応する表を追加。 デモプロジェクトの更新と追加
3.70	2021.03.31	1、7 6 13 49 プログラム	RX671 のサポートを追加。 「1.3 DTC FIT モジュールを使用する」のセクションを追加。 「1.3.DTC FIT モジュールを C++プロジェクト内で使用する」のセクションを追加。 RX671 に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.3.70 に対応する表を追加。 RX671 のサポートを追加。
3.80	2021.04.15	1、7 14 49 プログラム	RX140 のサポートを追加。 RX140 に対応するコードサイズを追加。 「6.1 動作確認環境」：Rev.3.80 に対応する表を追加。 RX140 のサポートを追加。 デモプロジェクトに CS+ のサポートを追加。 全てのデモプロジェクトのコンフィグ設定を更新。
3.90	2021.09.13	47 49 プログラム	「5.5 dtc_demo_rskrx671, dtc_demo_rskrx671_gcc」を追加。 「6.1 動作確認環境」： Rev.3.90 に対応する表を追加。 デモプロジェクトの更新と追加。
4.00	2022.03.14	49 プログラム	「6.1 動作確認環境」： Rev.4.00 に対応する表を追加。 RX140 の割り込みベクタ"DTCE_RNG_RNGRDI"を追加。 RX66T-48Pin のサポートを追加。
4.10	2022.03.31	1、7 14 49	RX660 のサポートを追加。 RX660 に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.4.10 に対応する表を追加。

		プログラム	RX660 のサポートを追加。
4.20	2022.06.28	47 49 プログラム	デモのヒープサイズ設定の注意事項を追加。 「6.1 動作確認環境」：。 Rev.4.20 に対応する表を追加。。 デモプロジェクトを更新
4.21	2022.12.27	49 プログラム	「6.1 動作確認環境」：。 Rev.4.21 に対応する表を追加。。 Linux 対応のため、インクルードヘッダファイルパスの形式を更新しました。
4.30	2023.03.31	1, 7 14 49 プログラム	RX26T のサポートを追加。 RX26T に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.4.30 に対応する表を追加。 RX26T のサポートを追加。
4.40	2023.05.29	1, 7 15 18 50 プログラム	RX23E-B のサポートを追加。 RX23E-B に対応するコードサイズを追加。 「2.13 FIT モジュールの追加方法」から FIT configurator の記述を削除 「6.1 動作確認環境」： Rev.4.40 に対応する表を追加。 RX23E-B のサポートを追加。 FIT Configurator の説明を削除しました。
4.50	2024.06.28	1, 7 15 51 プログラム	RX260, RX261 のサポートを追加。 RX260, RX261 に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.4.50 に対応する表を追加。 RX260, RX261 のサポートを追加。
4.51	2025.03.15	51 プログラム	「6.1 動作確認環境」： Rev.4.51 に対応する表を追加。 FIT モジュールの免責事項と著作権を更新。

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

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。

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/

© 2025 Renesas Electronics Corporation. All rights reserved.