RX ファミリ

DAC 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 D/A コンバータ（DAC）モジュールについて説明します。本モジュールは、D/A コンバータを使用して D/A コンバータ周辺ドライバの動作を制御します。以降、本モジュールを DAC FIT モジュールと称します。

対象デバイス

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

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 DAC FIT モジュールとは 4
1.2 DAC FIT モジュールの概要 4
1.3 DAC FIT モジュールを使用する 4
1.3.1 DAC FIT モジュールを C++プロジェクト内で使用する 4
1.4 API の概要 4

2. API 情報 5
2.1 ハードウェアの要求 5
2.2 ソフトウェアの要求 5
2.3 制限事項 5
2.3.1 RAM の配置に関する制限事項 5
2.4 サポートされているツールチェーン 5
2.5 使用する割り込みベクタ 5
2.6 ヘッダファイル 5
2.7 整数型 5
2.8 コンパイル時の設定 6
2.9 コードサイズ 7
2.10 引数 13
2.11 戻り値 13
2.12 コールバック関数 13
2.13 FIT モジュールの追加方法 13
2.14 for 文、while 文、do while 文について 15

3. API 関数 16
R_DAC_Open() 16
R_DAC_Close() 19
R_DAC_Write() 20
R_DAC_Control() 21
R_DAC_GetVersion() 23

4. 端子設定 24

5. デモプロジェクト 25
5.1 dac_demo_rskrx113, dac_demo_rskrx113_gcc 25
5.2 dac_demo_rskrx231, dac_demo_rskrx231_gcc 26
5.3 dac_demo_rskrx64m, dac_demo_rskrx64m_gcc 27
5.4 dac_demo_rskrx71m, dac_demo_rskrx71m_gcc 28
5.5 dac_demo_rskrx65n, dac_demo_rskrx65n_gcc 29
5.6 dac_demo_rskrx65n_2m, dac_demo_rskrx65n_2m_gcc 29
5.7 dac_demo_rskrx72m, dac_demo_rskrx72m_gcc 31
5.8 ワークスペースにデモを追加する 31
5.9 デモのダウンロード方法 32

6. 付録 33
6.1 動作確認環境 33

6.2 トラブルシューティング 44

7. 参考ドキュメント 45

テクニカルアップデートの対応について 45

改訂記録 46

1. 概要

1.1 DAC 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 DAC FIT モジュールの概要

DAC FIT モジュールは、RX111、RX113、RX130、RX13T、RX140、RX230、RX231、RX23T、RX23E-B、RX24T、RX24U、RX260、RX261、RX26T、RX64M、RX651、RX65N、RX66T、RX66N、RX660、RX71M、RX72T、RX72M、RX72N グループの D/A コンバータ周辺機能をサポートします。D/A コンバータ周辺機能の詳細は、ご使用の MCU のユーザーズマニュアル: ハードウェアの「D/A コンバータ（DA）」の章をご覧ください。

アナログに変換するデータは左揃え、または右揃えで配置でき、チャネルは個別に出力できます。各 MCU でサポートしているハードウェア機能にも対応しています。以下に機能の例を示します。基準電圧の選択、A/D コンバータ周辺機能との同期変換、出力禁止の場合は、変換を無効にする、負荷が大きい場合は、内蔵アンプを有効にする。

1.3 DAC FIT モジュールを使用する

1.3.1 DAC 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 DAC 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_dac_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 1.1API 関数一覧

関数	説明
R_DAC_Open()	D/A コンバータを起動し、関連するレジスタを初期化して、MCU ごとで選択可能なオプションを設定します。
R_DAC_Close()	D/A コンバータを停止します。
R_DAC_Write()	変換動作のために、チャネルに対応するレジスタにデータを書き込みます。
R_DAC_Control()	チャネルの出力を有効または無効します。内蔵アンプを有効または無効にします（RX64M、RX71M）。
R_DAC_GetVersion()	本モジュールのバージョン番号を返します。

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。

— DAC
— GPIO

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 FIT モジュールは「6.1 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。

2.5 使用する割り込みベクタ

なし

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_dac_rx_if.h に記載しています。

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_dac_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。

コンフィギュレーションオプション（r_dac_rx_config.h）	
#define DAC_CFG_PARAM_CHECKING_ENABLE ※デフォルト値は“1”	に設定した場合、ビルド時にパラメータチェック処理をコードに含めます。0 に設定した場合、ビルド時にパラメータチェック処理をコードから省略し、コードサイズを小さくすることができます。BSP_CFG_PARAM_CHECKING_ENABLE に設定した場合、システムのデフォルト設定を使用します。

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 およびスタックのコードサイズ

デバイス	分類	使用メモリ ルネサス製コンパイラ		備考
		パラメータチェック処理あり	パラメータチェック処理なし	
RX130	ROM	319 バイト	285 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	20 バイト		R_DAC_Open 関数使用時
RX13T	ROM	284 バイト	257 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	60 バイト		R_DAC_Open 関数使用時
RX231	ROM	338 バイト	289 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	20 バイト		R_DAC_Open 関数使用時
RX23W	ROM	329 バイト	286 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	32 バイト		R_DAC_Open 関数使用時
RX65N	ROM	403 バイト	359 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	20 バイト		R_DAC_Open 関数使用時
RX66T	ROM	405 バイト	373 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	20 バイト		R_DAC_Open 関数使用時
RX66N	ROM	551 バイト	497 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	8 バイト		R_DAC_Open 関数使用時

ROM, RAM およびスタックのコードサイズ				
デバイス	分類	使用メモリ ルネサス製コンパイラ		備考
		パラメータチェック処理あり	パラメータチェック処理なし	
RX72T	ROM	405 バイト	373 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	20 バイト		R_DAC_Open 関数使用時
RX72M	ROM	407 バイト	365 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	32 バイト		R_DAC_Open 関数使用時
RX72N	ROM	551 バイト	510 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	8 バイト		R_DAC_Open 関数使用時
RX140	ROM	360 バイト	329 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	8 バイト		R_DAC_Open 関数使用時
RX660	ROM	449 バイト	418 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	8 バイト		R_DAC_Open 関数使用時
RX26T	ROM	456 バイト	425 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	12 バイト		R_DAC_Open 関数使用時
RX23E-B	ROM	451 バイト	427 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	8 バイト		R_DAC_Open 関数使用時
RX260	ROM	358 バイト	327 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	8 バイト		R_DAC_Open 関数使用時
RX261	ROM	358 バイト	327 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	8 バイト		R_DAC_Open 関数使用時

ROM、RAM およびスタックのコードサイズ				
デバイス	カテゴリ	使用メモリ		備考
		GCC		
		パラメータチェック処理あり	パラメータチェック処理なし	
RX130	ROM	576 バイト	536 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX13T	ROM	440 バイト	400 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX231	ROM	616 バイト	552 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX65N	ROM	752 バイト	688 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX66T	ROM	776 バイト	728 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX66N	ROM	1048 バイト	975 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX72T	ROM	776 バイト	728 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX72M	ROM	746 バイト	704 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX72N	ROM	1048 バイト	975 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		

ROM、RAM およびスタックのコードサイズ				
デバイス	カテゴリ	使用メモリ		備考
		GCC		
		パラメータチェック処理あり	パラメータチェック処理なし	
RX140	ROM	640 バイト	600 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX660	ROM	824 バイト	784 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX26T	ROM	624 バイト	584 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX23E-B	ROM	600 バイト	552 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX260	ROM	456 バイト	408 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		
RX261	ROM	448 バイト	400 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	-		

ROM、RAM およびスタックのコードサイズ				
デバイス	カテゴリ	使用メモリ		備考
		IAR コンパイラ		
		パラメータチェック処理あり	パラメータチェック処理なし	
RX130	ROM	900 バイト	868 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	136 バイト		
RX13T	ROM	640 バイト	600 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	56 バイト		
RX231	ROM	932 バイト	876 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	136 バイト		
RX65N	ROM	1014 バイト	970 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	152 バイト		
RX66T	ROM	1034 バイト	1002 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	148 バイト		
RX66N	ROM	977 バイト	941 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	60 バイト		
RX72T	ROM	1038 バイト	1002 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	148 バイト		
RX72M	ROM	998 バイト	958 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	156 バイト		
RX72N	ROM	982 バイト	930 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	60 バイト		

ROM、RAM およびスタックのコードサイズ				
デバイス	カテゴリ	使用メモリ		備考
		IAR コンパイラ		
		パラメータチェック処理あり	パラメータチェック処理なし	
RX140	ROM	728 バイト	696 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	56 バイト		
RX660	ROM	933 バイト	881 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	188 バイト		
RX26T	ROM	921 バイト	889 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	60 バイト		
RX23E-B	ROM	952 バイト	920 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	56 バイト		
RX260	ROM	792 バイト	760 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	56 バイト		
RX261	ROM	800 バイト	768 バイト	
	RAM	0 バイト	0 バイト	
	最大のスタック使用量	56 バイト		

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_dac_rx_if.h に記載されています。

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_dac_rx_if.h で記載されています。

以下に本モジュールの API 関数で使用する戻り値（エラーコード）を示します。戻り値の列挙型は、API 関数の宣言と共に r_dac_rx_if.h に記載されています。

```
/* DAC で使用される API エラーコード */
typedef enum e_dac_err
{
    DAC_SUCCESS=0,
    DAC_ERR_BAD_CHAN,          // 存在しないチャネル番号
    DAC_ERR_INVALID_CMD,       // 無効な動作コマンド
    DAC_ERR_INVALID_ARG,       // パラメータに対して無効な引数です。
    DAC_ERR_NULL_PTR,          // null ptr を受信; 要求された引数がありません。
    DAC_ERR_LOCK_FAILED,       // D/A コンバータのロックに失敗しました（モジュールは既
                                  に起動しています）。
    DAC_ERR_UNLOCK_FAILED      // D/A コンバータのロック解除に失敗しました。
    DAC_ERR_ADC_NOT_POWERED,   // A/D コンバータが起動していないので、同期変換できま
                                  せん。
    DAC_ERR_ADC_CONVERTING,    // A/D コンバータが変換動作中なので、同期変換できません。
    DAC_ERR_BIAS_CURRENT_SOURCE // バイアス電流源を禁止。
} dac_err_t;
```

2.12 コールバック関数

なし

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+で開発中プロジェクトに FIT モジュールを追加
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_DAC_Open()

D/A コンバータを起動し、関連するレジスタを初期化して、MCU ごとで選択可能なオプションを設定します。

Format

```
dac_err_t    R_DAC_Open (
    dac_cfg_t       * p_cfg
)
```

Parameters

*dac_cfg_t *p_cfg*

設定構造体へのポインタ

“p_cfg”で使用される構造体サンプル：

```
typedef struct st_dac_cfg
{
    bool            fmt_flush_right;            // 全 MCU
    bool            sync_with_adc;               // RX113/RX130/RX230/RX231/
// RX24U/RX63N/RX631/RX64M/
// RX65N/RX651/RX71M
    uint8_t         sync_unit;                  // 0 or 1; RX64M/RX71M
                                                // RX65N/RX651
    bool            ch_conv_off_when_output_off; // RX210/RX63N/RX631/RX64M/
 // RX65N/RX651/RX71M
    dac_refv_t      ref_voltage;                // RX113/RX230/RX231
} dac_cfg_t;

typedef enum e_dac_refv  // D/A コンバータ基準電圧
{
    DAC_REFV_AVVC0_AVSS0    = 1,
    DAC_REFV_INTERNAL_AVSS0 = 3,
    DAC_REFV_VREFH_VREFL    = 6
} dac_refv_t;
```

Return Values

```
[DAC_SUCCESS]                /* 成功; D/A コンバータが初期化されました。          */
[DAC_ERR_NULL_PTR]           /* “p_cfg”ポインタが NULL です。                    */
[DAC_ERR_LOCK_FAILED]        /* DAC モジュールのロックに失敗; DAC は既に動作中です。*/
[DAC_ERR_INVALID_ARG]        /* 同期するユニットの番号が無効です。               */
[DAC_ERR_ADC_NOT_POWERED]/* A/D コンバータが起動していないので、同期変換できません*/
[DAC_ERR_ADC_CONVERTING]   /* A/D コンバータが変換動作中なので、同期変換できません。*/
```

Properties

ファイル r_dac_rx_if.h にプロトタイプ宣言されています。

説明

D/A コンバータを起動し、関連するレジスタを初期化して、MCU ごとで選択可能なオプションを設定します。

Example

```
dac_err_t  err;
  dac_cfg_t  config;

  /* RX63N DAC を初期化する */
  config.fmt_flush_right = true;
  config.sync_with_adc = false;
  config.ch_conv_off_when_output_off = true;
  err = R_DAC_Open(&config);
```

Special Notes:

データはアプリケーションによって、左揃え、または右揃えで配置できます。“fmt_flush_right”パラメータで、DAC にデータの配置方法を示します。

“DAC_ERR_ADC_CONVERTING”エラーを回避するためには、A/D コンバータが起動してスキャンを開始する前に、D/A コンバータを起動します。

D/A コンバータの I/O 端子は、本関数を呼び出す前に設定しておく必要があります。

```
    R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_MPC); // ロック解除
#ifdef BSP_MCU_RX113
    /*
     * RX113 グループユーザーズマニュアル:ハードウェアの注記に従って設定。
* 表 19.1「 マルチプル端子の割り当て端子一覧 (10/10)」 下の注 1 :
     *     「この端子機能を使用する場合は、該当端子の設定を汎用入力にしてください
     *     (PORT.PDR.Bm ビットおよび PORT.PMR.Bm ビットを"0"にする)。」
     *
     */
    PORTJ.PDR.BIT.B0 = 0;
    PORTJ.PMR.BIT.B0 = 0;
    PORTJ.PDR.BIT.B2 = 0;
    PORTJ.PMR.BIT.B2 = 0;

  /* PJ0 と PJ2 を D/A コンバータのアナログ出力端子として設定 */
    MPC.PJ0PFS.BIT.ASEL = 1;
    MPC.PJ2PFS.BIT.ASEL = 1;

    /*
     * D/A コンバータの基準電圧に VREFH/VREFL を使用する場合は、以下の 2 行のコメントを
     * 解除します。
     */
    //MPC.P41PFS.BIT.ASEL = 1;    // P41 を VREFH アナログ端子として設定
    //MPC.P42PFS.BIT.ASEL = 1;    // P42 を VREFL アナログ端子として設定

#else  /* RX111, RX210, RX63N */

    /* アナログ出力用の I/O ポート端子を汎用入力端子として設定
    PORT0.PDR.BIT.B3 = 0;
    PORT0.PMR.BIT.B3 = 0;
    PORT0.PDR.BIT.B5 = 0;
    PORT0.PMR.BIT.B5 = 0;

  /* P03 と P05 の端子機能を D/A コンバータのアナログ出力端子として設定 */
    MPC.P03PFS.BIT.ASEL = 1;
    MPC.P05PFS.BIT.ASEL = 1;

#endif

    R_BSP_RegisterProtectEnable(BSP_REG_PROTECT_MPC);  // ロック
```

アンプを使用する時の注意事項

アンプを使用するときは“ch_conv_off_when_output_off”を“true”に設定してください。

A/D コンバータ使用時の注意事項

D/A A/D 同期変換有効 (sync_with_adc = true) に設定していた場合、A/D コンバータ（注 1）をモジュールストップ状態にするのであれば、その前に R_DAC_Close 関数を実行してください。

注 1. RX64M/RX651/RX65N/RX66N/RX71M/RX72M/RX72N の場合はユニット 1、
RX24U/RX26T/RX66T/RX72T の場合はユニット 2 が対象になります。

R_DAC_Close()

この関数は D/A コンバータを停止します。

Format
dac_err_t　　R_DAC_Close (void)

Parameters
なし

Return Values
[DAC_SUCCESS]　　*/* 成功; チャネルの出力を無効にしました。 */*
[DAC_ERR_UNLOCK_FAILED]　　*/* D/A コンバータのロック解除に失敗しました。 */*

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

Description
DAC の出力を禁止して、モジュールストップに移行します。

Example
```
    :
/* D/A コンバータを初期化 */
err = R_DAC_Open(&config);
 :
 :
/* D/A コンバータを終了 */
err = R_DAC_Close();
```

Special Notes:
D/A A/D 同期変換有効 (sync_with_adc = true) に設定していた場合、A/D コンバータ（注 1）をモジュールストップ状態にするのであれば、その前に R_DAC_Close 関数を実行してください。

注 1. RX64M/RX651/RX65N/RX66N/RX71M/RX72M/RX72N の場合はユニット 1、
RX24U/RX26T/RX66T/RX72T の場合はユニット 2 が対象になります。

R_DAC_Write()

この関数は、チャネルに対応するデータレジスタにデータを書き込みます。

Format

```
dac_err_t     R_DAC_Write (
        uint8_t const     chan,
        uint16_t          data
)
```

Parameters

uint8_t const chan
書き込みするチャネル

uint16_t data
書き込むデータ

Return Values

```
[DAC_SUCCESS]          /* 成功; チャネルに対応するレジスタにデータが書き込まれました。*/
[DAC_ERR_BAD_CHAN]     /* 存在しないチャネル番号                                      */
```

Properties

ファイル r_dac_rx_if.h にプロトタイプ宣言されています。

Description

変換に際して、チャネルに対応するレジスタにデータを書き込みます。ご使用の MCU によって、データ長は 8/10/12/16 ビットのいずれかになります。Write()関数にてデータを格納する場合は、データは選択された右詰または左詰のフォーマットで格納されなければなりません。

Example

```
    dac_err_t  err;
    uint16_tg_short;
      :
  :
    /* チャネル 1 に 0V へ変換するためのデータを書き込む。 */
    g_short = 0x0000;
    err = R_DAC_Write(DAC_CH1, g_short);
```

Special Notes:

なし

R_DAC_Control()

この関数はチャネルを有効または無効にします。

Format

```
dac_err_t     R_DAC_Control (
        uint8_t const               chan,
        dac_cmd_t const             cmd
)
```

Parameters

uint8_t const chan
使用するチャネル

dac_cmd_t const cmd
実行するコマンド（以下の列挙型参照）

以下に“*cmd*”の値を示します。

```
typedef enum e_dac_cmd
{
    DAC_CMD_OUTPUT_ON,     // チャネルのアナログ出力が有効にされました。
    DAC_CMD_OUTPUT_OFF,    // チャネルのアナログ出力が無効にされました。
    DAC_CMD_AMP_ON,       // RX64M、RX71M: アンプ、ゲイン 1。ユーザーズマニュアル:
    DAC_CMD_AMP_OFF,      // ハードウェアで「電気的特性」の章をご覧ください。
    DAC_CMD_ASW_ON,      // RX65N/RX66N,RX72M,RX72N: チャネル 0 出力バッファアンプ
                            の安定待ち (端子は Hi-Z)

    DAC_CMD_ASW_OFF,      // チャネル 0 出力バッファアンプの安定待ちを解除 (出力許可)
    DAC_CMD_BUF_ON,       // バッファアンプの出力をプルダウンする。
    DAC_CMD_BUF_OFF,      // バッファアンプの出力をプルダウンしない。
    DAC_CMD_STB_ON,       // D/A 変換停止時のアナログ出力端子は 1kΩプルダウン。
    DAC_CMD_STB_OFF,      // D/A 変換停止時のアナログ出力端子は Hi-Z。
    DAC_CMD_END_ENUM
} dac_cmd_t;
```

Return Values

```
[DAC_SUCCESS]                   /* 成功; チャネルを無効にしました。   */
[DAC_ERR_BAD_CHAN]              /* 存在しないチャネル番号             */
[DAC_ERR_BIAS_CURRENT_SOURCE]   /* バイアス電流源を禁止。             */
[DAC_ERR_INVALID_CMD]           /* 無効なコマンド                     */
```

Properties

ファイル r_dac_rx_if.h にプロトタイプ宣言されています。

Description

OUTPUT コマンドを使って、Write()関数にてデータレジスタに書き込まれた変換データ値を出力します。AMP コマンドを使って、アンプを有効または無効にします。アンプを有効にした後は、出力を許可しておかなければなりません。

Example

```
dac_cfg_t  config;
dac_err_t  err;

/* RX64M, RX71M D/A コンバータを初期化する */
config.fmt_flush_right = true;
config.sync_with_adc = true;
config.sync_unit = 1;
config.ch_conv_off_when_output_off = true;

err = R_DAC_Open(&config);

/* チャネル 0 に 0V に変換するためのデータを書き込む */
err = R_DAC_Write(DAC_CH0, 0x0);

/* 大き目の負荷に対応 */
err = R_DAC_Control(DAC_CH0, DAC_CMD_AMP_ON);
/* 3us 以上の wait を入れてください */

/* 変換データの出力 */
err = R_DAC_Control(DAC_CH0, DAC_CMD_OUTPUT_ON);

/* チャネル 0 に 3.3V に変換するためのデータを書き込む */
err = R_DAC_Write(DAC_CH0, 0x0FFF);
```

Special Notes:

R_DAC_Write(DAC_CHx, 0x0)を実施して出力アンプを作動させること。

出力アンプ使用時（コマンド DAC_CMD_AMP_ON 実行時）はオープン関数にて“ch_conv_off_when_output_off”を“true”に設定してください。設定は無効となります。

アンプを使うときは、下記の手順で行ってください。

1. R_DAC_Control()関数で、DAC_CMD_ASW_ON コマンド実行 *
2. R_DAC_Control()関数で、DAC_CMD_AMP_ON コマンド実行
3. R_DAC_Control()関数で、DAC_CMD_OUTPUT_ON コマンドを実行
4. 3us 以上の時間待つ
5. R_DAC_Control()関数で、DAC_CMD_ASW_OFF コマンド実行 *
6. R_DAC_Write 関数にて D/A 出力値を書き込む

注 *. (RX65N、RX66N、RX72M、RX72N のみ)

DAC_CMD_OUTPUT_ON、および DAC_CMD_OUTPUT_OFF コマンドは、D/A A/D 同期変換を有効（R_DAC_Open 関数の p_cfg.sync_with_adc に true を設定して実行）にしている場合、同期する A/D コンバータ（注 1）が停止している状態で実行してください

注 1. RX64M/RX651/RX65N/RX66N/RX71M/RX72M/RX72N の場合、ユニット 1 を、
RX24U/RX26T/RX66T/RX72T の場合はユニット 2 を停止してください。その他の MCU グループではユニットが 1 つのため、指定はありません。

R_DAC_GetVersion()

この関数は実行時に本モジュールのバージョンを返します。

Format
uint32_t R_DAC_GetVersion (void)

Parameters
なし

Return Values
本モジュールのバージョン

Properties
ファイル r_dac_rx.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。バージョン番号は符号化され、最上位の 2 バイトがメジャーバージョン番号を、最下位の 2 バイトがマイナーバージョン番号を示しています。

Example
```
uint32_t   version;
  :
version = R_DAC_GetVersion();
```

Special Notes:
なし

4. 端子設定

DAC FIT モジュールを使用するには、周辺機能の出力信号をマルチファンクションピンコントローラ（MPC）で持つ端子に割り当てます。本書では、端子の割り当てを「端子設定」と呼びます。
R_DAC_Open 関数を呼び出す前に、端子設定を行なってください。

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。

5.1 dac_demo_rskrx113, dac_demo_rskrx113_gcc

dac_demo_rskrx113, dac_demo_rskrx113_gcc は、RSKRX113、RX113 D/A コンバータ（R12DAA）対応の DAC FIT モジュール（r_dac_rx）のシンプルなデモプロジェクトです。デモでは r_dac_rx API を使用して、DAC の起動、設定、書き込みを行います。LOW、MEDIUM、または HIGH レベルの値が DAC チャネル 1 に書き込まれると、連続的なループに 1 秒間遷移します。LOW レベルの値が書き込まれると LED0（緑）が点灯し、MEDIUM レベルの値が書き込まれると LED1（オレンジ）が、HIGH レベルの値が書き込まれると LED2（赤）が点灯します。RSKRX113 ボード使用時の DAC 出力チャネル信号のアクセスおよび設定、また基準電圧に関する詳細については、以下の「DAC チャネル 0、チャネル 1 出力信号測定時の注意事項」をご覧ください。

セットアップと実行

1. サンプルコードをコンパイルし、ダウンロードします。
2. 必要な場合、マルチメータ用リードかオシロスコープ用プローブを DAC チャネル出力端子に取り付けます。
3. 「Reset Go」をクリックしてソフトウェアを起動します。PC が Main 関数で停止した場合、F8 を押してレジュームしてください。
4. ブレークポイントを設定し、グローバル変数を確認します。

対応ボード

RSKRX113

DAC チャネル 0、チャネル 1 出力信号測定時の注意事項

— DAC チャネル 0（DA0）出力では、MCU の端子 2 にマップする I/O ポート PJ0 を使用します。
RSKRX113 では、DA0 は SW1 と端子 2 を共用しています。DA0 のアナログ出力信号用に端子 2 を使用するには、0 オーム抵抗を R241 から R239 に移動する必要があります。これによって、JA1_13 または J1_2 を介して DA0 にアクセスが可能になります。ただし、リンクの設定をこのように変更した場合、SW1 は使用不可となりますのでご注意ください。

— DAC チャネル 1（DA1）出力では、MCU の端子 100 にマップする I/O ポート PJ2 を使用します。
RSKRX113 では、JA1_14 を介して DA1 にアクセスできます。
DA1 には J4_25 を介してもアクセスが可能です。

— GROUND には JA1_2 を介してアクセスできます（その横の端子 4 もグランド端子です）。

— RX113 は、DAVREFCR レジスタを使用し、3 種類の DAC 基準電圧に対応しています。

1. AVCC0/AVSS0
   RSKRX113 ボードでは、AVCC0 および AVSS0 は、UC_VCC (typ. 3.3V)および GROUND にそれぞれ接続されます。
2. 内部基準電圧/AVSS0
   内部基準電圧は typ. 1.4 V、AVSS0 は GROUND です。
3. VREFH/VREFL
   RSKRX113 ボードでは、VREFH/VREFL は接続されていません。VREFH/VREFL には J4_18 (CON_VREFH)および J4_17 (CON_VREFL)が使用されます。VREFH/VREFL を DAC の基準電圧として使用するには、以下の条件が必要となります。

- I/O 端子 P41 と P42 は、MPC を介してアナログ端子として設定してください。
- VREFH/VREEFL は HIGH/LOW 供給電圧に接続してください。
  これらの信号の設定に使用するオプションリンクの詳細は、「RX113 グループ Renesas Starter Kit ユーザーズマニュアル（R20UT2762EJ0100）」の DAC 設定のセクションをご覧ください。
- 代替オプション
  J4_17 (CON_VREFL)には J3_12 (GROUND)を使用
  J4_18 (CON_VREFH)には J3_10 (UC_VCC, 3.3V)を使用

5.2　dac_demo_rskrx231, dac_demo_rskrx231_gcc

dac_demo_rskrx231, dac_demo_rskrx231_gcc は、RSKRX231、RX231 D/A コンバータ（R12DAA）対応の DAC FIT モジュール（r_dac_rx）のシンプルなデモプロジェクトです。デモでは r_dac_rx API を使用して、DAC の起動、設定、書き込みを行います。LOW、MEDIUM、または HIGH レベルの値が DAC チャネル 1 に書き込まれると、連続的なループに 1 秒間遷移します。LOW レベルの値が書き込まれると LED0（緑）が点灯し、MEDIUM レベルの値が書き込まれると LED1（オレンジ）が、HIGH レベルの値が書き込まれると LED2（赤）が点灯します。RSKRX231 ボード使用時の DAC 出力チャネル信号のアクセスおよび設定、また基準電圧に関する詳細については、以下の「DAC チャネル 0、チャネル 1 出力信号測定時の注意事項」をご覧ください。

セットアップと実行

1. サンプルコードをコンパイルし、ダウンロードします。
2. 必要な場合、マルチメータ用リードかオシロスコープ用プローブを DAC チャネル出力端子に取り付けます。
3. 「Reset Go」をクリックしてソフトウェアを起動します。PC が Main 関数で停止した場合、F8 を押してレジュームしてください。
4. ブレークポイントを設定し、グローバル変数を確認します。

対応ボード

RSKRX231

DAC チャネル 0、チャネル 1 出力信号測定時の注意事項

— DAC チャネル 0（DA0）出力では、MCU の端子 2 にマップする I/O ポート P03 を使用します。DA0 には J1_2 を介してアクセスが可能です。
— DAC チャネル 1（DA1）出力では、MCU の端子 100 にマップする I/O ポート P05 を使用します。DA1 には JA1_14（または J4_25）を介してアクセスが可能です。
— GROUND には JA1_2 を介してアクセスできます（その横の端子 4 もグランド端子です）。
— RX231 は、DAVREFCR レジスタを使用し、3 種類の DAC 基準電圧に対応しています。

1. AVCC0/AVSS0
   RSKRX231 ボードでは、AVCC0 および AVSS0 は、UC_VCC (typ. 3.3V)および GROUND にそれぞれ接続されます。
2. 内部基準電圧/AVSS0
   内部基準電圧は typ. 1.4 V、AVSS0 は GROUND です。
3. VREFH/VREFL
   RSKRX231 ボードでは、VREFH/VREFL は UC_VCC (typ. 3.3V)および GROUND に接続されています。下記の 0 オーム抵抗を移動した後で、CON_VREFH (J1_1)および CON_VREFL (J1_3)を使用して、外部基準電圧に接続できます。

- CON_VREFH: R68 を R67 に移動
- CON_VREFL: R65 を R66 に移動

5.3　dac_demo_rskrx64m, dac_demo_rskrx64m_gcc

dac_demo_rskrx64m, dac_demo_rskrx64m_gcc は、RSKRX64M、RX64M D/A コンバータ（R12DA）対応の DAC FIT モジュール（r_dac_rx）のシンプルなデモプロジェクトです。デモでは r_dac_rx API を使用して、DAC の起動、設定、書き込みを行います。LOW、MEDIUM、または HIGH レベルの値が DAC チャネル 0 に書き込まれると、連続的なループに 1 秒間遷移します。LOW レベルの値が書き込まれると LED1 が点灯し、MEDIUM レベルの値が書き込まれると LED2 が、HIGH レベルの値が書き込まれると LED3 が点灯します。RSKRX64M ボード使用時の DAC 出力チャネル信号のアクセスおよび設定、また基準電圧に関する詳細については、以下の「DAC チャネル 0、チャネル 1 出力信号測定時の注意事項」をご覧ください。

セットアップと実行

1. サンプルコードをコンパイルし、ダウンロードします。
2. 必要な場合、マルチメータ用リードかオシロスコープ用プローブを DAC チャネル出力端子に取り付けます。
3. 「Reset Go」をクリックしてソフトウェアを起動します。PC が Main 関数で停止した場合、F8 を押してレジュームしてください。
4. ブレークポイントを設定し、グローバル変数を確認します。

対応ボード

RSKRX64M

DAC チャネル 0、チャネル 1 出力信号測定時の注意事項

— DAC チャネル 0（DA0）出力では、MCU の端子 4 にマップする I/O ポート P03 を使用します。RSKRX64M では、DA0 は LED0 と端子 4 を共用しています。DA0 のアナログ出力信号用に端子 4 を使用するには、0 オーム抵抗を R277 から R189 に移動する必要があります。これによって、JA1_13 を介して DA0 にアクセスが可能になります。ただし、リンクの設定をこのように変更した場合、LED0 は使用不可となりますのでご注意ください。

— DAC チャネル 1（DA1）出力では、MCU の端子 2 にマップする I/O ポート P05 を使用します。RSKRX64M では、DA1 は LED1 と端子 2 を共用しています。DA1 のアナログ出力信号用に端子 2 を使用するには、0 オーム抵抗を R280 から R188 に移動する必要があります。これによって、JA1_14 を介して DA1 にアクセスが可能になります。ただし、リンクの設定をこのように変更した場合、LED1 は使用不可となりますのでご注意ください。

— RSKRX64M ボードでは、DA 基準電圧 AVCC1 (VREFH)および AVSS1 (VREFL)は、UC_VCC (typ. 3.3V)および GROUND にそれぞれ接続されます。

5.4　dac_demo_rskrx71m, dac_demo_rskrx71m_gcc

dac_demo_rskrx71m, dac_demo_rskrx71m_gcc は、RSKRX71M、RX71M D/A コンバータ（R12DA）対応の DAC FIT モジュール（r_dac_rx）のシンプルなデモプロジェクトです。デモでは r_dac_rx API を使用して、DAC の起動、設定、書き込みを行います。LOW、MEDIUM、または HIGH レベルの値が DAC チャネル 0 に書き込まれると、連続的なループに 1 秒間遷移します。LOW レベルの値が書き込まれると LED1 が点灯し、MEDIUM レベルの値が書き込まれると LED2 が、HIGH レベルの値が書き込まれると LED3 が点灯します。RSKRX71M ボード使用時の DAC 出力チャネル信号のアクセスおよび設定、また基準電圧に関する詳細については、以下の「DAC チャネル 0、チャネル 1 出力信号測定時の注意事項」をご覧ください。

セットアップと実行

1. サンプルコードをコンパイルし、ダウンロードします。
2. 必要な場合、マルチメータ用リードかオシロスコープ用プローブを DAC チャネル出力端子に取り付けます。
3. 「Reset Go」をクリックしてソフトウェアを起動します。PC が Main 関数で停止した場合、F8 を押してレジュームしてください。
4. ブレークポイントを設定し、グローバル変数を確認します。

対応ボード

RSKRX71M

DAC チャネル 0、チャネル 1 出力信号測定時の注意事項

— DAC チャネル 0（DA0）出力では、MCU の端子 4 にマップする I/O ポート P03 を使用します。RSKRX71M では、DA0 は LED0 と端子 4 を共用しています。DA0 のアナログ出力信号用に端子 4 を使用するには、0 オーム抵抗を R281 から R195 に移動する必要があります。これによって、JA1_13 を介して DA0 にアクセスが可能になります。ただし、リンクの設定をこのように変更した場合、LED0 は使用不可となりますのでご注意ください。

— DAC チャネル 1（DA1）出力では、MCU の端子 2 にマップする I/O ポート P05 を使用します。RSKRX71M では、DA1 は LED1 と端子 2 を共用しています。DA1 のアナログ出力信号用に端子 2 を使用するには、0 オーム抵抗を R284 から R194 に移動する必要があります。これによって、JA1_14 を介して DA1 にアクセスが可能になります。ただし、リンクの設定をこのように変更した場合、LED1 は使用不可となりますのでご注意ください。

— RSKRX71M ボードでは、DA 基準電圧 AVCC1 (VREFH)および AVSS1 (VREFL)は、UC_VCC (typ. 3.3V)および GROUND にそれぞれ接続されます。

5.5　dac_demo_rskrx65n, dac_demo_rskrx65n_gcc

dac_demo_rskrx65n, dac_demo_rskrx65n_gcc は、RSKRX65n、RX65n D/A コンバータ（R12DA）対応の DAC FIT モジュール（r_dac_rx）のシンプルなデモプロジェクトです。デモでは r_dac_rx API を使用して、DAC の起動、設定、書き込みを行います。LOW、MEDIUM、または HIGH レベルの値が DAC チャネル 0 に書き込まれると、連続的なループに 1 秒間遷移します。LOW レベルの値が書き込まれると LED1 が点灯し、MEDIUM レベルの値が書き込まれると LED2 が、HIGH レベルの値が書き込まれると LED3 が点灯します。RSKRX65n ボード使用時の DAC 出力チャネル信号のアクセスおよび設定、また基準電圧に関する詳細については、以下の「DAC チャネル 0、チャネル 1 出力信号測定時の注意事項」をご覧ください。

セットアップと実行

1. サンプルコードをコンパイルし、ダウンロードします。
2. 必要な場合、マルチメータ用リードかオシロスコープ用プローブを DAC チャネル出力端子に取り付けます。
3. 「Reset Go」をクリックしてソフトウェアを起動します。PC が Main 関数で停止した場合、F8 を押してレジュームしてください。
4. ブレークポイントを設定し、グローバル変数を確認します。

対応ボード

RSKRX65N

DAC チャネル 0、チャネル 1 出力信号測定時の注意事項

— DAC チャネル 0（DA0）出力では、MCU の端子 4 にマップする I/O ポート P03 を使用します。
RSKRX65n では、DA0 は LED0 と端子 4 を共用しています。DA0 のアナログ出力信号用に端子 4 を使用するには、0 オーム抵抗を R306 から R192 に移動する必要があります。これによって、JA1_13 を介して DA0 にアクセスが可能になります。ただし、リンクの設定をこのように変更した場合、LED0 は使用不可となりますのでご注意ください。

— DAC チャネル 1（DA1）出力では、MCU の端子 2 にマップする I/O ポート P05 を使用します。
RSKRX65n では、DA1 は LED1 と端子 2 を共用しています。DA1 のアナログ出力信号用に端子 2 を使用するには、0 オーム抵抗を R308 から R184 に移動する必要があります。これによって、JA1_14 を介して DA1 にアクセスが可能になります。ただし、リンクの設定をこのように変更した場合、LED1 は使用不可となりますのでご注意ください。
RSKRX65n ボードでは、DA 基準電圧 AVCC1 (VREFH)および AVSS1 (VREFL)は、UC_VCC (typ. 3.3V)および GROUND にそれぞれ接続されます。

5.6　dac_demo_rskrx65n_2m, dac_demo_rskrx65n_2m_gcc

dac_demo_rskrx65n_2m, dac_demo_rskrx65n_2m_gcc は、RSKRX65N-2MB、RX65N-2MB D/A コンバータ（R12DA）対応の DAC FIT モジュール（r_dac_rx）のシンプルなデモプロジェクトです。デモでは r_dac_rx API を使用して、DAC の起動、設定、書き込みを行います。LOW、MEDIUM、または HIGH レベルの値が DAC チャネル 0 に書き込まれると、連続的なループに 1 秒間遷移します。LOW レベルの値が書き込まれると LED1 が点灯し、MEDIUM レベルの値が書き込まれると LED2 が、HIGH レベルの値が書き込まれると LED3 が点灯します。

RSKRX65N-2MB ボード使用時の DAC 出力チャネル信号のアクセスおよび設定、また基準電圧に関する詳細については、以下の「DAC チャネル 0、チャネル 1 出力信号測定時の注意事項」をご覧ください。

セットアップと実行

1. サンプルコードをコンパイルし、ダウンロードします。
2. 必要な場合、マルチメータ用リードかオシロスコープ用プローブを DAC チャネル出力端子に取り付けます。
3. 「Reset Go」をクリックしてソフトウェアを起動します。PC が Main 関数で停止した場合、F8 を押してレジュームしてください。
4. ブレークポイントを設定し、グローバル変数を確認します。

対応ボード

RSKRX65N-2MB

DAC チャネル 0、チャネル 1 出力信号測定時の注意事項

- DAC チャネル 0（DA0）出力では、MCU の端子 4 にマップする I/O ポート P03 を使用します。RSKRX65N-2MB では、DA0 は Switch1 と端子 4 を共用しています。DA0 のアナログ出力信号用に端子 4 を使用するには、0 オーム抵抗を R480 から R120 に移動する必要があります。これによって、JA1_13 を介して DA0 にアクセスが可能になります。ただし、リンクの設定をこのように変更した場合、Switch1 は使用不可となりますのでご注意ください。
- DAC チャネル 1（DA1）出力では、MCU の端子 2 にマップする I/O ポート P05 を使用します。RSKRX65N-2MB では、DA1 は Switch2 と端子 2 を共用しています。DA1 のアナログ出力信号用に端子 2 を使用するには、0 オーム抵抗を R479 から R119 に移動する必要があります。これによって、JA1_14 を介して DA1 にアクセスが可能になります。ただし、リンクの設定をこのように変更した場合、Switch2 は使用不可となりますのでご注意ください。
- RSKRX65N-2MB ボードでは、DA 基準電圧 AVCC1 (VREFH)および AVSS1 (VREFL)は、UC_VCC (typ. 3.3V)および GROUND にそれぞれ接続されます。

5.7　dac_demo_rskrx72m, dac_demo_rskrx72m_gcc

dac_demo_rskrx72m, dac_demo_rskrx72m_gcc は、RSKRX72M、RX72M D/A コンバータ（R12DA）対応の DAC FIT モジュール（r_dac_rx）のシンプルなデモプロジェクトです。デモでは r_dac_rx API を使用して、DAC の起動、設定、書き込みを行います。LOW、MEDIUM、または HIGH レベルの値が DAC チャネル 0 に書き込まれると、連続的なループに 1 秒間遷移します。LOW レベルの値が書き込まれると LED1 が点灯し、MEDIUM レベルの値が書き込まれると LED2 が、HIGH レベルの値が書き込まれると LED3 が点灯します。

RSKRX72M ボード使用時の DAC 出力チャネル信号のアクセスおよび設定、また基準電圧に関する詳細については、以下の「DAC チャネル 0、チャネル 1 出力信号測定時の注意事項」をご覧ください。

セットアップと実行

5. サンプルコードをコンパイルし、ダウンロードします。
6. 必要な場合、マルチメータ用リードかオシロスコープ用プローブを DAC チャネル出力端子に取り付けます。
7. 「Reset Go」をクリックしてソフトウェアを起動します。PC が Main 関数で停止した場合、F8 を押してレジュームしてください。
8. ブレークポイントを設定し、グローバル変数を確認します。

対応ボード

RSKRX72M

DAC チャネル 0、チャネル 1 出力信号測定時の注意事項

- DAC チャネル 0（DA0）出力では、MCU の端子 4 にマップする I/O ポート P03 を使用します。RSKRX72M では、DA0 は SERIAL-CTS と端子 4 を共用しています。DA0 のアナログ出力信号用に端子 4 を使用するには、0 オーム抵抗を R275 から R218 に移動する必要があります。これによって、JA1_13 を介して DA0 にアクセスが可能になります。ただし、リンクの設定をこのように変更した場合、SERIAL-CTS は使用不可となりますのでご注意ください。
- DAC チャネル 1（DA1）出力では、MCU の端子 2 にマップする I/O ポート P05 を使用します。RSKRX72M では、DA1 は DSW-CATID3 と端子 2 を共用しています。DA1 のアナログ出力信号用に端子 2 を使用するには、0 オーム抵抗を R201 から R214 に移動する必要があります。これによって、JA1_14 を介して DA1 にアクセスが可能になります。ただし、リンクの設定をこのように変更した場合、DSW-CATID3 は使用不可となりますのでご注意ください。
- RSKRX72M ボードでは、DA 基準電圧 AVCC1 (VREFH)および AVSS1 (VREFL)は、UC_VCC (typ. 3.3V)および GROUND にそれぞれ接続されます。

5.8　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。

5.9 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

本 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.5.31)

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.5.31
使用ボード	-

表 6.2 動作確認環境 (Rev.5.30)

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.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.5.30
使用ボード	Evaluation Kit for RX261 （型名：RTK5EK2610S00011BJ）

表 6.3 動作確認環境 (Rev.5.20)

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.2024-04 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 V3.05.00 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 8.3.0.202305 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.5.20
使用ボード	Renesas Starter Kit for RX72T (型名：RTK5572TKCC00000BE)

表 6.4 動作確認環境 (Rev.5.10)

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.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.5.10
使用ボード	Renesas Solution Starter Kit for RX23E-B (型名：RTK0ES1001C00001BJ)

表 6.5 動作確認環境 (Rev.5.00)

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.2022-10 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.5.00
使用ボード	Renesas Flexible Motor Control Kit for RX26T(型名：RTK0EMXE70S00020BJ)

表 6.6 動作確認環境 (Rev.4.90)

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.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 8.3.0.202202 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.4.90
使用ボード	Renesas Starter Kit for RX113 (型名: R0K505113CxxxBE) Renesas Starter Kit for RX231 (型名: R0K505231SxxxBE) Renesas Starter Kit for RX64M (型名: R0K50564MxxxxBE) Renesas Starter Kit+ for RX65N (型名: RTK5005651CxxxxxBE) Renesas Starter Kit+ for RX65N-2MB (型名: RTK50565N2CxxxxxBR) Renesas Starter Kit for RX71M (型名: R0K50571MCxxxBE) Renesas Starter Kit+ for RX72M (型名: RTK5572MNDCxxxxxBJ)

表 6.7 動作確認環境 (Rev.4.80)

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.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.4.80
使用ボード	Renesas Starter Kit for RX660（型名：RTK556609HCxxxxxBJ）

表 6.8 動作確認環境 (Rev.4.70)

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.2021-10 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 8.3.0.202104 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.4.70
使用ボード	Renesas Starter Kit for RX66T (型名: RTK50566T0SxxxxxBE)

表 6.9 動作確認環境 (Rev.4.60)

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.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.4.60
使用ボード	Target board for RX140 （型名：RTK5RX140xxxxxxxxx）

表 6.10 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.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.4.50
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 RX65N （型名：RTK50565NCxxxxxBE） Renesas Starter Kit+ for RX64M （型名：RTK50564Mxxxxxxxx） Renesas Starter Kit+ for RX71M （型名：RTK50571Mxxxxxxxx） Renesas Starter Kit+ for RX113 （型名：RTK505113xxxxxxxx） Renesas Starter Kit+ for RX231 （型名：RTK505231xxxxxxxx）

表 6.11 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.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.4.40
使用ボード	Renesas Starter Kit+ for RX72N（型名：RTK5572Nxxxxxxxxxx）

表 6.12 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.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.4.30
使用ボード	RX13T CPU Card（型名：RTK0EMXA10C00000BJ）

表 6.13 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.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.4.20
使用ボード	Renesas Starter Kit+ for RX72M（型名：RTK5572Mxxxxxxxxxx）

表 6.14 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 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.4.10
使用ボード	Renesas Solution Starter Kit for RX23W（型名：RTK5523Wxxxxxxxxx）

表 6.15 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.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.4.00
使用ボード	Renesas Starter Kit+ for RX65N-2MB（型名：RTK50565Nxxxxxxxxx）

表 6.16 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 e^2 studio V.7.3.0
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.3.30
使用ボード	Renesas Starter Kit for RX72T（型名：RTK5572Txxxxxxxxx）

表 6.17 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 e^2 studio V.7.3.0
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.3.21
使用ボード	Renesas Starter Kit for RX66T（型名：RTK50566T0SxxxxxBE） Renesas Starter Kit+ for RX65N-2M（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

表 6.18 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.7.0.0
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 V3.00.00
	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.3.20
使用ボード	Renesas Starter Kit for RX66T（型名：RTK50566T0SxxxxxBE） Renesas Starter Kit+ for RX65N-2M（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

表 6.19 動作確認環境 (Rev.3.11)

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.6.0.0
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 V2.07.00
	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
モジュールのリビジョン	Rev.3.11
使用ボード	Renesas Starter Kit+ for RX65N-2M（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

表 6.20 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 e² studio V.6.0.0
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 V2.07.00
	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
モジュールのリビジョン	Rev.3.10
使用ボード	Renesas Starter Kit+ for RX65N-2M（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

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)」
- e² 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_dac_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：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「コンフィグ設定が間違っている場合のエラーメッセージ」エラーが発生します。

A：“r_dac_rx_config.h”ファイルの設定値が間違っている可能性があります。“r_dac_rx_config.h”ファイルを確認して正しい値を設定してください。詳細は「2.7 コンパイル時の設定」を参照してください。

(4) Q：アナログ出力されません。

A：正しく端子設定が行われていない可能性があります。本 FIT モジュールを使用する場合は端子設定が必要です。詳細は「4 端子設定」を参照してください。

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。
　なし

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2013.11.15	—	初版発行
2.00	2014.04.02	—	新しい API、RX210、RX63N/631 のサポートを更新。
2.10	2014.09.08	—	RX64M のサポートを追加。
2.20	2015.01.20	—	RX113 のサポートを追加。
2.30	2015.03.19	—	RX71M のサポートを追加。
2.40	2015.06.30	—	RX231 のサポートを追加。
2.50	2015.09.30	—	RX23T のサポートを追加。
2.60	2015.10.01	—	FIT モジュールの RX130 グループ対応。
2.70	2015.12.01	— 1, 5 3 3 7, 11 14	FIT モジュールの RX230 グループ対応。 『ボードサポートパッケージモジュール Firmware Integration Technology』アプリケーションノートのドキュメント番号を変更。 セクション 2 の説明を変更。 「2.1 ハードウェアの要求」、「2.2 ハードウェアリソースの要求」：“DAA”の記載を削除。 「3.3 R_DAC_Open()」、「3.6 R_DAC_Control()」：“Parameters”および “Example”に記載のコードを修正。 「4. デモプロジェクト」を追加。
2.80	2016.02.01	— 18	FIT モジュールの RX24T グループ対応。 「テクニカルアップデートの対応について」を追加。
2.91	2016.10.01	— 5 8 11, 12	FIT モジュールの RX65N グループ対応。 ROM,RAM およびスタックのコードサイズ変更。 アンプを使用する時の注意事項を追加。 Description、Example 記述変更 Special Notes 記述追加。
3.00	2017.02.28	— 3 9, 10, 13 プログラム	FIT モジュールの RX24T グループ（ROM 512KB 版を含む）、RX24U グループ対応。 「2.5 対応ツールチェーン」に RXC v2.06.00 を追加。 3.3 R_DAC_Open()、3.4 R_DAC_Close()、3.6 R_DAC_Control(): Special Notes に、D/A A/D 同期変換有効設定時の注意事項を追加。 RX64M、RX71M、RX65N において、R_DAC_Open 関数で同期対象ユニットにユニット 0 を設定、かつ D/A A/D 同期変換を有効に設定した場合はエラーとなるように修正。
3.10	2017.07.21	— 3 6 15	FIT モジュールの RX130 グループ（ROM 512KB 版）と RX65N グループ（ROM 2MB 版）対応。 「2.5 対応ツールチェーン」に RXC v2.07.00 を追加。 「2.11 FIT モジュールをプロジェクトに追加する方法」を変更。 「4. 端子設定」を追加。
3.11	2017.10.31	18 19, 20 20 21	「5.4 dac_demo_rskrx71m の DAC チャネル 0/1 の出力信号の測定に関する注意」：チャネル 0 の抵抗を R281 と R195 に、またチャネル 1 の抵抗を R284 と R194 に変更。 「5. デモプロジェクト」に RSKRX65N, と RSKRX65N-2M を追加。 「5.8 デモのダウンロード方法」を追加。 「6. 付録」を追加。
3.20	2018.09.28	1, 3 5 23	RX66T のサポートを追加。 RX66T 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。

3.21	2018.11.16	— 23	XML 内にドキュメント番号を追加。 Renesas Starter Kit+ for RX66T の型名を変更。 Rev.3.21 に対応する表を追加。
3.30	2019.02.01	— 1、3 5 9-16 24	RX72T グループのサポートを追加。 RX72T グループのサポートを追加。 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。 「6.1 動作確認環境」Rev.3.30 に対応する表を追加。
4.00	2019.05.20	— 1 3 4 5-7 26 30 プログラム	以下のコンパイラをサポート。 - GCC for Renesas RX - IAR C/C++ Compiler for Renesas RX RX210、RX631、RX63N の更新終了につき、「対象デバイス」からこれらのデバイスを削除。 「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。 「1.2 DAC FIT モジュールの概要」 RX210、RX631、RX63N の説明を削除。 「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 6.1「動作確認環境」： Rev.4.00 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。 GCC と IAR コンパイラに関して、以下を変更。 R_DAC_GetVersion 関数のインライン展開を削除。
4.10	2019.06.28	1 5 26 プログラム	RX23W のサポートを追加。 RX23W 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。 RX23W のサポートを追加。
4.20	2019.08.15	1 6-8 27 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.4.20 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。
4.30	2019.11.25	1, 3 4 7-9 28 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.4.30 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。

4.40	2019.12.30	1, 3 7-9 18 19 28 プログラム	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。 RX65N, RX66N, RX72M, RX72N の dac_cmd_t に新しい cmd 値を追加。 Special Notes に出力アンプ安定待ち使用時の手順を追加。 「6.1 動作確認環境」： Rev.4.40 に対応する表を追加。 RX66N, RX72N のサポートを追加。 RX65N, RX66N, RX72M, RX72N の出力アンプ安定待 ちのサポートを追加。
4.50	2020.06.30	23-28 30 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.4.50 に対応する表を追加。 デモプロジェクトの更新と追加
4.60	2021.04.15	1, 4 4 8-12 33 プログラム	RX140 のサポートを追加。 「1.3 DTC FIT モジュールを使用する」のセクションを加。 「1.3.DTC FIT モジュールを C++プロジェクト内で使用る」 RX140 に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.4.60 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。
4.70	2022.03.14	33 プログラム	「6.1 動作確認環境」： Rev.4.70 に対応する表を追加。 RX66T-48Pin のサポートを追加。
4.80	2022.03.31	1, 4 8, 10, 12 33 プログラム	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.4.80 に対応する表を追加。 RX660 のサポートを追加。
4.90	2022.07.29	33 プログラム	「6.1 動作確認環境」： Rev.4.90 に対応する表を追加。 デモプロジェクトを更新。
5.00	2022.08.15	1, 4 8, 10, 12 33 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.5.00 に対応する表を追加。 RX26T のサポートを追加。

5.10	2023.05.29	1, 4 8, 10, 12 13 13, 21 18, 19, 23 20 21 34 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 RX23E-B の dac_err_t に新しい戻り値を追加。 R_DAC_Open()、 R_DAC_Close()、 R_DAC_Control()：Special Notes の D/A A/D 同期変換許可時の注意事項を更新。 「R_DAC_Write()」の Description の記載を変更 RX23E-B の dac_cmd_t に新しいコマンドを追加。 「6.1 動作確認環境」： Rev.5.10 に対応する表を追加。 RX23E-B のサポートを追加。
5.20	2024.06.14	33 プログラム	「6.1 動作確認環境」： Rev.5.20 に対応する表を追加。 RX66T、RX72T、RX660、RX26T の DA0 と DA1 への同時出力用に dac_out_da_t enum に DAC_OUT_SEL_DA0_DA1 を追加。 RX66T、RX72T、RX660、RX26T の DA0 と DA1 からの Vref 同時出力用に dac_out_ref_t enum に DAC_OUT_SEL_REF0_REF1 を追加。
5.30	2024.06.28	1, 4 8, 10, 12 33 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.5.30 に対応する表を追加。 RX260, RX261 のサポートを追加。
5.31	2025.03.15	33 プログラム	「6.1 動作確認環境」： Rev.5.31 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

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。

© 2025 Renesas Electronics Corporation. All rights reserved.