RX671 グループ

QSPIX XIP モードを使用したシリアル ROM 上のプログラム実行例

要旨

本アプリケーションノートは、RX671 グループの QSPIX モジュール(以下、QSPIX)に搭載されている XIP モードを使用して、シリアル ROM 上に配置されたプログラムを実行する例について説明します。

本アプリケーションノートでは、一例として、以下の 3 つのサンプルプログラムを提供します。

- アプリケーションプログラム
  (シリアル ROM 上に配置するプログラムを含んだアプリケーションプログラム)
- ライタ用プログラム 1
  (アプリケーションプログラムの一部をライタ用プログラム 1 の内蔵 ROM に取り込みシリアル ROM へ書き込むプログラム)
- ライタ用プログラム 2
  (アプリケーションプログラムの一部をホスト PC からシリアル通信で受信しシリアル ROM へ書き込むプログラム)

動作確認デバイス

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合わせて変更し、十分評価してください。

目次

1.　QSPIX の XIP モードとプリフェッチ機能

本アプリケーションノートでは、シリアル ROM から命令コードを読み出す際、QSPIX の XIP モード、およびプリフェッチ機能を使用します。

XIP モードとプリフェッチ機能について以下に説明します。

1.1　XIP モードの概要

シリアル ROM の中には、命令コード受信を省略することで ROM 読み出しを高速化できるものがあります。この機能は、直前の SPI バスサイクル内のダミーサイクルに含まれるモードデータによって制御されます。

1.2　XIP モードの有効化

XIP モードはメモリマップドモード時(SPMR1 レジスタの AMOD ビットが“1”に使用できます。

XIP モードを有効にするには、シリアル ROM を XIP モードにするための値を SPDCR レジスタの MODE[7:0]ビットに指定し、SPDCR レジスタの XIPE ビットを“1”にします。

これにより、次の SPI バスサイクルでは、図 1 の XIP モード制御データに示すように、SPDCR レジスタの MODE[7:0]ビットに指定した値がダミーサイクルに含まれて送信されます。

XIP モードが有効になったかどうかは、上記の SPI バスサイクル後に SPDCR レジスタの XIPS フラグをリードすることで確認できます。

なお、XIP モードを有効にするモードデータは、シリアル ROM ごとに異なりますので、使用するシリアル ROM に合わせて SPDCR レジスタの MODE[7:0]ビットを設定する必要があります。

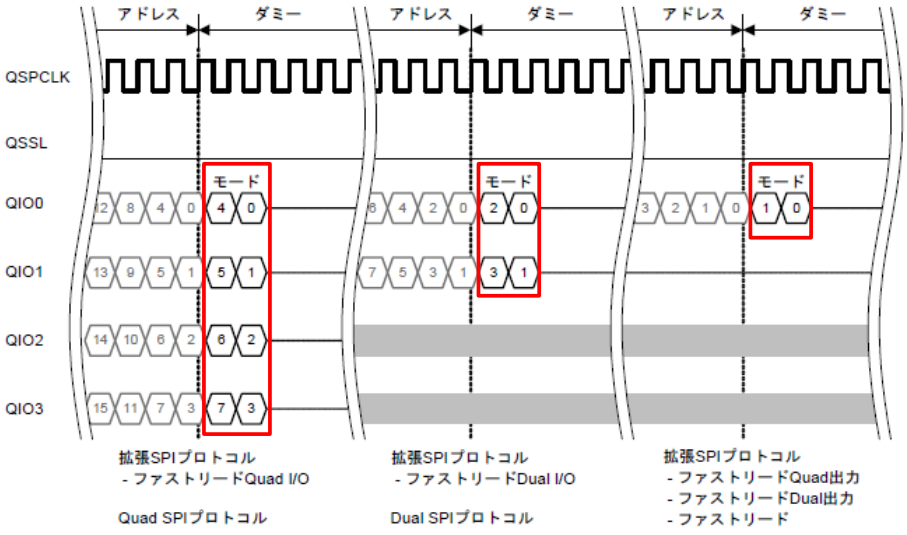


図 1　XIP モード制御データ

1.3　XIP モードの終了

XIP モードを終了するには、使用するシリアル ROM の XIP モードを解除するための値を SPDCR レジスタの MODE[7:0]ビットに指定し、SPDCR レジスタの XIPE ビットを“0”にします。次の SPI バスサイクルでは、SPDCR レジスタの MODE[7:0]ビットに指定した値がダミーサイクルに含まれて送信されます。

XIP モードが終了したかどうかは、上記の SPI バスサイクル後に SPDCR レジスタの XIPS フラグをリードすることで確認できます。

1.4　プリフェッチ機能について

QSPIX はプリフェッチ機能を備えています。

シリアル ROM のメモリリードコマンドでは、1 回の SPI バスサイクルで無限にデータを読み出すことができる特性がありますが、CPU が発行するバスサイクルを個別に SPI バスサイクルに変換していては、SPI バスサイクルが分断され、シリアル ROM が持つこの特性を活かせません。

プリフェッチ機能により、この特性を活かして命令実行を高速化することができます。

プリフェッチ機能は、SPMR0 レジスタの PFE ビットを“1”にすると有効になります。プリフェッチ機能が有効になっていると、QSPIX は次の ROM 読み出し要求を待つことなく、データを連続で受信してバッファに格納します。次に CPU が ROM 読み出しを行うと、アドレスの比較を行い、アドレスが一致していればバッファ内のデータを CPU に返します。アドレスが一致していなければ、バッファ内のデータを破棄し、新たな SPI バスサイクルを生成します。

2. ハードウェア構成

2.1 Renesas Starter Kit+ for RX671

図 2 に Renesas Starter Kit+ for RX671(以下、RSK)ボード搭載の RX671 とシリアル ROM との接続図を示します。

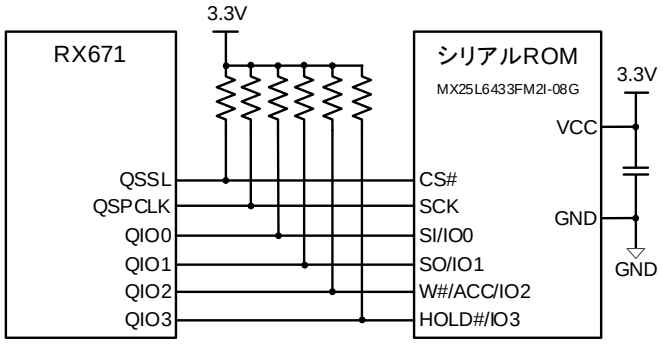


図 2　Renesas Starter Kit+ for RX671 ボード搭載の RX671 とシリアル ROM との接続図

また、ライタ用プログラム 2 では、アプリケーションプログラムの一部をホスト PC からシリアル通信を使用して受信します。図 3 に RX671 とホスト PC との接続図を示します。

RSK は USB シリアル変換回路が搭載されており、RSK とホスト PC とを USB 接続することにより仮想 COM ポートとして RX671 とシリアル通信で送受信することができます。

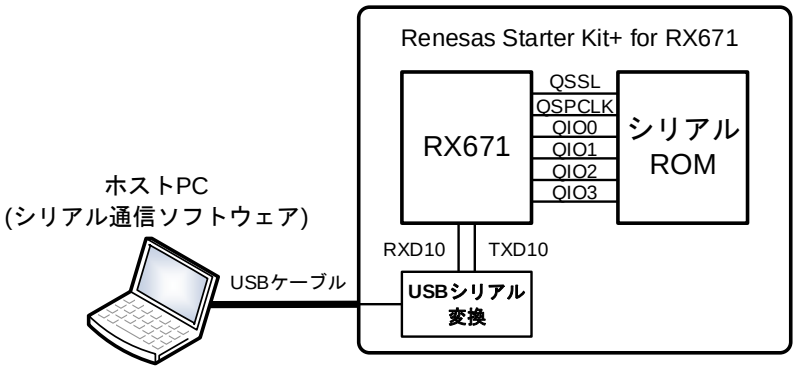


図 3　RX671 とホスト PC との接続図

表 2.1 に RX671 とシリアル ROM との接続に使用する QSPIX 端子を示します。

表 2.1　RX671 とシリアル ROM との接続に使用する QSPIX 端子

端子名	入出力	機能
QSSL	出力	スレーブセレクト端子
QSPCLK	出力	クロック出力端子
QIO0	入出力	データ 0 入出力
QIO1	入出力	データ 1 入出力
QIO2	入出力	データ 2 入出力
QIO3	入出力	データ 3 入出力

表 2.2 に RX671 とホスト PC との接続に使用する SCI 端子を示します。

表 2.2　RX671 とホスト PC との接続に使用する SCI 端子

端子名	入出力	機能
RXD10	入力	受信データ入力端子
TXD10	出力	送信データ出力端子

本アプリケーションでは、RSK 搭載の LED を制御します。

表 2.3 に LED 制御に使用している端子を示します。

表 2.3　LED 制御に使用している端子

端子名	機能
P17	LED0 制御
PF5	LED1 制御
P03	LED2 制御
P05	LED3 制御

2.2　EK-RX671

図 4 に EK-RX671 ボード搭載の RX671 とシリアル ROM との接続図を示します。

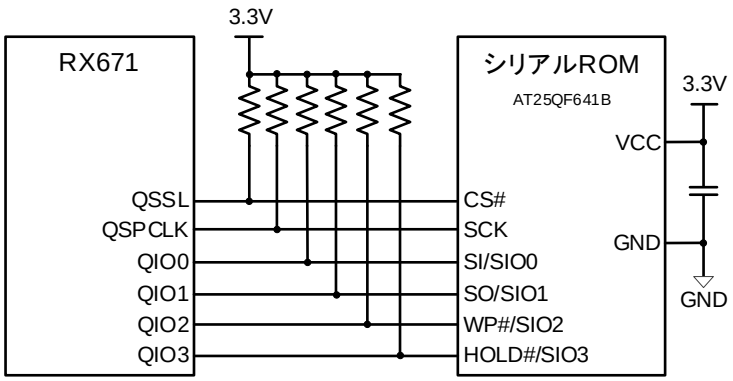


図 4　EK-RX671 ボード搭載の RX671 とシリアル ROM との接続図

また、ライタ用プログラム 2 では、アプリケーションプログラムの一部をホスト PC からシリアル通信を使用して受信します。図 5 に RX671 とホスト PC との接続図を示します

EK は USB シリアル変換回路が搭載されており、EK とホスト PC とを USB 接続することにより仮想 COM ポートとして RX671 とシリアル通信で送受信することができます。

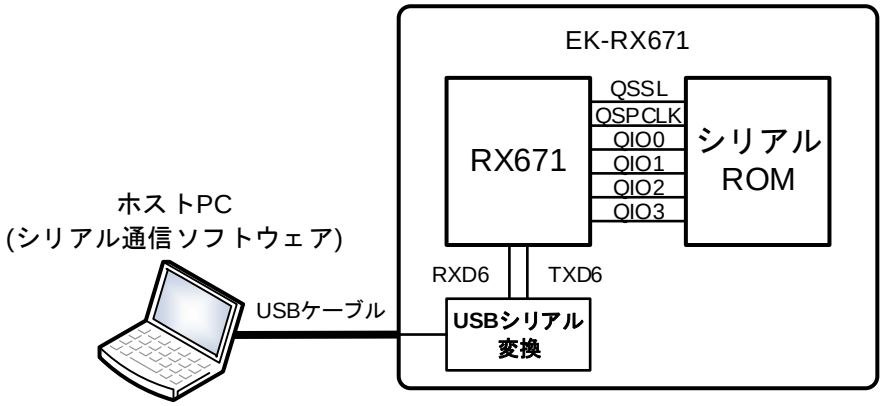


図 5　RX671 とホスト PC との接続図

表 2.4 に RX671 とシリアル ROM との接続に使用する QSPIX 端子を示します。

表 2.4　RX671 とシリアル ROM との接続に使用する QSPIX 端子

端子名	入出力	機能
QSSL	出力	スレーブセレクト端子
QSPCLK	出力	クロック出力端子
QIO0	入出力	データ 0 入出力
QIO1	入出力	データ 1 入出力
QIO2	入出力	データ 2 入出力
QIO3	入出力	データ 3 入出力

表 2.5 に RX671 とホスト PC との接続に使用する SCI 端子を示します。

表 2.5　RX671 とホスト PC との接続に使用する SCI 端子

端子名	入出力	機能
RXD6	入力	受信データ入力端子
TXD6	出力	送信データ出力端子

本アプリケーションでは、EK 搭載の LED を制御します。

表 2.6 に LED 制御に使用している端子を示します。

表 2.6　LED 制御に使用している端子

端子名	機能
P56	LED1 制御
P82	LED2 制御
P25	LED3 制御

3. サンプルプログラム

本アプリケーションノートのサンプルプログラムは、外部メモリとしてシリアル ROM を使用し、シリアル ROM 上のプログラムを QSPIX の XIP モードを使用して読み出して実行します。

サンプルプログラムは、RSK ボードと EK ボードに向け、それぞれ３つ用意しています。

表 3.1　サンプルプログラム(RSK ボード向け)

プロジェクト名	説明
xip_sample_rx671	・アプリケーションプログラム シリアル ROM 上に配置するプログラムを含んだ アプリケーションプログラム
serialROM_write1_direct_rx671	・ライタ用プログラム 1 アプリケーションプログラムの一部を内蔵 ROM に取り込み シリアル ROM へ書き込むプログラム
serialROM_write2_serial_rx671	・ライタ用プログラム 2 アプリケーションプログラムの一部をホスト PC から シリアル通信で受信しシリアル ROM へ書き込むプログラム

表 3.2　サンプルプログラム(EK ボード向け)

プロジェクト名	説明
xip_sample_rx671_ek	・アプリケーションプログラム シリアル ROM 上に配置するプログラムを含んだ アプリケーションプログラム
serialROM_write1_direct_rx671_ek	・ライタ用プログラム 1 アプリケーションプログラムの一部を内蔵 ROM に取り込み シリアル ROM へ書き込むプログラム
serialROM_write2_serial_rx671_ek	・ライタ用プログラム 2 アプリケーションプログラムの一部をホスト PC から シリアル通信で受信しシリアル ROM へ書き込むプログラム

本サンプルプログラムは、統合開発環境として e^2 studio とスマートコンフィグレータ(以下、SC)を使用しています。また、周辺機能の設定および制御用のプログラムに Firmware Integration Technology (以下、FIT) モジュールを使用しています。

使用している FIT モジュールや設定内容の詳細は「3.3 使用 FIT モジュール」を参照してください。

3.1　アプリケーションプログラム

3.1.1　プログラムの仕様

3.1.1.1　ソフトウェアの説明

本アプリケーションプログラムは、プログラム配置を内蔵 ROM とシリアル ROM に振り分けます。

高速動作が必要なプログラムは内蔵 ROM に配置し、低速動作で問題ないプログラムはシリアル ROM 上に配置することを推奨します。

また、アプリケーションプログラムのビルド生成ファイルは、内蔵 ROM 上に配置するプログラムとシリアル ROM 上に配置するプログラムとで分割出力します。

(1)　本アプリケーションプログラムのアドレス配置

図 6 にアプリケーションプログラムのアドレス配置を示します。

内蔵 ROM 上に配置するプログラムは、RX671 のクロックや QSPIX、シリアル ROM の初期設定、XIP モードへの遷移、シリアル ROM 上のプログラムへの分岐処理などです。

シリアル ROM 上に配置するプログラムは、ボード上に搭載されている LED を順次点灯させます。

また、シリアル ROM 上に配置するプログラムは、セクション名を SerialROM_sec としています。

図 6　アプリケーションプログラムのアドレス配置

(2)　アプリケーションプログラムのビルド生成ファイルの分割出力

図 7 にアプリケーションプログラムのビルド生成ファイルの分割出力を示します。

本図に示すように、内蔵 ROM 上に配置するプログラムはモトローラ S 形式で出力します(ROM_block.mot)。

シリアル ROM 上に配置するプログラムはバイナリとモトローラ S 形式で出力します。

バイナリファイル(SerialROM_block.bin)はライタ用プログラム 1 で使用します。

モトローラ S 形式ファイル(SerialROM_block.mot)はライタ用プログラム 2 で使用します。

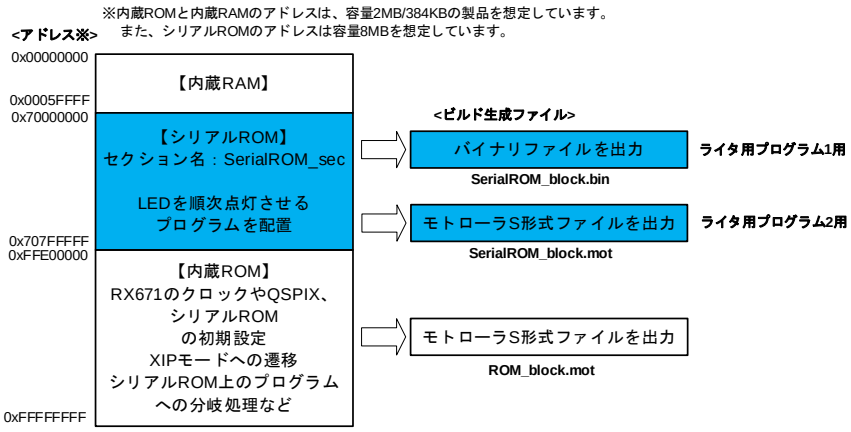


図 7　アプリケーションプログラムのビルド生成ファイルの分割出力

3.1.1.2　e² studio でのビルド設定

e² studio で必要なオプション設定について説明します。

(1)　シリアル ROM 上に配置するプログラムのセクション割り当て

シリアル ROM 上に配置するプログラムにセクションを割り当てます。

プロジェクトのプロパティを開き、「C/C++ビルド」→「設定」をクリックし、右の表示タブから「ツール設定」を選択します。そこから、「Linker」→「セクション」を選択し、図 8 のシリアル ROM 上に配置するプログラムのセクション割り当て(1/2)の画面を表示します。

「セクション(-start)」の右にある[…]ボタンをクリックします。

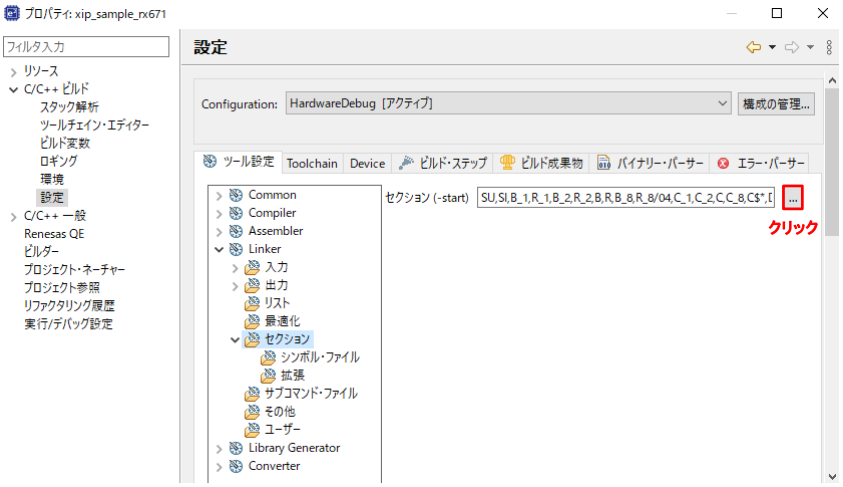


図 8　シリアル ROM 上に配置するプログラムのセクション割り当て(1/2)

次に、図 9 に示すように、「セクション・ビューアー」で「セクションの追加」ボタンをクリックし、シリアル ROM 上に配置するプログラムのアドレスとセクション名を追加します。

本アプリケーションプログラムでは次のように設定しています。

アドレス：0x70000000(シリアル ROM が割り当てられる QSPI 領域の先頭アドレス)

ライタ用プログラム 1 使用時は上記のアドレスを変更しないでください。

ライタ用プログラム 2 使用時はアドレスを QSPI 領域の 256 の倍数(下位 1 バイトが 0x00)にしてください。

セクション名：SerialROM_sec

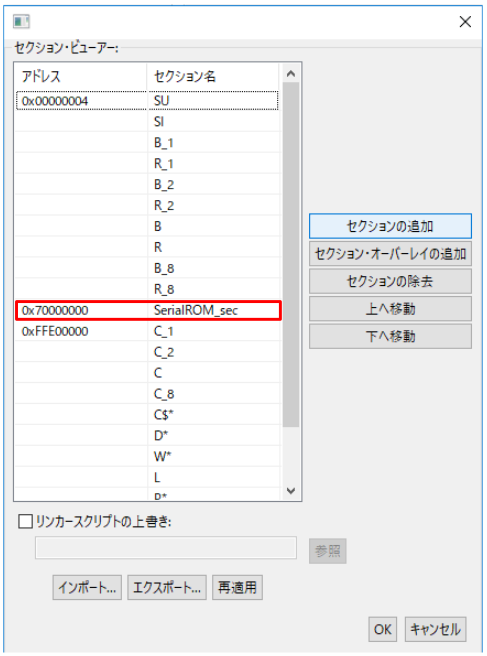


図 9　シリアル ROM 上に配置するプログラムのセクション割り当て(2/2)

(2)　ビルド生成ファイルの分割出力

プロジェクトのプロパティを開き、「C/C++ビルド」→「設定」をクリックし、右の表示タブから「ツール設定」を選択します。そこから、「Converter」→「出力」を選択し、図 10 の　e² studio でのビルド生成ファイルの分割出力設定(1/2)の画面を開きます。

「モトローラ S 形式ファイルを出力する(-form=stype)」と「バイナリ・ファイルを出力する(-form=binary)」のチェックボックスにチェックを入れます。

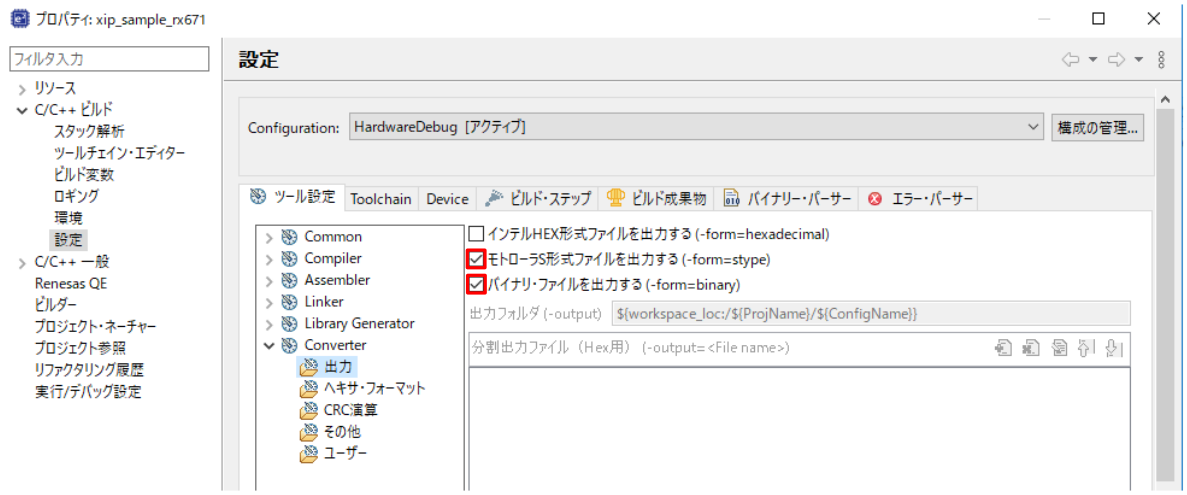


図 10　e² studio でのビルド生成ファイルの分割出力設定(1/2)

次に、右のスクロールバーを下に移動し、図 11 の e² studio でのビルド生成ファイルの分割出力設定(2/2)に示すように、「分割出力ファイル (Stype 用) (-output=<File name>)」の追加ボタンをクリックし

ROM_block.mot=ffe00000-ffffffff

と

SerialROM_block.mot=SerialROM_sec

を追加し、「分割出力ファイル (Bin 用) (-output=<File name>)」の追加ボタンをクリックし

SerialROM_block.bin=SerialROM_sec

を追加します。

ここで、SerialROM_sec は、「3.1.1.2(1) シリアル ROM 上に配置するプログラムのセクション割り当て」で説明したシリアル ROM 上に配置するプログラムのセクション名です。

「適用して閉じる」ボタンをクリックしてプロジェクトのプロパティを閉じます（以降の設定も続けて実施する場合は不要です）。

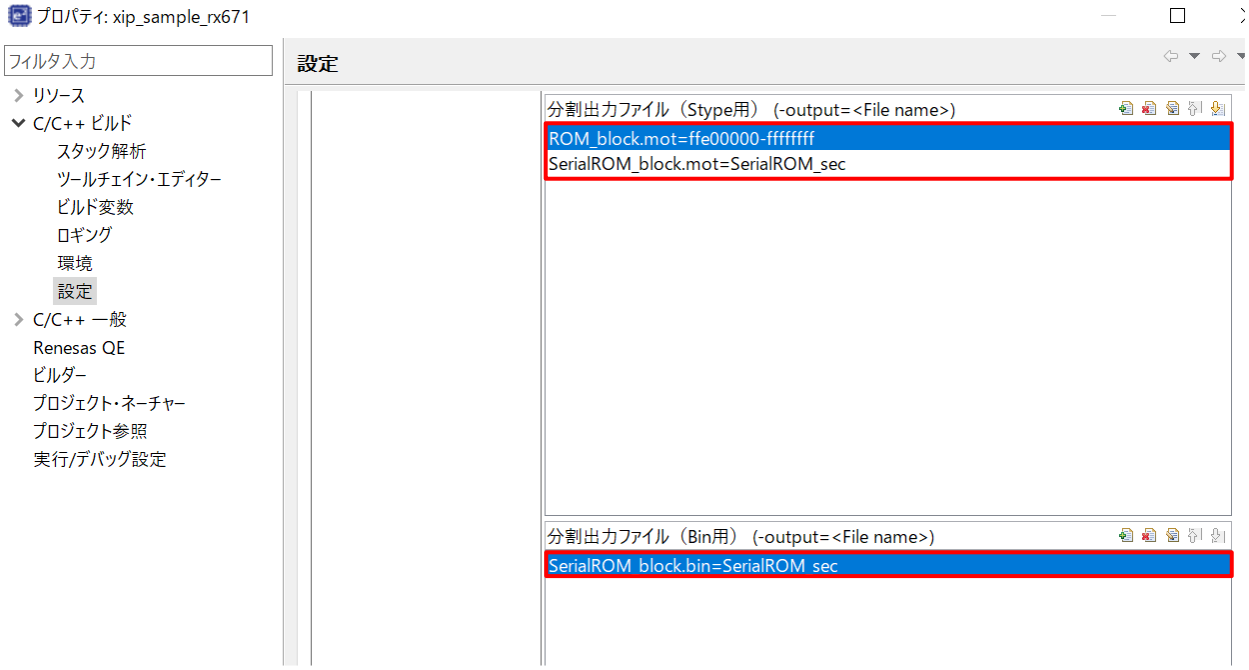


図 11　e² studio でのビルド生成ファイルの分割出力設定(2/2)

(3)　分岐幅(-branch)オプション設定

シリアル ROM が割り当てられる QSPI 領域は、アドレスが 0x70000000～0x77FFFFFF であるため、プログラムが内蔵 ROM からシリアル ROM に分岐するには、24bit の分岐幅では足りません。このため、分岐幅(-branch)オプションを変更します。

分岐幅(-branch)オプションを設定するには、プロジェクトのプロパティを開き、「C/C++ビルド」→「設定」をクリックし、右の表示タブから「ツール設定」を選択します。そこから、「Common」→「CPU」を選択し、図 12 の分岐幅(-branch)オプション設定の画面を開きます。

「分岐幅(-branch)」のプルダウンメニューから「指定しない」を選択します。
「適用して閉じる」ボタンをクリックしてプロジェクトのプロパティを閉じます（以降の設定も続けて実施する場合は不要です）。

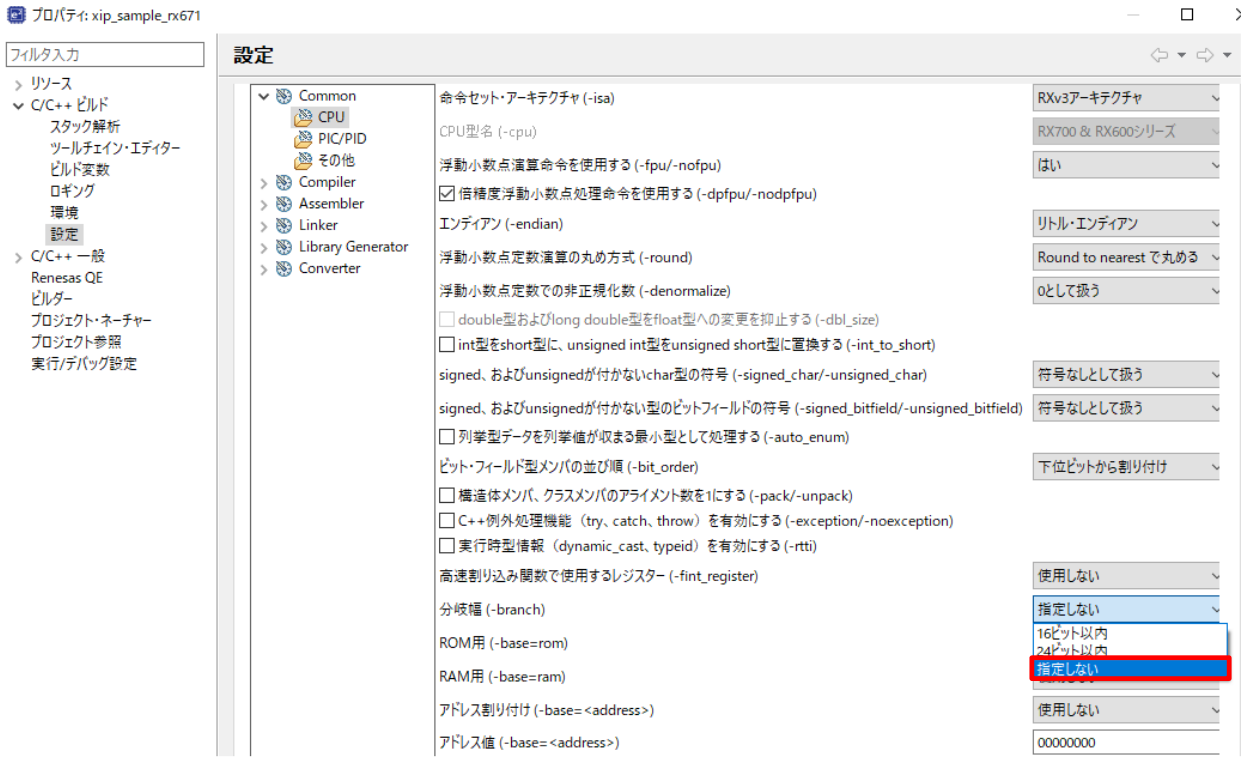


図 12　分岐幅(-branch)オプション設定

(4)　セクションの割り付けアドレスのチェック(-cpu)オプション設定

シリアル ROM が割り当てられる QSPI 領域(アドレス 0x70000000～0x77FFFFFF)にプログラムを配置した場合、デフォルトの-cpu オプション設定ではエラーが発生します。

これは、-cpu オプションによってセクションの割り付けアドレスがチェックされ、QSPI 領域がアドレス指定範囲外と判断されるためです。

本アプリケーションプログラムでは、セクションの割り付けアドレスのチェックを外しています。

プロジェクトのプロパティを開き、「C/C++ビルド」→「設定」をクリックし、右の表示タブから「ツール設定」を選択します。そこから、「Linker」→「セクション」→「拡張」を選択し、図 13 のセクションの割り付けアドレスチェック(-cpu)オプション設定の画面を開きます。

「セクションの割り付けアドレスをチェックする(-cpu)」のチェックボックスのチェックを外します。

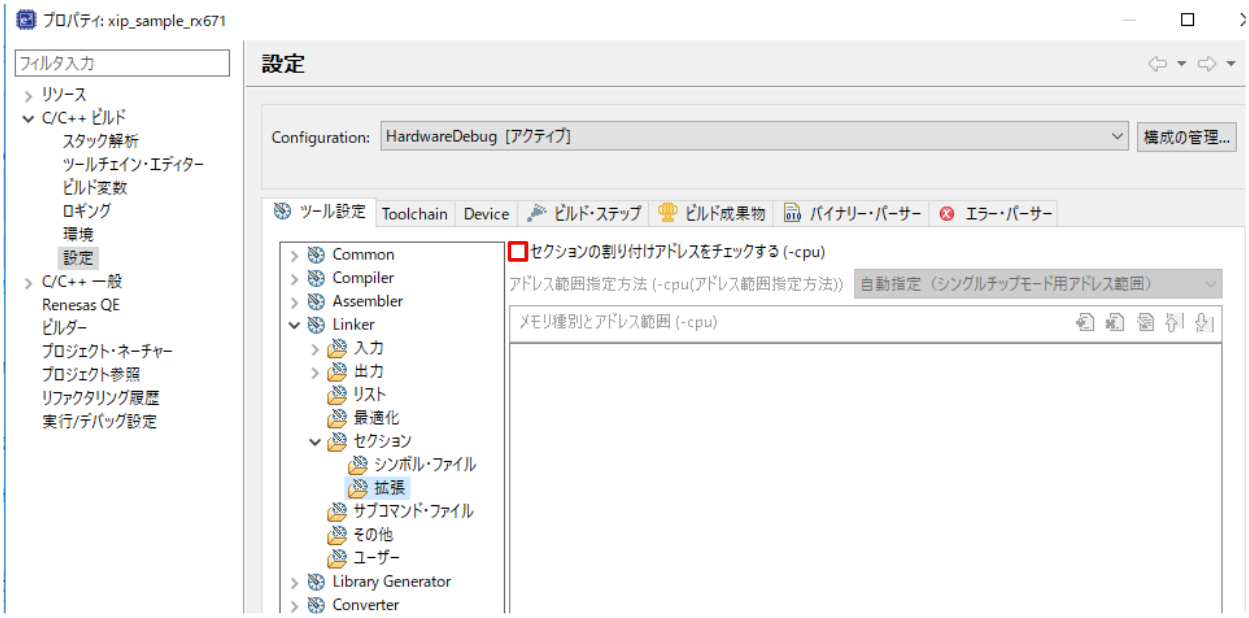


図 13　セクションの割り付けアドレスチェック(-cpu)オプション設定

なお、「セクションの割り付けアドレスをチェックする(-cpu)」のチェックボックスにチェックし、-cpu オプションに QSPI 領域を追加することもできます。

-cpu オプションの設定方法は、「CC-RX コンパイラ ユーザーズマニュアル(R20UT3248)」をご参照ください。

3.1.1.3　概略フロー

図 14 にアプリケーションプログラム（内蔵 ROM 上で実行）の概略フローを示します。

図 15 にアプリケーションプログラム（シリアル ROM 上で実行）の概略フローを示します。

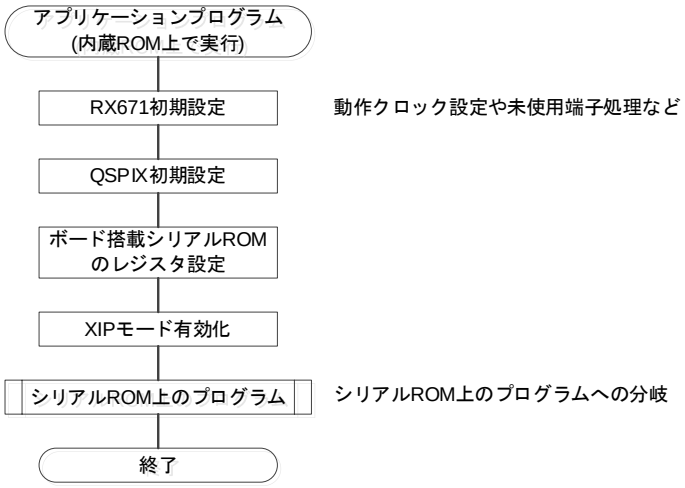
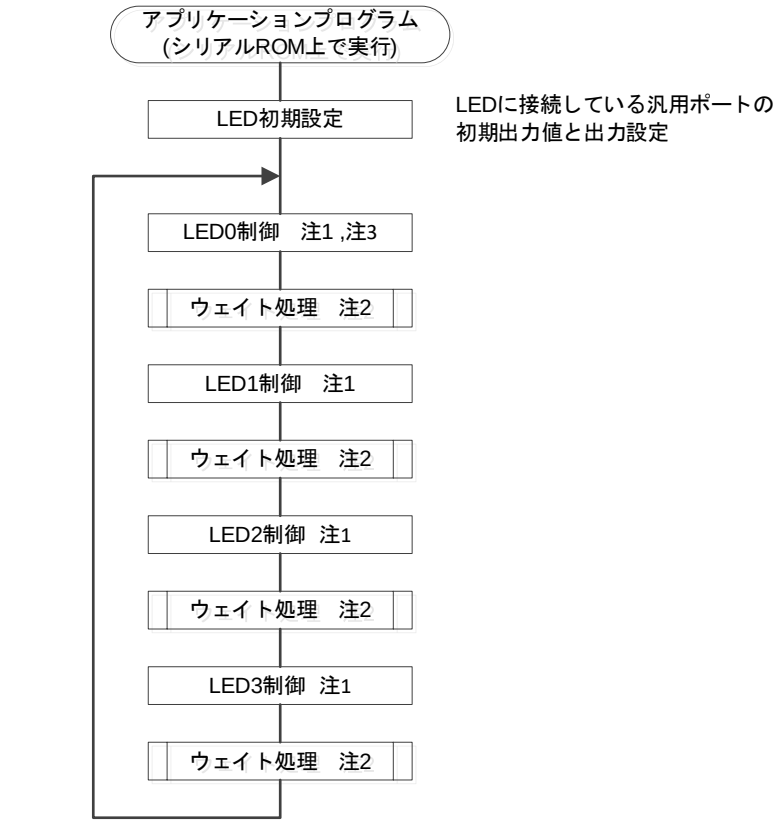


図 14　アプリケーションプログラム（内蔵 ROM 上で実行）の概略フロー



注1 : ループする度にLEDの点灯/消灯を切り替えます。
注2 ： ウェイト処理はソフトウェアループとし、ウェイト時間は0.5秒程度。
注3 : EK-RX671ボード向けサンプルプログラムでは実施しません

図 15　アプリケーションプログラム（シリアル ROM 上で実行）の概略フロー

3.1.2　プログラムの構成

3.1.2.1　ファイル構成

アプリケーションプログラムで使用するファイルを示します。なお、FIT モジュールおよび SC で自動生成されるファイルは除きます。

表 3.3 アプリケーションプログラムで使用するファイル

ファイル名	概要
main.c	本アプリケーションプログラムのメイン処理。 QSPIX、シリアル ROM のステータスレジスタの初期化、XIP モードへの遷移、シリアル ROM 上のプログラムへの分岐を行います。
main_serial_rom.c	シリアル ROM 上に配置するプログラム
serial_rom.h	シリアル ROM 制御コマンド定義

3.1.2.2　オプション設定メモリ

アプリケーションプログラムで使用するオプション設定メモリの設定を示します。

表 3.4 アプリケーションプログラムで使用するオプション設定メモリ

シンボル	アドレス	設定値	内容
MDE	FE7F 5D00h～FE7F 5D03h	FFFF FFFFh	リトルエンディアン

3.1.2.3　定数一覧

アプリケーションプログラムで使用する定数を示します。

表 3.5 アプリケーションプログラムで使用する定数(RSK ボード向け)

定数名	設定値	内容
LED_ON	(0)	LED 点灯
LED_OFF	(1)	LED 消灯
LED0	PORT1.PODR.BIT.B7	LED0 のポート出力データ格納ビット
LED1	PORTF.PODR.BIT.B5	LED1 のポート出力データ格納ビット
LED2	PORT0.PODR.BIT.B3	LED2 のポート出力データ格納ビット
LED3	PORT0.PODR.BIT.B5	LED3 のポート出力データ格納ビット
LED0_PDR	PORT1.PDR.BIT.B7	LED0 のポート方向制御ビット
LED1_PDR	PORTF.PDR.BIT.B5	LED1 のポート方向制御ビット
LED2_PDR	PORT0.PDR.BIT.B3	LED2 のポート方向制御ビット
LED3_PDR	PORT0.PDR.BIT.B5	LED3 のポート方向制御ビット
LED_INTERVAL	(0x16000)	LED 点灯間隔を 0.5 秒に設定
CMD_WREN	(0x06)	シリアル ROM への Write Enable (WREN)コマンド
CMD_WRSR	(0x01)	シリアル ROM への Write Status Register (WRSR)コマンド
CMD_RDSR	(0x05)	シリアル ROM への Read Status Register (RDSR)コマンド
SERIALROM_ENTER_QSPI_MODE	(0x40)	シリアル ROM の Status Register 設定データ(Quad mode 有効設定)
SERIALROM_CONFIG_REG	(0x00)	シリアル ROM の Configuration Register 設定データ

表 3.6 アプリケーションプログラムで使用する定数(EK-RX671 向け)

定数名	設定値	内容
LED_ON	(1)	LED 点灯
LED_OFF	(0)	LED 消灯
LED1	PORT5.PODR.BIT.B6	LED1 のポート出力データ格納ビット
LED2	PORT8.PODR.BIT.B2	LED2 のポート出力データ格納ビット
LED3	PORT2.PODR.BIT.B5	LED3 のポート出力データ格納ビット
LED1_PDR	PORT5.PDR.BIT.B6	LED1 のポート方向制御ビット
LED2_PDR	PORT8.PDR.BIT.B2	LED2 のポート方向制御ビット
LED3_PDR	PORT2.PDR.BIT.B5	LED3 のポート方向制御ビット
LED_INTERVAL	(0x16000)	LED 点灯間隔を 0.5 秒に設定
CMD_WREN	(0x06)	シリアル ROM への Write Enable (WREN)コマンド
CMD_WRSR1	(0x01)	シリアル ROM への Write Status Register (WRSR)コマンド
CMD_WRSR2	(0x31)	シリアル ROM への Write Status Register (WRSR)コマンド
CMD_RDSR1	(0x05)	シリアル ROM への Read Status Register (RDSR)コマンド
CMD_RDSR2	(0x35)	シリアル ROM への Read Status Register (RDSR)コマンド
SERIALROM_ENTER_QSPI_MODE	(0x02)	シリアル ROM の Status Register 設定データ(Quad mode 有効設定)
SERIALROM_CONFIG_REG	(0x00)	シリアル ROM の Configuration Register 設定データ

3.1.2.4　関数一覧

アプリケーションプログラムの関数一覧を示します。

表 3.7 アプリケーションプログラムの関数一覧

関数名	概要
main	メイン処理 QSPIX、シリアル ROM のステータスレジスタの初期化、XIP モードへの遷移、シリアル ROM 上のプログラムへの分岐
rom_access_error_callback	QSPIX FIT モジュールのコールバック関数 ROM アクセスエラー割り込み発生時の確認
main_serial_rom	シリアル ROM に配置するプログラム ボード搭載の LED を順次点灯させる
led_wait	LED 点灯間隔を確保するためのソフトウェアウェイト処理(ウェイト時間は約 0.5 秒)

3.2　ライタ用プログラム

本アプリケーションノートでは、動作確認用にシリアル ROM 向けの簡易的なライタ用プログラムを 2 つ用意しています。

1 つはアプリケーションプログラムの一部を内蔵 ROM に取り込みシリアル ROM へ書き込むプログラムです(ライタ用プログラム 1)。

もう 1 つはアプリケーションプログラムの一部をホスト PC からシリアル通信で受信しシリアル ROM へ書き込むプログラムです(ライタ用プログラム 2)。

3.2.1　ライタ用プログラム 1

3.2.1.1　プログラムの仕様

(1)　ソフトウェアの説明

図 16 にライタ用プログラム 1 のバイナリファイル入力とアドレス配置を示します。本図に示すようにライタ用プログラム 1 は、アプリケーションプログラムで分割出力したバイナリファイル(SerialROM_block.bin)を入力します。そして、入力したバイナリファイルを ROM データとして内蔵 ROM の任意のアドレスに配置します(本ライタ用プログラム 1 では、内蔵 ROM 先頭アドレスに配置しています)。

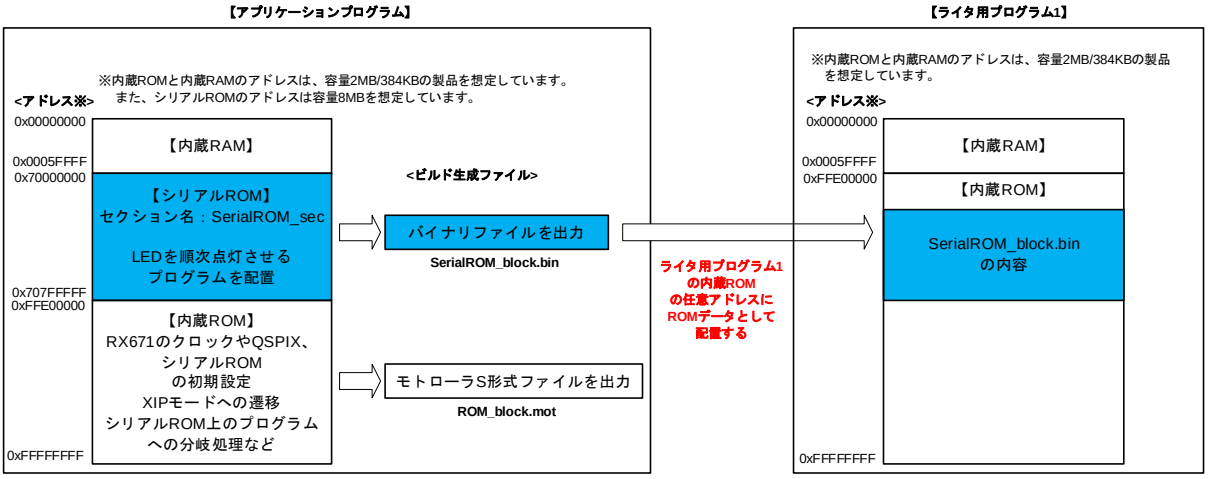


図 16　ライタ用プログラム 1 のバイナリファイル入力とアドレス配置

ライタ用プログラム 1 では、ボード搭載の LED を使用して、シリアル ROM への書き込み状態を確認することができます。
搭載されている LED 個数が異なるため、RSK ボード向けと EK ボード向けサンプルプログラムでは書き込み状態を表す LED 状態が異なります。

表 3.8 にシリアル ROM の書き込み状態表示を示します。

表 3.8 シリアル ROM の書き込み状態表示

書き込み状態		RSK ボード	EK ボード
イレーズ完了	OK	LED0 点灯	LED1 点灯
	NG	LED3 点灯	LED1 点滅（周期：1Hz）
書き込み完了	OK	LED1 点灯	LED2 点灯
	NG	LED3 点灯	LED2 点滅（周期：1Hz）
ベリファイ	OK	LED2 点灯	LED3 点灯
	NG	LED3 点灯	LED3 点滅（周期：1Hz）

(2)　e² studio でのビルド設定

e² studio で必要なオプション設定について説明します。

(a)　バイナリデータを配置するアドレスへのセクション割り当て

入力したバイナリファイルを ROM データとして内蔵 ROM の任意のアドレスに配置するための準備として、まず、バイナリデータを配置するアドレスにセクションを割り当てます。

プロジェクトのプロパティを開き、「C/C++ビルド」→「設定」をクリックし、右の表示タブから「ツール設定」を選択します。そこから、「Linker」→「セクション」を選択し、図 17 の入力したバイナリファイルの配置アドレスへのセクション割り当て(1/2)の画面を表示します。

「セクション(-start)」の右にある[…]ボタンをクリックします。

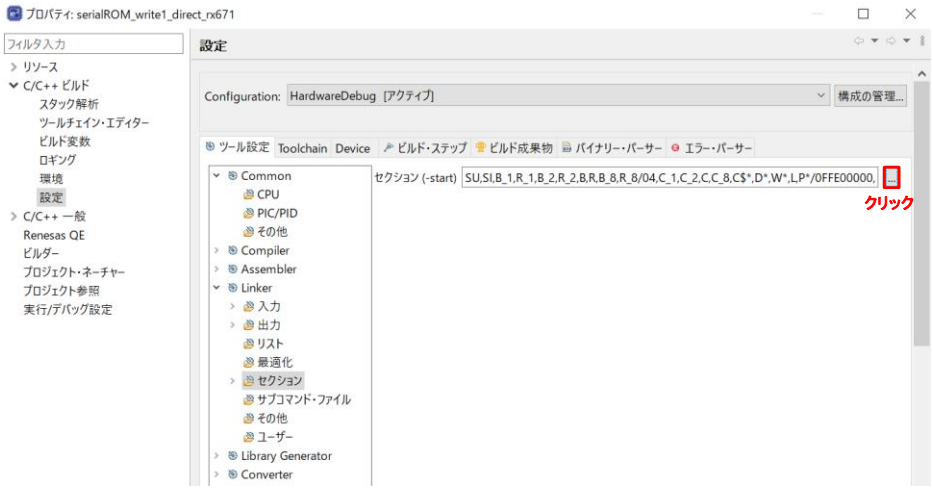


図 17　入力したバイナリファイルの配置アドレスへのセクション割り当て(1/2)

次に、図 18 に示すように、「セクション・ビューアー」で「セクションの追加」ボタンをクリックし、内蔵 ROM 上の任意のアドレスにセクションを追加します。本ライタ用プログラム 1 では、

配置アドレス：0xFFE00000(内蔵 ROM 先頭アドレス)

セクション名：SerialROM_WriteData_sec

としています。

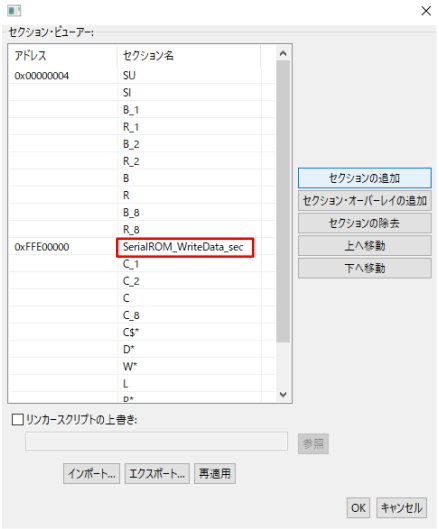


図 18　入力したバイナリファイルの配置アドレスへのセクション割り当て(2/2)

(b)　入力バイナリファイル指定(-binary)オプション設定

-binary オプションを使用して、アプリケーションプログラムで生成したバイナリファイルを入力します。

プロジェクトのプロパティを開き、「C/C++ビルド」→「設定」をクリックし、右の表示タブから「ツール設定」を選択します。そこから、「Linker」→「ユーザー」を選択し、図 19 のバイナリファイルの入力設定の画面を表示します。

「追加するオプション(すべての指定オプションの後ろに追加)」の追加ボタンをクリックし、-binary オプションの設定を追加します。

本ライタ用プログラム 1 では、以下を設定しています。

-binary="${WorkspaceDirPath}/xip_sample_rx671/HardwareDebug/SerialROM_block.bin"

(SerialROM_WriteData_sec:4/DATA)

注 1. 本ドキュメントでは内容説明のため、上記のようにオプション記述の途中で改行していますが、実際のオプション設定では改行していません。

注 2. 上記の記述は RSK ボード向けのファイルパス記述です。EK ボード使用時は、プロジェクト名を EK ボード向けプロジェクト名に置き換える必要があります。

本設定により、入力したバイナリファイルは以下のように配置されます。

配置先セクション：SeriarlROM_WriteData_sec セクション(アライメント数 4)

配置先セクション属性：DATA

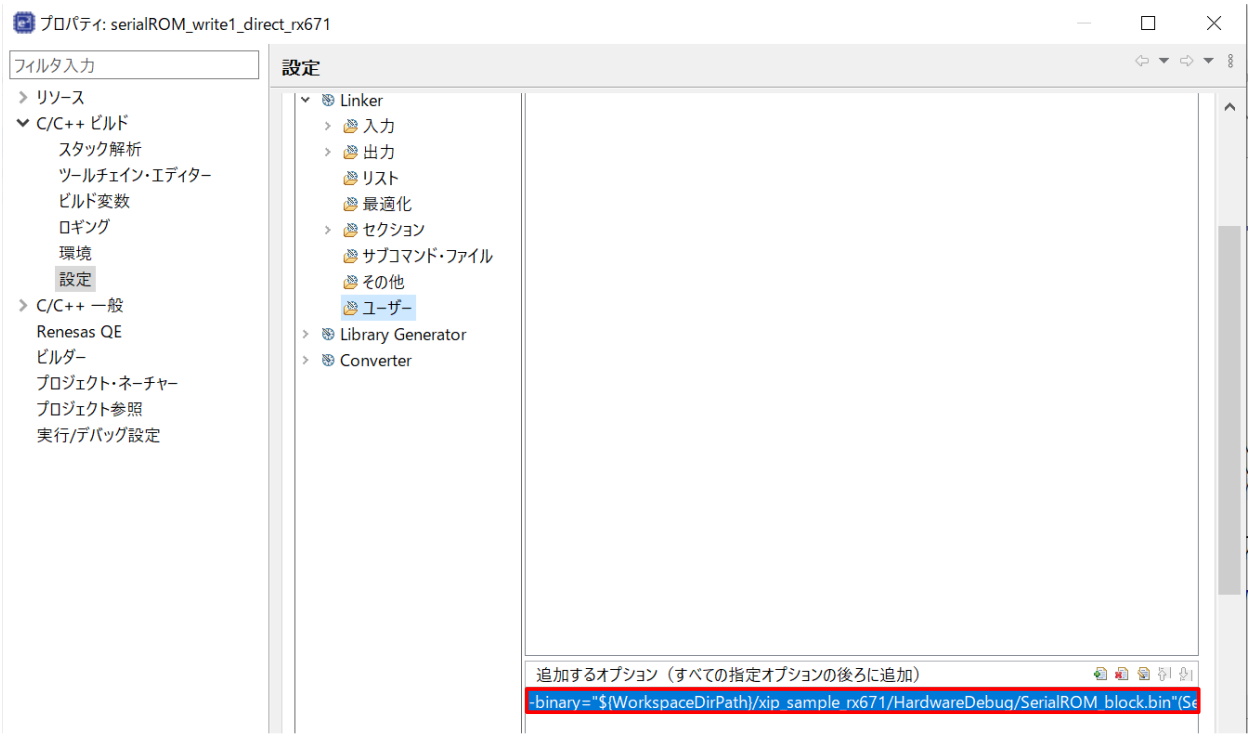


図 19 バイナリファイルの入力設定

(3)　概略フロー

図 20 にライタ用プログラム 1(RSK ボード向け)の概略フローを、図 21 にライタ用プログラム 1(EK ボード向け)の概略フローを示します。

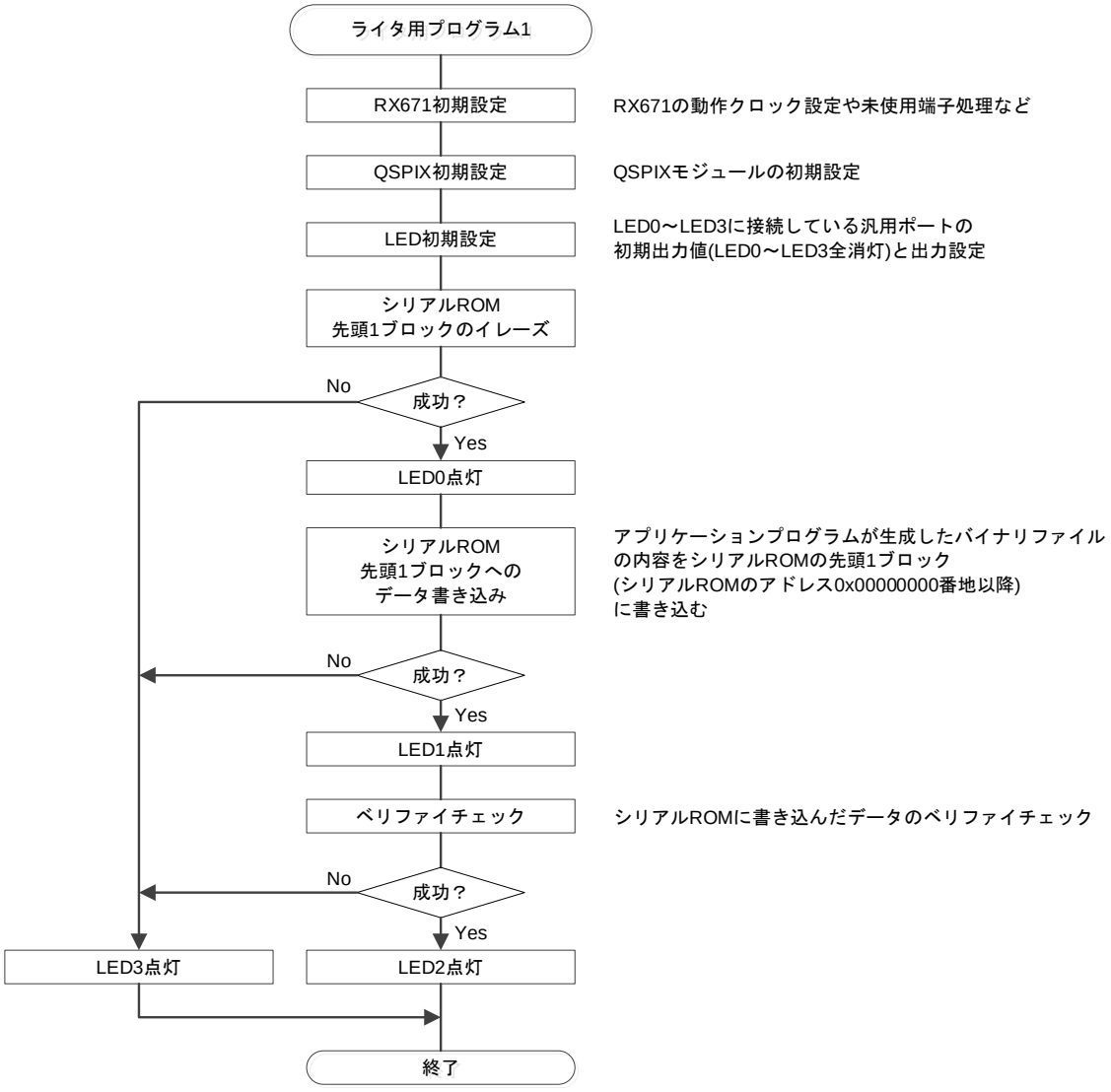


図 20　ライタ用プログラム 1(RSK ボード向け)の概略フロー

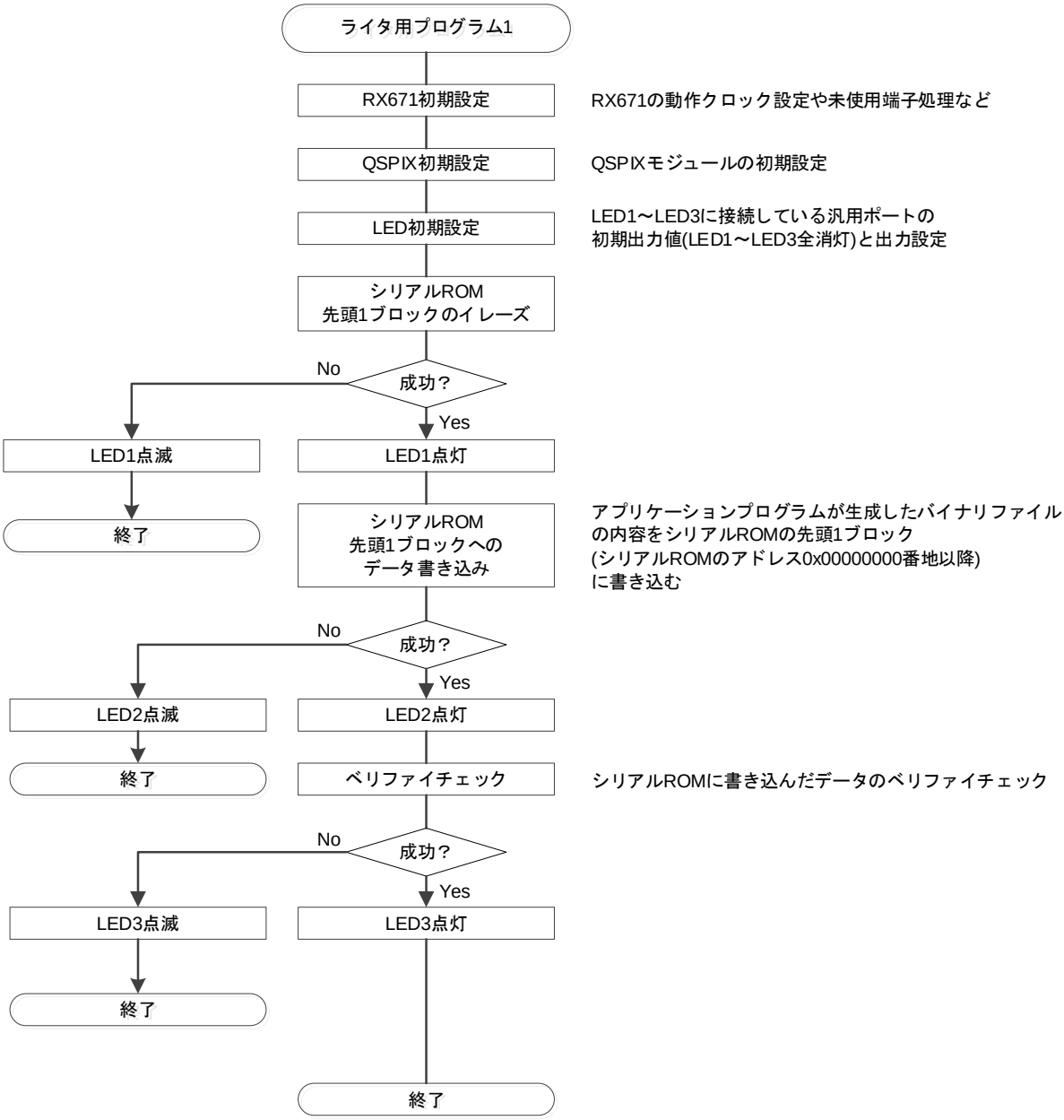


図 21　ライタ用プログラム 1(EK ボード向け)の概略フロー

3.2.1.2　プログラムの構成

(1)　ファイル構成

ライタ用プログラム 1 で使用するファイルを示します。なお、FIT モジュールおよび SC で自動生成されるファイルは除きます。

表 3.9 ライタ用プログラム 1 で使用するファイル

ファイル名	概要
serial_rom_write1_direct_rx671.c	ライタ用プログラム 1 のメイン処理。 QSPIX、シリアル ROM のステータスレジスタの初期化、シリアル ROM のブロックイレーズ、書き込み、ベリファイを行います。
serial_rom.h	シリアル ROM 制御コマンド定義

(2)　オプション設定メモリ

ライタ用プログラム 1 で使用するオプション設定メモリの設定を示します。

表 3.10 ライタ用プログラム 1 で使用するオプション設定メモリ

シンボル	アドレス	設定値	内容
MDE	FE7F 5D00h～FE7F 5D03h	FFFF FFFFh	リトルエンディアン

(3)　定数一覧

ライタ用プログラム 1 で使用する定数を示します。

表 3.11 ライタ用プログラム 1 で使用する定数(RSK ボード向け)

定数名	設定値	内容
LED_ON	(0)	LED 点灯
LED_OFF	(1)	LED 消灯
LED0	PORT1.PODR.BIT.B7	LED0 のポート出力データ格納ビット
LED1	PORTF.PODR.BIT.B5	LED1 のポート出力データ格納ビット
LED2	PORT0.PODR.BIT.B3	LED2 のポート出力データ格納ビット
LED3	PORT0.PODR.BIT.B5	LED3 のポート出力データ格納ビット
LED0_PDR	PORT1.PDR.BIT.B7	LED0 のポート方向制御ビット
LED1_PDR	PORTF.PDR.BIT.B5	LED1 のポート方向制御ビット
LED2_PDR	PORT0.PDR.BIT.B3	LED2 のポート方向制御ビット
LED3_PDR	PORT0.PDR.BIT.B5	LED3 のポート方向制御ビット
CMD_WREN	(0x06)	シリアル ROM への Write Enable (WREN)コマンド
CMD_WRSR	(0x01)	シリアル ROM への Write Status Register (WRSR)コマンド
CMD_RDSR	(0x05)	シリアル ROM への Read Status Register (RDSR)コマンド
CMD_RDSCUR	(0x2B)	シリアル ROM への Read Security Register (RDSCUR)コマンド
CMD_BE	(0x52)	シリアル ROM への Block Erase (BE)コマンド
CMD_PP	(0x02)	シリアル ROM への Page Program (PP)コマンド
SERIALROM_EXIT_QSPI_MODE	(0x00)	シリアル ROM の Status Register 設定データ (Quad mode 無効設定)
SERIALROM_CONFIG_REG	(0x00)	シリアル ROM の Configuration Register 設定データ

表 3.12 ライタ用プログラム 1 で使用する定数(EK ボード向け)

定数名	設定値	内容
LED_ON	(0)	LED 点灯
LED_OFF	(1)	LED 消灯
LED0	PORT1.PODR.BIT.B7	LED0 のポート出力データ格納ビット
LED1	PORTF.PODR.BIT.B5	LED1 のポート出力データ格納ビット
LED2	PORT0.PODR.BIT.B3	LED2 のポート出力データ格納ビット
LED0_PDR	PORT1.PDR.BIT.B7	LED0 のポート方向制御ビット
LED1_PDR	PORTF.PDR.BIT.B5	LED1 のポート方向制御ビット
LED2_PDR	PORT0.PDR.BIT.B3	LED2 のポート方向制御ビット
CMD_WREN	(0x06)	シリアル ROM への Write Enable (WREN)コマンド
CMD_WRSR	(0x01)	シリアル ROM への Write Status Register (WRSR)コマンド

定数名	設定値	内容
CMD_RDSR	(0x05)	シリアル ROM への Read Status Register (RDSR)コマンド
CMD_RDSCUR	(0x2B)	シリアル ROM への Read Security Register (RDSCUR)コマンド
CMD_BE	(0x52)	シリアル ROM への Block Erase (BE)コマンド
CMD_PP	(0x02)	シリアル ROM への Page Program (PP)コマンド
SERIALROM_EXIT_QSPI_MODE	(0x00)	シリアル ROM の Status Register 設定データ (Quad mode 無効設定)
SERIALROM_CONFIG_REG	(0x00)	シリアル ROM の Configuration Register 設定データ

(4)　関数一覧

ライタ用プログラム 1 の関数一覧を示します。

表 3.13 ライタ用プログラム 1 の関数一覧

関数名	概要
main	メイン処理 QSPIX、シリアル ROM のステータスレジスタの初期化、シリアル ROM のブロックイレーズ、プログラム、ベリファイチェック
rom_access_error_callback	QSPIX FIT モジュールのコールバック関数 ROM アクセスエラー割り込み発生時の確認
write_data_func	シリアル ROM へのデータ書き込み処理

3.2.2　ライタ用プログラム 2

3.2.2.1　プログラムの仕様

(1)　ソフトウェアの説明

ライタ用プログラム 2 は、ホスト PC のターミナルソフトウェアを使用してシリアル通信(XMODEM/SUM プロトコル)により、SerialROM_block.mot ファイルを受信し、シリアル ROM に書き込みます。

ここで、上記の SerialROM_block.mot とは、アプリケーションプログラムにおいてシリアル ROM 上に配置するプログラムをモトローラ S 形式でビルド生成した SerialROM_block.mot ファイルを示します。

詳細は「3.1.1.1(2) アプリケーションプログラムのビルド生成ファイルの分割出力」をご参照ください。

表 3.14 に RX671 とホスト PC とのシリアル通信仕様を示します。ターミナルソフトウェアの設定方法はターミナルソフトウェアのマニュアルをご参照ください。

表 3.14　RX671 とホスト PC とのシリアル通信仕様

項目	内容
通信方式	調歩同期式通信
通信プロトコル	XMODEM/SUM
ビットレート	115200 bps
データ長	8 ビット
パリティ	なし
ストップビット	1 ビット
フロー制御	なし

(2)　概略フロー

図 22 にライタ用プログラム 2 のメイン処理の概略フローを示します。

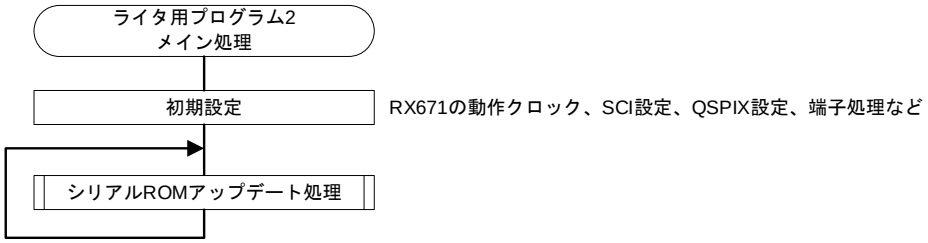
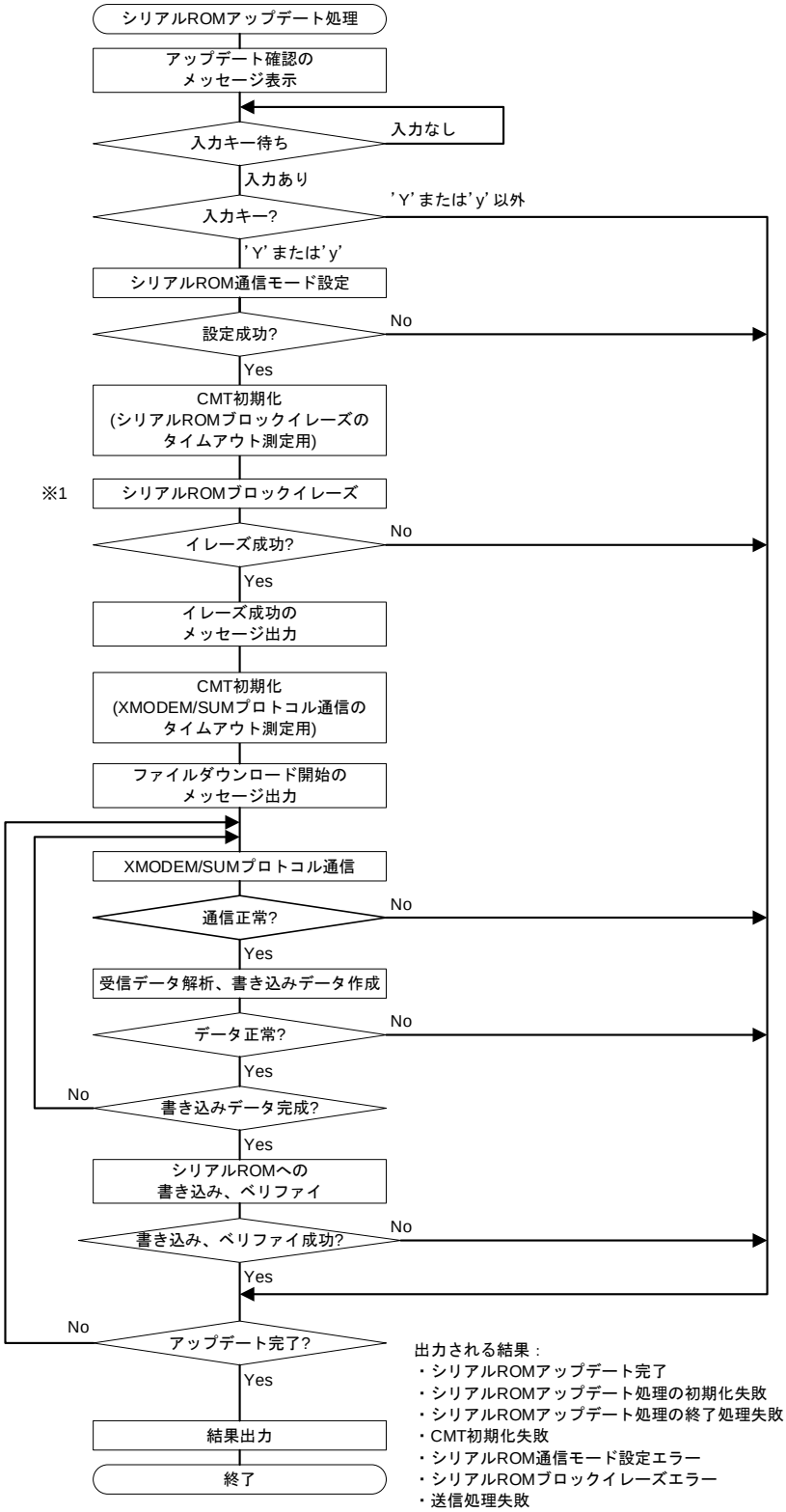


図 22　ライタ用プログラム 2 のメイン処理の概略フロー

図 23 にライタ用プログラム 2 のシリアル ROM アップデート処理の概略フローを示します。



※1
イレーズするブロック数はserial_rom_config.hファイルの定数
SEL_ERASE_BLOCK_NUM
で設定します。
シリアルROMの先頭ブロックからSEL_ERASE_BLOCK_NUM
で指定したブロック数をイレーズします。

図 23　ライタ用プログラム 2 のシリアル ROM アップデート処理の概略フロー

(3)　ターミナルソフトウェア画面出力とライタ用プログラム 2 の動作

(a)　アップデート確認

ライタ用プログラム 2 は、メイン処理で RX671 動作クロック、SCI、QSPIX の初期設定、端子処理などをした後、SCI を使用してホスト PC のターミナルソフトウェアに図 24 のメッセージを出力します。その後、ターミナルソフトウェアからの入力を待ちます。

```
RX671 Serial ROM Update ver1.00
Erase and write (Y/N)?
```

図 24 アップデート確認の画面出力

(b)　SerialROM_block.mot ファイルダウンロード開始

ライタ用プログラム 2 は、ターミナルソフトウェアから'Y'または'y'を受信すると、シリアル ROM をブロックイレーズしファイル受信待ち状態となり、図 25 のメッセージを出力します。

ターミナルソフトウェアから XMODEM/SUM プロトコルを使用して mot ファイル(アプリケーションプログラムでビルド生成した SerialROM_block.mot)を送信してください。

なお、ターミナルソフトウェアからの XMODEM/SUM プロトコルでのファイル送信方法は、ターミナルソフトウェアのマニュアルを参照してください。

```
Erasing has been done.
Start XMODEM download…
```

図 25 ファイルダウンロード開始の画面出力

(c)　シリアル ROM アップデート完了

シリアル ROM への書き込みが完了すると、図 26 のメッセージを出力します。

```
Updating has been done.
>
```

図 26 シリアル ROM アップデート完了の画面出力

(d)　エラー出力

エラーが発生した場合、その内容に応じて表 3.15 のメッセージを出力します。

表 3.15　エラー発生時のメッセージ

エラーメッセージ	内容
Initialize update error.	アップデート処理の初期化失敗
Finalize update error.	アップデート処理の終了処理失敗
CMT module error.	CMT 初期化失敗
Serial ROM mode setting error.	シリアル ROM の通信モード設定エラー
Serial ROM Erasing error.	シリアル ROM ブロックイレーズエラー
Send error	送信処理失敗
Receive error.	受信処理失敗
Timeout.	XMODEM/SUM プロトコル通信のタイムアウト
Data error.	XMODEM/SUM プロトコル通信のデータエラー
Block processing error.	データ解析エラー、シリアル ROM 書き込みエラー

(e)　アップデートキャンセル

「(a) アップデート確認」において、ライタ用プログラム 2 は、'Y'または'y'以外のコマンドを受信すると、図 27 メッセージを出力しアップデートをキャンセルします。

```
Command canceled.
>
```

図 27 アップデートキャンセルの画面出力

3.2.2.2　プログラムの構成

(1)　ファイル構成

ライタ用プログラム 2 で使用するファイルを示します。なお、FIT モジュールおよび SC で自動生成されるファイルは除きます。

表 3.16 ライタ用プログラム 2 で使用するファイル

ファイル名	概要
serial_rom_write2_serial_rx671.c	ライタ用プログラム 2 のメイン処理 SCI、QSPIX の初期化、アップデート確認メッセージ出力、アップデート処理の呼び出しなどを行います。
r_xmodem.c	XMODEM/SUM 通信処理
r_xmodem_if.h	XMODEM/SUM 通信処理インタフェースファイル
r_fw_up_rx.c	シリアル ROM アップデート処理
r_fw_up_rx_if.h	シリアル ROM アップデート処理インタフェースファイル
r_fw_up_rx_private.h	シリアル ROM アップデート処理ヘッダファイル
r_fw_up_buf.c	シリアル ROM アップデートデータのバッファ処理
r_fw_up_buf.h	シリアル ROM アップデートデータのバッファ処理ヘッダファイル
serial_rom.h	シリアル ROM 制御コマンド定義
serial_rom_config.h	シリアル ROM イレーズブロック数設定ファイル シリアル ROM の先頭ブロックから幾つのブロックをイレーズするかを本ファイルの定数”SEL_ERASE_BLOCK_NUM”に設定してください。

(2)　オプション設定メモリ

ライタ用プログラム 2 で使用するオプション設定メモリの設定を示します。

表 3.17 ライタ用プログラム 2 で使用するオプション設定メモリ

シンボル	アドレス	設定値	内容
MDE	FE7F 5D00h～FE7F 5D03h	FFFF FFFFh	リトルエンディアン

(3)　定数一覧

表 3.18～表 3.24 にライタ用プログラム 2 で使用する定数を示します。

表 3.18 ライタ用プログラム 2 で使用する定数(serial_rom_write2_serial_rx671.c)

定数名	設定値	内容
RECV_BYTE_SIZE	(1)	SCI FIT モジュールに要求する受信データのバイト数
SEND_BYTE_SIZE	(1)	SCI FIT モジュールに要求する送信データのバイト数
COMMAND_YES_UPPER	('Y')	入力コマンド用文字コード（“Y”）
COMMAND_YES_LOWER	('y')	入力コマンド用文字コード（“y”）
COMMAND_CR	('¥r')	入力コマンド用文字コード（改行コード）
CMT_FREQUENCY_HZ	(2)	CMT 設定周波数(XMODEM/SUM プロトコル通信のタイムアウト測定用)
STRING_MAX_SIZE	SCI_CFG_CH10_TX_BUFSIZ (RSK 向け) SCI_CFG_CH6_TX_BUFSIZ (EK 向け)	出力する文字列の最大サイズ

表 3.19 ライタ用プログラム 2 で使用する定数(r_xmodem.c)

定数名	設定値	内容
XM_SOH	(0x01)	XMODEM/SUM コントロールコード（SOH）
XM_EOT	(0x04)	XMODEM/SUM コントロールコード（EOT）
XM_ACK	(0x06)	XMODEM/SUM コントロールコード（ACK）
XM_NAK	(0x15)	XMODEM/SUM コントロールコード（NAK）
XM_CAN	(0x18)	XMODEM/SUM コントロールコード（CAN）
XM_HEADER_SIZE	(1+1+1)	XMODEM/SUM データブロックのヘッダサイズ（バイト数）
XM_DATA_SIZE	(128)	XMODEM/SUM データブロックのデータサイズ（バイト数）
XM_SUM_SIZE	(1)	XMODEM/SUM データブロックのチェックサムサイズ（バイト数）
XM_BLOCK_SIZE	(XM_HEADER_SIZE + XM_DATA_SIZE + XM_SUM_SIZE)	XMODEM/SUM データブロックサイズ（バイト数）
XM_RETRY_COUNT	(10)	XMODEM/SUM プロトコル通信タイムアウト判定のためのリトライ回数
UINT8T_0	(0)	uint8_t 型の 0
UINT8T_1	(1)	uint8_t 型の 1

表 3.20 ライタ用プログラム 2 で使用する定数(r_fw_up_rx.c)

定数名	設定値	内容
FW_UP_FIRM_EN_8MB_ADDRESS	(0x707FFFFF)	QSPI 領域の上位 8M バイトの最終アドレス
CMT_FOR_ERASE_FREQUENCY_HZ	(2)	CMT 設定周波数(シリアル ROM ブロックイレーズのタイムアウト測定用)

表 3.21 ライタ用プログラム 2 で使用する定数(r_fw_up_rx_private.h)

定数名	設定値	内容
FW_UP_BINARY_BUF_SIZE	(256)	シリアル ROM 書き込み用データのバッファサイズ
FW_UP_BINARY_BUF_NUM	(2)	シリアル ROM 書き込み用データのバッファ数
FW_UP_BUF_NUM	(60)	モトローラ S レコードデータバッファ数(モトローラ S フォーマットのレコードを解析し各フィールドの情報を格納したバッファの数)

表 3.22 ライタ用プログラム 2 で使用する定数(r_fw_up_buf.h)

定数名	設定値	内容
MOT_S_CHECK_SUM_FIELD	(0x02)	モトローラ S フォーマットのチェックサムフィールドの文字数
ADDRESS_LENGTH_S1	(0x04)	モトローラ S フォーマットのアドレスフィールドの文字数（S1 タイプ）
ADDRESS_LENGTH_S2	(0x06)	モトローラ S フォーマットのアドレスフィールドの文字数（S2 タイプ）
ADDRESS_LENGTH_S3	(0x08)	モトローラ S フォーマットのアドレスフィールドの文字数（S3 タイプ）
BUF_LOCK	(1)	モトローラ S レコードデータバッファをロックするための値
BUF_UNLOCK	(0)	モトローラ S レコードデータバッファを開放するための値

表 3.23 ライタ用プログラム 2 で使用する定数(serial_rom.h)

定数名	設定値	内容
CMD_WREN	(0x06)	シリアル ROM への Write Enable (WREN)コマンド
CMD_WRSR(注 1)	(0x01)	シリアル ROM への Write Status Register (WRSR)コマンド
CMD_WRSR1(注 2)	(0x01)	シリアル ROM への Write Status Register (WRSR)コマンド
CMD_WRSR2(注 2)	(0x31)	シリアル ROM への Write Status Register (WRSR)コマンド
CMD_RDSR(注 1)	(0x05)	シリアル ROM への Read Status Register (RDSR)コマンド
CMD_RDSR1(注 2)	(0x05)	シリアル ROM への Read Status Register (RDSR)コマンド
CMD_RDSR2(注 2)	(0x35)	シリアル ROM への Read Status Register (RDSR)コマンド
CMD__RDSCUR(注 1)	(0x2B)	シリアル ROM への Read Security Register (RDSCUR)コマンド
CMD_BE	(0x52)	シリアル ROM への Block Erase (BE)コマンド
CMD_PP	(0x02)	シリアル ROM への Page Program (PP)コマンド
SERIALROM_EXIT_QSPI_MODE	(0x00)	シリアル ROM の Status Register 設定データ (Quad mode 無効設定)
SERIALROM_CONFIG_REG(注 1)	(0x00)	シリアル ROM の Configuration Register 設定データ

注 1. EK ボード向けプロジェクトには存在しません。

注 2. RSK ボード向けプロジェクトには存在しません。

表 3.24 ライタ用プログラム 2 で使用する定数(serial_rom_config.h)

定数名	設定値	内容
SEL_ERASE_BLOCK_NUM	(1)	シリアル ROM ブロックイレーズ数

SEL_ERASE_BLOCK_NUM はユーザ設定が可能です。

ライタ用プログラム 2 はシリアル ROM の先頭ブロックからイレーズします。

このため、先頭ブロックから幾つのブロックをイレーズするかを SEL_ERASE_BLOCK_NUM に設定してください。デフォルトの設定値は“1”です。

SEL_ERASE_BLOCK_NUM に設定できる値は、“1”～“128”の範囲です。

(4)　型定義一覧

図 28～図 31 にライタ用プログラム 2 で使用する型定義を示します。

```
typedef enum e_xmodem_proc_stage
{
	XMODEM_PROC_END = 0,
	XMODEM_PROCESSING,
	XMODEM_SOH_RECEIVED
} e_xmodem_proc_stage_t;

typedef struct st_xmodem_states
{
	uint8_t retry_counter;
	uint8_t expected_block_number;
	uint8_t recv_buf_index;
	uint8_t can_counter;
	uint8_t *precv_buf;
	e_xmodem_proc_stage_t proc_stage;
	xm_recv_func_t recv_func;
	xm_send_func_t send_func;
	xm_exec_func_t exec_func;
} st_xmodem_states_t;
```

図 28 ライタ用プログラム 2 で使用する型定義(r_xmodem.c)

```
typedef enum e_xmodem_err
{
	XMODEM_SUCCESS,
	XMODEM_SEND_ERR,
	XMODEM_RECV_ERR,
	XMODEM_TIMEOUT,
	XMODEM_PROC_BLOCK_ERR,
	XMODEM_RECV_CAN,
	XMODEM_DATA_ERR
} e_xmodem_err_t;

typedef e_xmodem_err_t (*xm_recv_func_t)(uint8_t* p_arg);
typedef e_xmodem_err_t (*xm_send_func_t)(uint8_t arg);
typedef e_xmodem_err_t (*xm_exec_func_t)(const uint8_t* p_buf, uint16_t size);
```

図 29 ライタ用プログラム 2 で使用する型定義(r_xmodem_if.h)

```
typedef enum e_fw_up_return_t
{
	FW_UP_SUCCESS,
	FW_UP_ERR_OPENED,
	FW_UP_ERR_NOT_OPEN,
	FW_UP_ERR_NULL_PTR,
	FW_UP_ERR_INVALID_RECORD,
	FW_UP_ERR_BUF_FULL,
	FW_UP_ERR_BUF_EMPTY,
	FW_UP_ERR_INITIALIZE,
	FW_UP_ERR_ERASE,
	FW_UP_ERR_CMT_FOR_ERASE,
	FW_UP_ERR_WRITE,
	FW_UP_ERR_VERIFY,
	FW_UP_ERR_INVALID_ADDRESS,
	FW_UP_ERR_INVALID_WRITE_SIZE,
	FW_UP_ERR_INTERNAL
} fw_up_return_t;

typedef struct st_fw_up_fl_data_t
{
	uint32_t src_addr;
	uint32_t dst_addr;
	uint32_t len;
	uint16_t count;
} fw_up_fl_data_t;
```

図 30 ライタ用プログラム 2 で使用する型定義(r_fw_up_rx_if.h)

```
typedef enum fw_up_mot_s_cnt_t
{
	STATE_MOT_S_RECORD_MARK = 0,
	STATE_MOT_S_RECORD_TYPE,
	STATE_MOT_S_LENGTH_1,
	STATE_MOT_S_LENGTH_2,
	STATE_MOT_S_ADDRESS,
	STATE_MOT_S_DATA,
	STATE_MOT_S_CHKSUM_1,
	STATE_MOT_S_CHKSUM_2
} fw_up_mot_s_cnt_t;

typedef struct MotSBufS
{
	uint8_t addr_length;
	uint8_t data_length;
	uint8_t *paddress;
	uint8_t *pdata;
	uint8_t type;
	uint8_t act;
	struct MotSBufS *pnext;
} fw_up_mot_s_buf_t;

typedef struct WriteDataS
{
	uint32_t addr;
	uint32_t len;
	uint8_t data[FW_UP_BINARY_BUF_SIZE];
	struct WriteDataS *pnext;
	struct WriteDataS *pprev;
} fw_up_write_data_t;
```

図 31 ライタ用プログラム 2 で使用する型定義(r_fw_up_buf.h)

(5)　変数一覧

表 3.25～表 3.28 にライタ用プログラム 2 で使用する static 型変数を示します。

表 3.29 にライタ用プログラム 2 で使用する const 型変数を示します。

表 3.25 ライタ用プログラム 2 で使用する static 型変数（serial_rom_write2_serial_rx671.c）

型	変数名	内容	使用関数
static sci_hdl_t	s_sci_handle	SCI モジュール制御ハンドル	main send_string_sci recv_byte_xm send_byte_xm update_serial_rom exec_firmware
static volatile bool	s_sci_send_end_flag	SCI 送信完了判定用フラグ	sci_callback send_string_sci
static volatile int32_t	s_timeout_count	XMODEM/SUM プロトコル通信のタイムアウト判定用カウンタ	cmt_callback recv_byte_xm
static volatile bool	s_timeout_flag	XMODEM/SUM プロトコル通信のタイムアウト検出フラグ	cmt_callback recv_byte_xm
static volatile bool	s_start_timer_flag	XMODEM/SUM プロトコル通信のタイムアウト判定開始フラグ	cmt_callback recv_byte_xm

表 3.26 ライタ用プログラム 2 で使用する static 型変数（r_xmodem.c）

型	変数名	内容	使用関数
static uint8_t	recv_buf[XM_BLOCK_SIZE]	XMODEM/SUM プロトコル受信データ用バッファ	exec_xmodem

表 3.27 ライタ用プログラム 2 で使用する static 型変数（r_fw_up_rx.c）

型	変数名	内容	使用関数
static bool	is_opened	シリアル ROM アップデート初期設定完了フラグ	fw_up_open fw_up_close fw_up_put_data fw_up_get_data disable_quad_mode_serial_rom erase_serial_rom write_serial_rom
Static volatile int32_t	s_timeout_count_for_erase	シリアル ROM ブロックイレーズのタイムアウト判定用カウンタ	erase_serial_rom cmt_callback_for_erase
static volatile bool	s_timeout_flag_for_erase	シリアル ROM ブロックイレーズのタイムアウト検出フラグ	erase_serial_rom cmt_callback_for_erase
static volatile bool	s_start_timer_flag_for_erase	シリアル ROM ブロックイレーズのタイムアウト判定開始フラグ	erase_serial_rom cmt_callback_for_erase

表 3.28 ライタ用プログラム 2 で使用する static 型変数（r_fw_up_buf.c）

型	変数名	内容	使用関数
static fw_up_mot_s_buf_t	mot_s_buf [FW_UP_BUF_NUM]	モトローラ S レコードデータバッファ	fw_up_buf_init fw_up_memory_init
static fw_up_mot_s_buf_t	*papp_put_mot_s_buf	モトローラ S フォーマット解析処理で現在使用しているモトローラ S レコードデータバッファへのポインタ	fw_up_buf_init fw_up_put_mot_s
static fw_up_mot_s_buf_t	*papp_get_mot_s_buf	シリアル ROM 書き込み用データ作成処理で現在使用しているモトローラ S レコードデータバッファへのポインタ	fw_up_buf_init fw_up_get_binary
static fw_up_write_data_t	write_buf [FW_UP_BINARY_BUF_NUM]	シリアル ROM 書き込み用データバッファ	fw_up_buf_init
static fw_up_write_data_t	*papp_write_buf	現在使用しているシリアル ROM 書き込み用データバッファへのポインタ	fw_up_buf_init fw_up_get_binary
static fw_up_mot_s_cnt_t	mot_s_data_state	モトローラ S フォーマットのレコードの解析状態	fw_up_buf_init fw_up_put_mot_s
static uint32_t	write_current_address	現在のシリアル ROM 書き込み先アドレス(QSPI 領域のアドレス)	fw_up_buf_init fw_up_get_binary
static bool	detect_terminal_flag	終端レコード検出フラグ	fw_up_buf_init fw_up_put_mot_s fw_up_get_binary

表 3.29 ライタ用プログラム 2 で使用する const 型変数（serial_rom_write2_serial_rx671.c）

型	変数名	内容	使用関数
static const uint8_t	s_string_menu0[]	"RX671 Serial ROM Update ver1.00¥r¥n"	update_serial_rom
static const uint8_t	s_string_update[]	"Erase and Write (Y/N)?"	update_serial_rom
static const uint8_t	s_string_erase_success[]	“Erasing has been done.¥r¥n”	update_serial_rom
static const uint8_t	s_string_download[]	“Start XMODEM download…¥r¥n”	update_serial_rom
static const uint8_t	s_string_finish_xmodem[]	“Updating has been done.¥r¥n”	update_serial_rom
static const uint8_t	s_string_cancel[]	“Command canceled.¥r¥n”	update_serial_rom
static const uint8_t	s_string_input[]	“> ”	update_serial_rom
static const uint8_t	s_string_crlf[]	“¥r¥n”	main update_serial_rom
static const_uin8_t	s_string_cmt_err[]	"CMT module error.¥r¥n"	update_serial_rom
static const uint8_t	s_string_mode_setting_err[]	"Serial ROM mode setting error.¥r¥n"	update_serial_rom
static const uint8_t	s_string_erase_err[]	“Serial ROM Erasing error.¥r¥n”	update_serial_rom
static const uint8_t	s_string_send_err[]	“Send error.¥r¥n”	update_serial_rom
static const uint8_t	s_string_recv_err[]	“Receive error.¥r¥n”	update_serial_rom
static const uint8_t	s_string_timeout[]	“Timeout.¥r¥n”	update_serial_rom
static const uint8_t	s_string_block_err[]	“Block processing error.¥r¥n”	update_serial_rom
static const uint8_t	s_string_data_err[]	“Data error.¥r¥n”	update_serial_rom
static const uint8_t	s_string_init_update_err[]	“Initialize update error.¥r¥n”	update_serial_rom
static const uint8_t	s_string_fin_update_err[]	“Finalize update error.¥r¥n”	update_serial_rom

(6)　関数一覧

表 3.30～表 3.33 にライタ用プログラム 2 の関数一覧を示します。

表 3.30 ライタ用プログラム 2 の関数一覧(serial_rom_write2_serial_rx671.c)

関数名	概要
main	メイン処理 SCI、QSPIX の初期化、シリアル ROM アップデート関数の呼び出し処理
update_serial_rom	シリアル ROM アップデート関数 ホスト PC のターミナルソフトウェアへのメッセージ出力やコマンド入力、シリアル ROM の通信モード変更関数やブロックイレーズ関数、XMODEM/SUM プロトコル通信処理関数の呼び出し処理など
send_byte_xm	XMODEM/SUM プロトコル用コールバック関数 1 バイトのデータを送信
recv_byte_xm	XMODEM/SUM プロトコル用コールバック関数 1 バイトのデータを受信
block_proc_xm	XMODEM/SUM プロトコル用コールバック用関数 1 データブロックのデータ処理
send_string_sci	文字列送信処理
rom_access_error_callback	QSPIX FIT モジュールのコールバック関数 ROM アクセスエラー割り込み発生時の確認
sci_callback	SCI FIT モジュールのコールバック関数 SCI 送信完了の確認
cmt_callback	CMT FIT モジュールコールバック関数 XMODEM/SUM プロトコル通信のタイムアウト検出

表 3.31 ライタ用プログラム 2 の関数一覧(r_modem.c)

関数名	概要
exec_xmodem	XMODEM/SUM プロトコル通信処理
xmodem_recv_soh	XMODEM/SUM プロトコルのデータブロックのヘッダ受信
xmodem_check_eot	XMODEM/SUM プロトコルのデータブロックのヘッダチェック
xmodem_recv_block	XMODEM/SUM プロトコルの 1 データブロック受信
xmodem_analyze_block	XMODEM/SUM プロトコルのデータブロック解析
xmodem_proc_data	XMODEM/SUM プロトコルの 1 データブロックのデータ処理
xmodem_send_response	XMODEM/SUM プロトコルの応答処理

表 3.32 ライタ用プログラム 2 の関数一覧(r_fw_up_rx.c)

関数名	概要
fw_up_open	シリアル ROM アップデートの初期化
fw_up_close	シリアル ROM アップデートの終了処理
analyze_and_write_data	受信データ解析関数、シリアル ROM 書き込みデータ取得関数、シリアル ROM 書き込み関数などの呼び出し処理
fw_up_put_data	受信データ解析
fw_up_get_data	シリアル ROM 書き込みデータ取得
disable_quad_mode_serial_rom	シリアル ROM の QUAD モードの無効化
erase_serial_rom	シリアル ROM ブロックイレーズ シリアル ROM 先頭ブロックから、SEL_ERASE_BLOCK_NUM で指定された数のブロックをイレーズ
write_serial_rom	シリアル ROM 書き込み
write_serial_rom_send_command	シリアル ROM 書き込みのためのコマンド送信処理
cmt_callback_for_erase	CMT FIT モジュールコールバック関数 シリアル ROM ブロックイレーズのタイムアウト検出

表 3.33 ライタ用プログラム 2 の関数一覧(r_fw_up_buf.c)

関数名	概要
fw_up_buf_init	シリアル ROM アップデートで使用するバッファの初期化
fw_up_memory_init	バッファへのポインタの初期化
fw_up_put_mot_s	モトローラ S フォーマットのレコード解析
fw_up_get_binary	シリアル ROM 書き込みデータの取得
fw_up_ascii_to_hexbyte	アスキー形式データからバイナリ形式データへの変換

3.3　使用 FIT モジュール

アプリケーションプログラム、ライタ用プログラム 1、ライタ用プログラム 2 で使用している FIT モジュール、また、各 FIT モジュールの設定を以下に示します。

3.3.1　使用 FIT モジュール一覧

表 3.34 に使用している FIT モジュール一覧を示します。

表 3.34 使用している FIT モジュール一覧

FIT モジュール	ドキュメントタイトル	アプリケーションプログラム	ライタ用プログラム 1	ライタ用プログラム 2
BSP	RX ファミリ ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)	使用	使用	使用
QSPIX	RX ファミリ QSPIX モジュール Firmware Integration Technology (R01AN5685)	使用	使用	使用
CMT	RX ファミリ CMT モジュール Firmware Integration Technology (R01AN1856)	-	-	使用
SCI	RX ファミリ SCI モジュール Firmware Integration Technology (R01AN1815)	-	-	使用
BYTEQ	RX ファミリ バイト型キューバッファ（BYTEQ）モジュール Firmware Integration Technology (R01AN1683)	-	-	使用

3.3.2　FIT モジュールの設定

使用している FIT モジュールおよび e² studio の SC の設定を下記に示します。SC の設定における各表の項目、設定内容は設定画面の表記で記載しています。各 FIT モジュールの詳細は、各 FIT モジュールのドキュメントを参照してください。

表 3.35 BSP モジュールの設定

(アプリケーションプログラム、ライタ用プログラム 1、ライタ用プログラム 2 共通)

分類	項目	設定、説明
スマート・コンフィグレータ >> コンポーネント >> r_bsp		プロパティはデフォルト設定
スマート・コンフィグレータ >> クロック		「クロック」タブはデフォルト設定
	VCC 設定	3.3(V)
	メインクロック設定	動作：チェックする 発振源：発振子 周波数：24MHz 発振安定時間：9980(us) (実際の値：10000)
	PLL 回路設定	分周比：x1 逓倍比：x10.0
	システムクロック設定	クロックソース：PLL 回路 システムクロック(ICLK)：x1/2　120 (MHz) 周辺モジュールクロック(PCLKA)：x1/2　120 (MHz) 周辺モジュールクロック(PCLKB)：x1/4　60 (MHz) 周辺モジュールクロック(PCLKC)：x1/4　60 (MHz) 周辺モジュールクロック(PCLKD)：x1/4　60 (MHz) バスクロック(BCLK)：x1/4　60 (MHz) FlashIF クロック(FCLK)：x1/4　60 (MHz)
	サブクロック発振器設定	動作：チェックする (サブクロックは使用していないがデフォルト設定とする)
	HOCO クロック設定	停止：チェックを外す
	LOCO クロック設定	停止：チェックを外す
	IWDT 専用クロック設定	停止：チェックを外す

表 3.36 QSPIX モジュールの設定

(アプリケーションプログラム、ライタ用プログラム 1、ライタ用プログラム 2 共通)

分類	項目	設定、説明
スマート・コンフィグレータ >> コンポーネント >> r_qspix_rx		下記の変更以外はデフォルト設定
	リソース >> QSPIX	QSPIX0：チェックする QSPCLK 端子：「使用する」をチェックする QSSL 端子：「使用する」をチェックする QIO0 端子：「使用する」をチェックする QIO1 端子：「使用する」をチェックする QIO2 端子：「使用する」をチェックする QIO3 端子：「使用する」をチェックする
スマート・コンフィグレータ >> 端子		以下の変更以外はデフォルト設定
	機能：QIO0	端子割り当てで PD6/D6/MTIC5V/MTIOC8A/POE4#/SSLC2-A/SDHI_D0-B/QIO0-B/IRQ6/AN101 を選択する
	機能：QIO1	端子割り当てで PD7/D7/MTIC5U/POE0#/SSLC3-A/SDHI_D1-B/QIO1-B/IRQ7/AN100 を選択する
	機能：QIO2	端子割り当てで PD2/D2/MTIOC4D/TIC2/CRX0/MISOC-A/SDHI_D2-B/QIO2-B/IRQ2/AN105 を選択する
	機能：QIO3	端子割り当てで PD3/D3/MTIOC8D/POE8#/TOC2/RSPCKC-A/SDHI_D3-B/QIO3-B/IRQ3/AN104 を選択する
	機能：QSPCLK	端子割り当てで PD5/D5/MTIC5W/MTIOC8C/POE10#/SSLC1-A/SDHI_CLK-B/QSPCLK-B/IRQ5/AN102 を選択する
	機能：QSSL	端子割り当てで PD4/D4/MTIOC8B/POE11#/SSLC0-A/SDHI_CMD-B/QSSL-B/IRQ4/AN103 を選択する

表 3.37 CMT モジュールの設定(ライタ用プログラム 2 のみ)

分類	項目	設定、説明
スマート・コンフィグレータ >> コンポーネント >> r_cmt_rx		デフォルト設定

表 3.38 SCI モジュールの設定(ライタ用プログラム 2 のみ)(RSK ボード向け)

分類	項目	設定、説明
スマート・コンフィグレータ >> コンポーネント >> r_sci_rx		下記の変更以外はデフォルト設定
	Configurations	Include software support for channel1：Not Include software support for channel10：Include Transmit end interrupt：Enable
	リソース >> SCI	SCI10：チェックする SCK10 端子：「使用する」をチェックする RXD10/SMISO10/SSCL10 端子：「使用する」をチェックする TXD10/SMOSI10/SSDA10 端子：「使用する」をチェックする
スマート・コンフィグレータ >> 端子		下記の変更以外はデフォルト設定
	機能：RXD10	端子割り当てで P86/MTIOC4D/TIOCA0/SMISO10/SSCL10/ RXD10/SMISO010/ SSCL010/RXD010/IRQ14 を選択する。
	機能：TXD10	端子割り当てで P87/MTIOC4C/TIOCA2/SMOSI10/SSDA10/TXD10/ SMOSI010/SSDA010/TXD010/SDHI_DS-C/IRQ15 を選択する。

表 3.39 SCI モジュールの設定(ライタ用プログラム 2 のみ)(EK ボード向け)

分類	項目	設定、説明
スマート・コンフィグレータ >> コンポーネント >> r_sci_rx		下記の変更以外はデフォルト設定
	Configurations	Include software support for channel1：Not Include software support for channel10：Include Transmit end interrupt：Enable
	リソース >> SCI	SCI6：チェックする SCK6 端子：「使用する」をチェックする RXD6/SMISO6/SSCL6 端子：「使用する」をチェックする TXD6/SMOSI6/SSDA6 端子：「使用する」をチェックする
スマート・コンフィグレータ >> 端子		下記の変更以外はデフォルト設定
	機能：RXD6	端子割り当てで P01/TMCI0/RXD6/SMISO6/SSCL6/IRQ9/AN110 を選択する。
	機能：TXD6	端子割り当てで P02/TMCI1/SCK6/IRQ10/AN109 を選択する。

表 3.40 BYTEQ モジュールの設定(ライタ用プログラム 2 のみ)

分類	項目	設定、説明
スマート・コンフィグレータ >> コンポーネント >> r_byteq		デフォルト設定

3.4　動作確認条件

アプリケーションプログラム、ライタ用プログラム 1、ライタ用プログラム 2 は、下記の条件で動作を確認しています。

表 3.41　動作確認条件

項目	内容
使用マイコン	R5F5671EHDFB (RX671 グループ)
動作周波数	• メインクロック: 24MHz • PLL 回路出力クロック: 240MHz • システムクロック(ICLK): 120MHz (PLL 回路出力クロック 2 分周) • 周辺モジュールクロック A(PCLKA): 120MHz (PLL 回路出力クロック 2 分周) • 周辺モジュールクロック B(PCLKB): 60MHz (PLL 回路出力クロック 4 分周) • 周辺モジュールクロック C(PCLKC): 60MHz (PLL 回路出力クロック 4 分周) • 周辺モジュールクロック D(PCLKD): 60MHz (PLL 回路出力クロック 4 分周) • バスクロック(BCLK): 60MHz (PLL 回路出力クロック 4 分周) • FlashIF クロック(FCLK): 60MHz (PLL 回路出力クロック 4 分周)
動作電圧	3.3V
統合開発環境	ルネサスエレクトロニクス製 e² studio Version 2022-04
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 V.3.04.00
	コンパイルオプション -lang = c99 その他の設定は、 「3.1.1.2 e² studio でのビルド設定」 「3.2.1.1 (2) e² studio でのビルド設定」 を参照してください。
iodefine.h のバージョン	V1.00
エンディアン	リトルエンディアン
動作モード	シングルチップモード
プロセッサモード	スーパバイザモード
サンプルプログラムのバージョン	Version 2.10
エミュレータ	E2 エミュレータ Lite
使用ボード	Renesas Starter Kit+ forRX671 (製品型名: RTK55671EHSxxxxxxx) EK-RX671 (製品型名: RTK5EK6710Sxxxxxxx)

3.5　サンプルプログラムの動作確認

図 32 にライタ用プログラム 1 を使用する場合のアプリケーションプログラムの動作確認手順を示します。

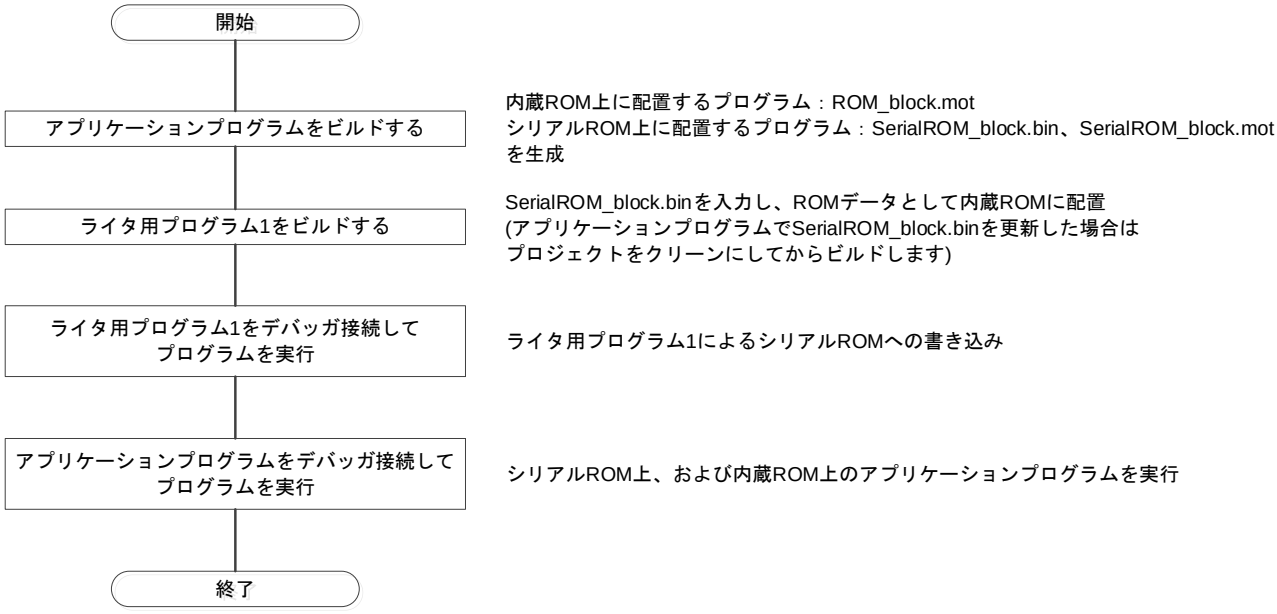


図 32 ライタ用プログラム 1 を使用する場合のアプリケーションプログラムの動作確認手順

図 33 にライタ用プログラム 2 を使用する場合のアプリケーショングラムの動作確認手順を示します。

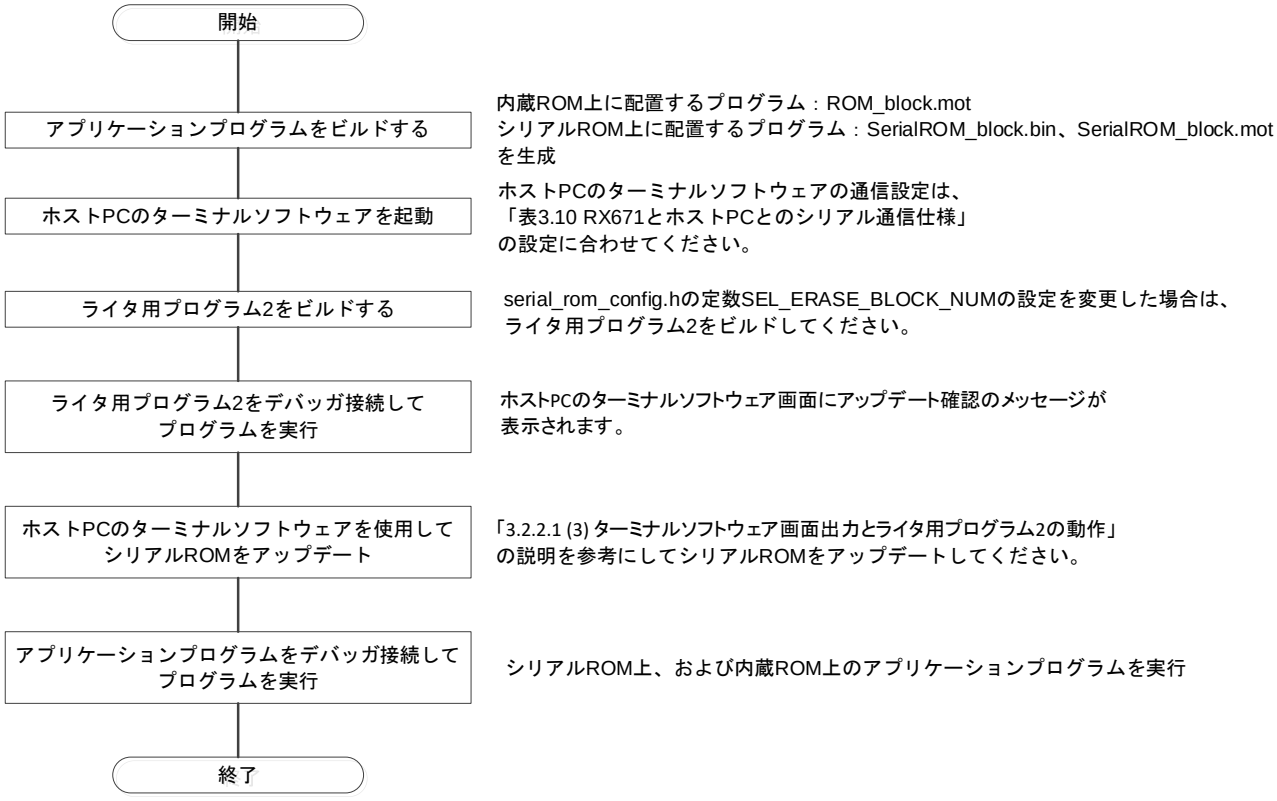


図 33 ライタ用プログラム 2 を使用する場合のアプリケーショングラムの動作確認手順

3.5.1　アプリケーションプログラムのデバッガ接続設定

e² studio を使用してアプリケーションプログラムをデバッガ接続するための設定について説明します。

図 34～図 39 にアプリケーションプログラムのデバッガ接続設定を示します。

「実行(R)」メニューから「デバッグの構成(B)...」を選択して、「デバッグ構成」ダイアログボックスを表示します。

「Renesas GDB Hardware Debugging」から「xip_sample_rx671 HardwareDebug」を選択し(図の①)、

「Startup」タブを選択し(図の②)、「追加...」ボタンをクリックします(図の③)。

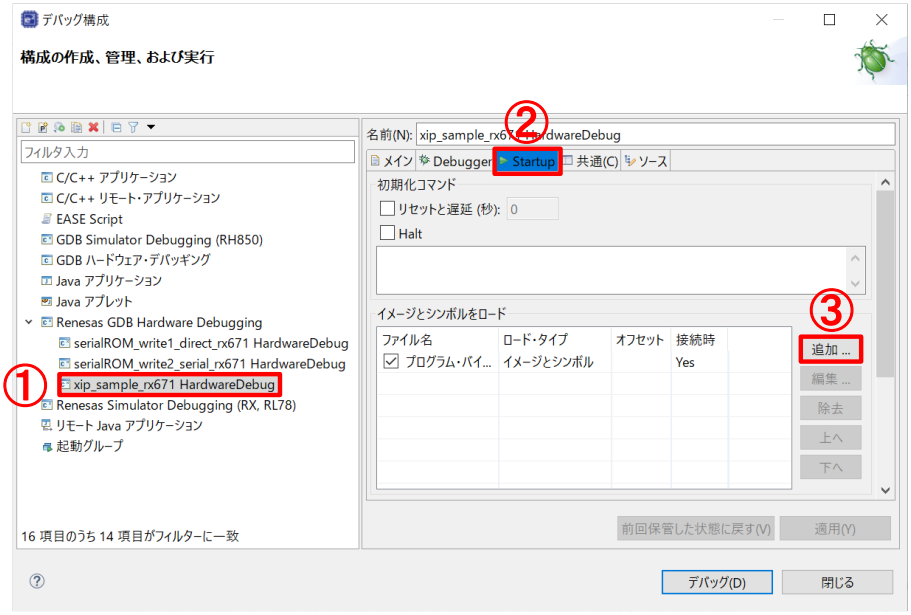


図 34　アプリケーションプログラムのデバッガ接続設定(1/6)

「ダウンロードモジュールの追加」ダイアログボックスが表示されたら「ワークスペース...」ボタンをクリックします。

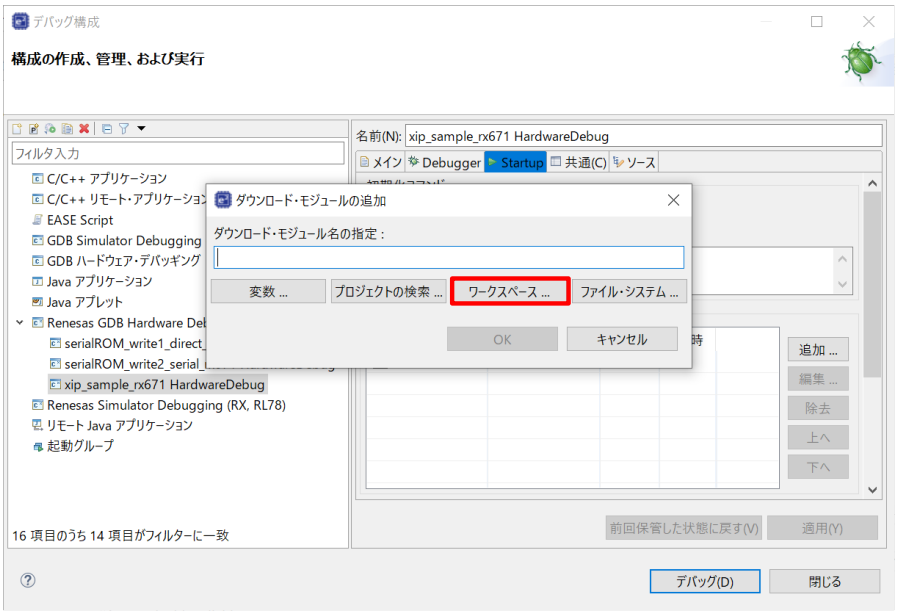


図 35　アプリケーションプログラムのデバッガ接続設定(2/6)

「xip_sample_rx671」→「HardwareDebug」→「ROM_block.mot」を選択し、「OK」ボタンをクリックします。

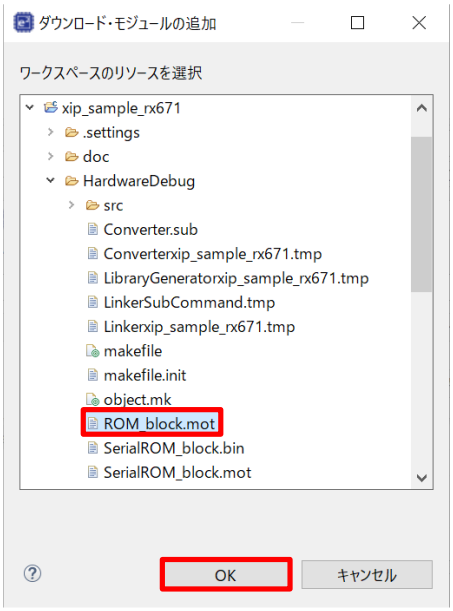


図 36　アプリケーションプログラムのデバッガ接続設定(3/6)

「ダウンロードモジュールの追加」ダイアログボックスの「OK」ボタンをクリックします。

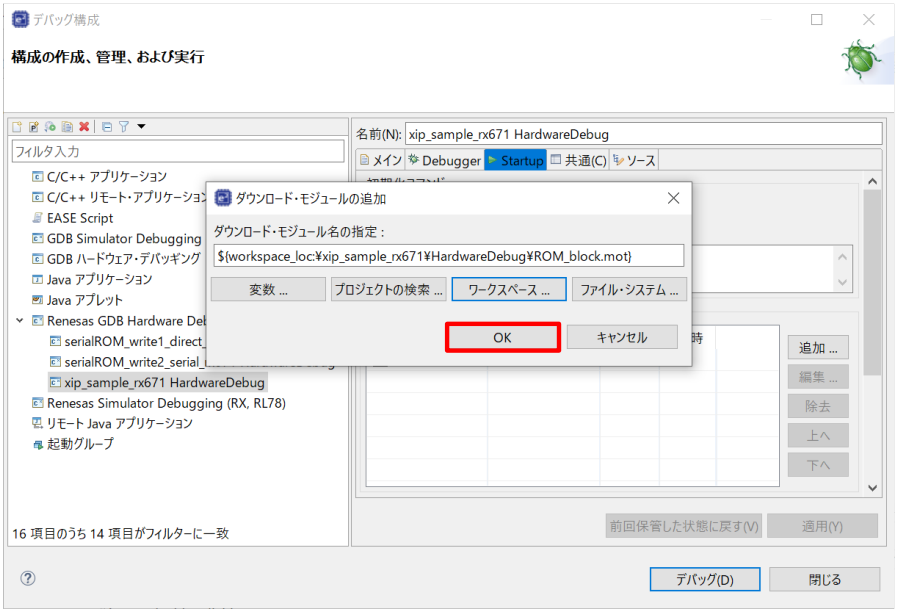


図 37　アプリケーションプログラムのデバッガ接続設定(4/6)

ファイル名「プログラム・バイナリ[xip_sample_rx671.x]」の「ロード・タイプ」のプルダウンメニューから「シンボルのみ」を選択します。

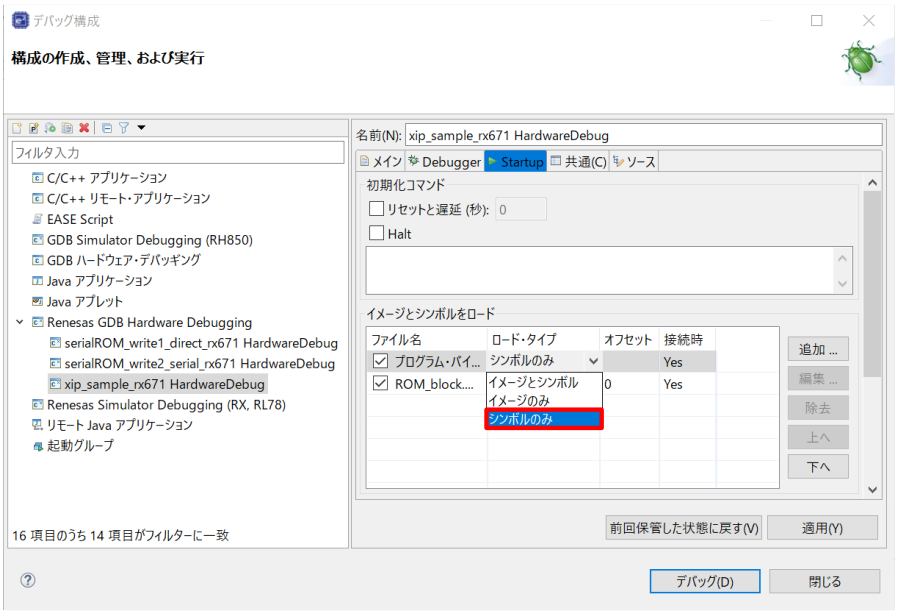


図 38　アプリケーションプログラムのデバッガ接続設定(5/6)

ファイル名「ROM_block.mot」の「ロード・タイプ」のプルダウンメニューから「イメージのみ」を選択します。最後に「適用(Y)」ボタンをクリックします。

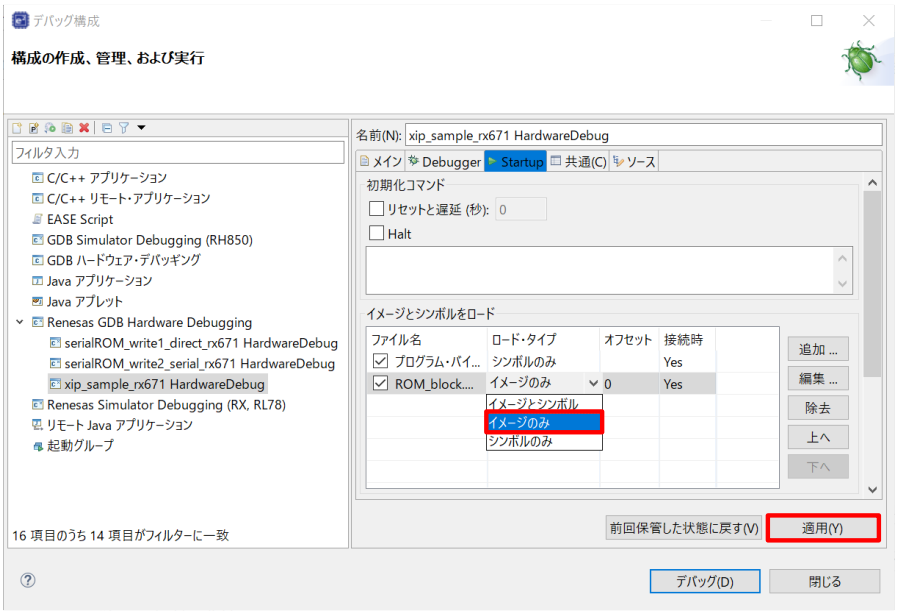


図 39　アプリケーションプログラムのデバッガ接続設定(6/6)

3.5.2　注意事項

3.5.2.1　シリアル ROM 上に配置するアプリケーションプログラムの配置アドレスについて

ライタ用プログラム 1 を使用する場合、シリアル ROM 上に配置するアプリケーションプログラムの先頭アドレス(セクション SerialROM_sec のアドレス割り当て)は、0x70000000 としてください。

ライタ用プログラム 1 はシリアル ROM 上に配置するアプリケーションプログラムをバイナリデータとして取り込み、シリアル ROM の先頭ブロック(アドレス 0x00000000)に書き込むためです。

ライタ用プログラム 2 を使用する場合、シリアル ROM 上に配置するアプリケーションプログラムの先頭アドレスは、QSPI 領域の 256 の倍数(下位 1 バイトが 0x00)にしてください。

RSK 搭載のシリアル ROM は、ページ(256 バイト)境界を跨いで書き込みをすると、選択したページの先頭に折り返してデータが上書きされます。このため、書き込みコマンド(Page Program コマンド)発行で最大サイズ(256 バイト)を書き込むためには、シリアル ROM への書き込み先頭アドレスが 256 の倍数である必要があります。

3.5.2.2　プロジェクト構成について

アプリケーションプログラムのプロジェクト xip_sample_rx671(または xip_sample_rx671_ek)とライタ用プログラム 1 のプロジェクト serialROM_write1_direct_rx671(または serialROM_write1_direct_rx671_ek)は、同一ワークスペースに配置してください。

また、ライタ用プログラム 1 を使用する場合、アプリケーションプログラムのプロジェクト名は変更しないでください。

変更する際は、「3.2.1.1(2)(b) 入力バイナリファイル指定(-binary)オプション設定」に記述した-binary オプション設定において、SerialROM_block.bin ファイル格納先(下記の赤字部分)を変更する必要があります。

-binary="${WorkspaceDirPath}/xip_sample_rx671/HardwareDebug/SerialROM_block.bin"

(SerialROM_WriteData_sec:4/DATA,_g_SerialROM_WriteData)

注 1. 本ドキュメントでは内容説明のため、上記のようにオプション記述の途中で改行していますが、実際のオプション設定では改行していません。

注 2. 上記の記述は RSK ボード向けのファイルパス記述です。EK ボード使用時は、プロジェクト名を EK ボード向けプロジェクト名に置き換える必要があります。

3.5.2.3　ライタ用プログラム 1 をビルドする際の注意事項

アプリケーションプログラムを変更した場合、ライタ用プログラム 1 はプロジェクトをクリーンにしてからビルドしてください。

3.5.2.4　シリアル ROM 上のプログラムのデバッグについて

アプリケーションプログラムをデバッガ接続して、シリアル ROM 上のプログラムをデバッグする場合、シリアル ROM 上のプログラムに対してソフトウェアブレークは設定できません。

3.5.2.5　アプリケーションプログラムを Renesas Flash Programmer で RX671 に書き込む場合

アプリケーションプログラムを Renesas Flash Programmer(以下、RFP)を使用して RX671 に書き込む際は、アプリケーションプログラムのビルド生成ファイル ROM_block.mot をご使用ください。

RFP の使用方法は、「Renesas Flash Programmer フラッシュ書き込みソフトウェア ユーザーズマニュアル(R20UT5038)」をご参照ください。

4.　プロジェクトをインポートする方法

サンプルコードは e² studio のプロジェクト形式で提供しています。本章では、 e² studio へプロジェクトをインポートする方法を示します。インポート完了後、ビルドおよびデバッグの設定を確認してください。

4.1　e² studio でのインポート手順

下記の手順で e² studio にインポートしてください。
(使用する e² studio のバージョンによっては画面が異なる場合があります。)

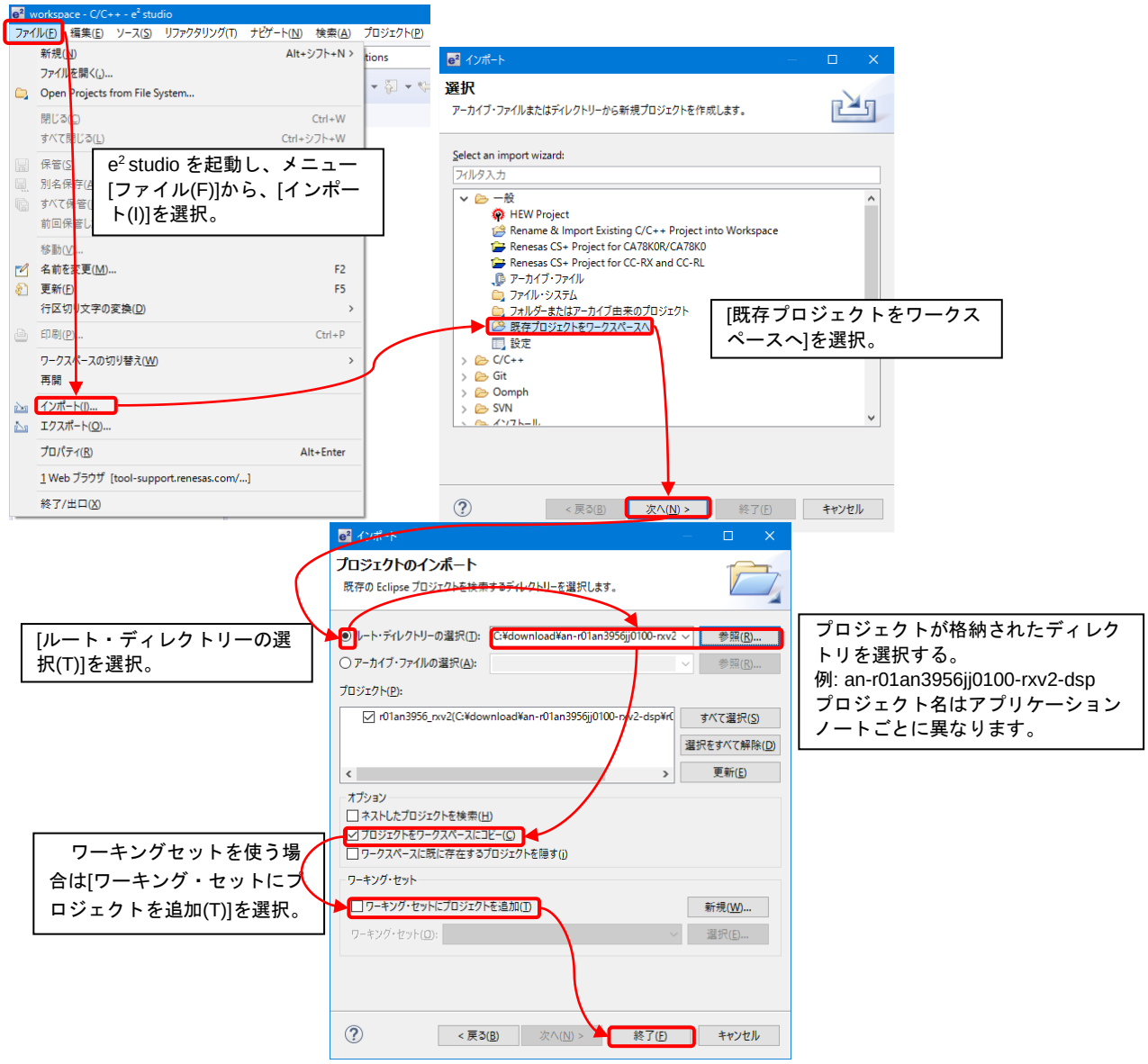


図 4.1　プロジェクトを e² studio にインポートする方法

5. 開発環境の入手

5.1 e² studio の入手方法

以下の URL にアクセスし、e² studio をダウンロードしてください。
https://www.renesas.com/jp/ja/software-tool/e-studio

なお、本ドキュメントは、「表 3.41 動作確認条件」に記載しているバージョン以降を使用することを前提としています。それよりも古いバージョンを使用した場合、e² studio の一部機能を使用できない可能性があります。ダウンロードする場合、ホームページに掲載されている最新バージョンの e² studio を入手してください。

5.2 コンパイラパッケージの入手方法

以下の URL にアクセスして、RX ファミリ用 C/C++コンパイラパッケージをダウンロードしてください。
https://www.renesas.com/jp/ja/software-tool/cc-compiler-package-rx-family

6. 補足

6.1 無償評価版の「RX ファミリ用 C/C++コンパイラパッケージ」を利用する場合の注意事項

無償評価版の「RX ファミリ用 C/C++コンパイラパッケージ」には、使用期限と使用制限があります。使用期限が過ぎた場合、リンクサイズが 128K バイト以内に制限されるためロードモジュールが正しく生成されなくなる場合があります。

詳しくは、ルネサスのホームページにある、無償版ソフトウェアツールのページを参照してください。
https://www.renesas.com/jp/ja/software-tool/evaluation-software-tools

7. 参考資料

- RX671 グループ ユーザーズマニュアル ハードウェア編 (R01UH0899)
- CC-RX コンパイラ ユーザーズマニュアル(R20UT3248)
- RX ファミリ ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)
- RX ファミリ QSPIX モジュール Firmware Integration Technology (R01AN5685)
- RX ファミリ CMT モジュール Firmware Integration Technology (R01AN1856)
- RX ファミリ SCI モジュール Firmware Integration Technology (R01AN1815)
- RX ファミリバイト型キューバッファ(BYTEQ)モジュール Firmware Integration Technology (R01AN1683)
- Renesas e² studio スマート・コンフィグレータ ユーザーガイド(R20AN0451)
- Renesas Flash Programmer フラッシュ書き込みソフトウェア ユーザーズマニュアル(R20UT5038)
- Renesas Starter Kit+ for RX671 ユーザーズマニュアル (R20UT4879)
- Renesas Starter Kit+ for RX671 回路図 (R20UT4878)
- EK-RX671 ユーザーズマニュアル (R20UT5234)
- EK-RX671 回路図(D019442_04)

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。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Jan.21.22	-	新規作成
2.00	Jun.30.22	全体	関数仕様の内容を関数一覧に移動し、関数仕様の項目を削除。 rev1.00 のライタ用プログラムをライタ用プログラム 1 に変更。 Rev1.00 のライタ用プログラムのプロジェクト名 serialROM_write_rx671 を serialROM_write1_direct_rx671 に変更。
		P6	「図 3　RX671 とホスト PC との接続図」追加。 表 2.1 タイトル変更。 「表 2.2　RX671 とホスト PC との接続に使用する SCI 端子」追加。
		P10	使用 FIT モジュールの参照先を修正。
		P12、P15	図 7 において、シリアル ROM 上に配置するプログラムのビルド生成ファイルとしてモトローラ S 形式ファイルの出力を追加。また、本図の説明文を変更。 これに伴い図 11 において、分割出力ファイル(Stype 用)に SerialROM_block.mot=SerialROM_sec を追加。また、本図の説明文を変更。
		P13	シリアル ROM 上に配置するプログラムについて、アプリケーションプログラムで設定しているアドレス値の下に、設定アドレスに関する補足説明を追加。
		P19	「表 3.3 アプリケーションプログラムで使用するファイル」に記載しているファイル名 cmd_serial_rom.h を serial_rom.h に変更。
		P22	rev2.00 ではライタ用プログラム 2 を追加したため、「3.2 ライタ用プログラム」の項目の下にライタ用プログラム 1(Rev1.00 のライタ用プログラム)の項目とライタ用プログラム 2 の項目を記述するように変更。
		P24、P25	ライタ用プログラム 1 のプロジェクト名変更に伴い、図 17、図 19 を差し替え。
		P26	図 20 のフローチャートの「シリアル ROM 先頭 1 ブロックへのデータ書き込み」補足説明のシリアル ROM のアドレスの表現を QSPI 領域(0x70000000)からシリアル ROM アドレス(0x00000000)に変更。
		P28	「表 3.9 ライタ用プログラム 1 で使用するファイル」に記載しているファイル名 cmd_serial_rom.h を serial_rom.h に変更。
		P31～P45	「3.2.2 ライタ用プログラム 2」を追加。
		P46	「表 3.34 使用している FIT モジュール一覧」を変更。
		P47	「3.3.2 FIT モジュールの設定」に CMT、SCI、BYTEQ の各 FIT モジュールの設定を追加。
		P50	「表 3.41 動作確認条件」について 統合開発環境、コンパイラ、サンプルプログラムのバージョン変更。 項目構成の変更に伴いコンパイラオプション欄に記載している参照先を変更。

2.00	Jun.30.22	P51	アプリケーションプログラムのビルド生成ファイル増加に伴い図 32 を変更。 ライタ用プログラム 2 の追加に伴い図 33 を追加。
		P52～P54	プロジェクトの名称変更、追加に伴い図 34～図 39 を差し替え。
		P55	「3.5.2 注意事項」に「3.5.2.1 シリアル ROM 上に配置するアプリケーションプログラムの配置アドレスについて」を追加。 項目構成の変更に伴い「3.5.2.2 プロジェクト構成について」の参照先項目を変更。
		P57	「7 参考資料」に CMT、SCI、BYTEQ の FIT モジュールの情報を追加。
2.10	Jan.10.24	全体	EK ボード向けサンプルプログラムの追加
		P6～P9	「2.ハードウェア構成」の章に「2.1 Renesas Starter Kit+ for RX671」と「2.2 EK-RX671」を追加
		P10	EK ボード向けに記載を一部変更 表 3.1 のタイトル変更、表 3.2 を追加
		P11	EK ボード向けに記載を一部変更 図 6 の LED 個数表記を削除
		P12	図 7 の LED 個数表記を削除
		P18	EK ボード向けに記載を一部変更 図 15 内に LED 制御への注釈を追加
		P20	表 3.5 のタイトル変更
		P21	表 3.6 の追加、表 3.7 内の記載を変更
		P22	EK ボード向けに記載を一部変更 図 16 内の LED 個数表記を削除。
		P23	「表 3.8 シリアル ROM の書き込み状態表示」を追加
		P24,25	表記修正（内容に変更なし）
		P26,P27	図 20 のタイトル変更、「図 21　ライタ用プログラム 1(EK ボード向け)の概略フロー」を追加
		P29,30	表 3.11 のタイトル変更、「表 3.12 ライタ用プログラム 1 で使用する定数　(EK ボード向け)」の追加
		P36	表 3.18 内、STRING_MAX_SIZE の設定値変更
		P38	表 3.23 内、定数 CMD_WRSR1、CMD_WRSR2、CMD_RDSR1、CMD_RDSR の追加
		P49	表 3.38 タイトル変更、表 3.39 SCI モジュールの設定(ライタ用プログラム 2 のみ)(EK ボード向け)の追加
		P50	表 3.41 内「使用ボード」項目に EK-RX671 を追加
		P55	「3.5.2.2 プロジェクト構成について」に EK ボード向けの記載を追加
		P57	参考資料の更新

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。