RL78/G23

I2C 通信によるブート・スワップを使用したセルフ・プログラミング

要旨

本アプリケーションノートは、I2C 通信を使用したセルフ・プログラミングの概要を説明します。

フラッシュ・セルフ・プログラミング・ライブラリ (Renesas Flash Driver RL78 Type01) を使用し、フラッシュ・メモリの書き換えとブート・スワップを行います。

動作確認デバイス

RL78/G23

本アプリケーションノートを他のマイクロコントローラへ適用する場合、そのマイクロコントローラの仕様に合わせて変更し、十分評価してください。

関連ドキュメント

本アプリケーションノートに関連するドキュメントを以下に示します。あわせて参照してください。

- RL78 ファミリ Renesas Flash Driver RL78 Type01 ユーザーズマニュアル (R20UT4830)
- RL78 ファミリ Renesas Flash Driver RL78 Type01 SC 対応仕様 (Code Flash) (R20AN0653)
- RL78 ファミリ Renesas Flash Driver RL78 Type01 SC 対応仕様 (Data Flash) (R20AN0654)
- RL78 ファミリ Renesas Flash Driver RL78 Type01 SC 対応版 (Extra Area) (R20AN0655)
- RL78 ファミリ Renesas Flash Driver RL78 Type01 SC 対応仕様(Common) (R20AN0656)
- RL78/G23-128p Fast Prototyping Board ユーザーズマニュアル (R20UT4870)

目次

1.　概要

本アプリケーションノートでは、I2C 通信により RL78/G23 のフラッシュ・メモリの書き換えとブート・スワップをセルフ・プログラミングにより行うサンプル・プログラムについて説明します。

1.1　構成

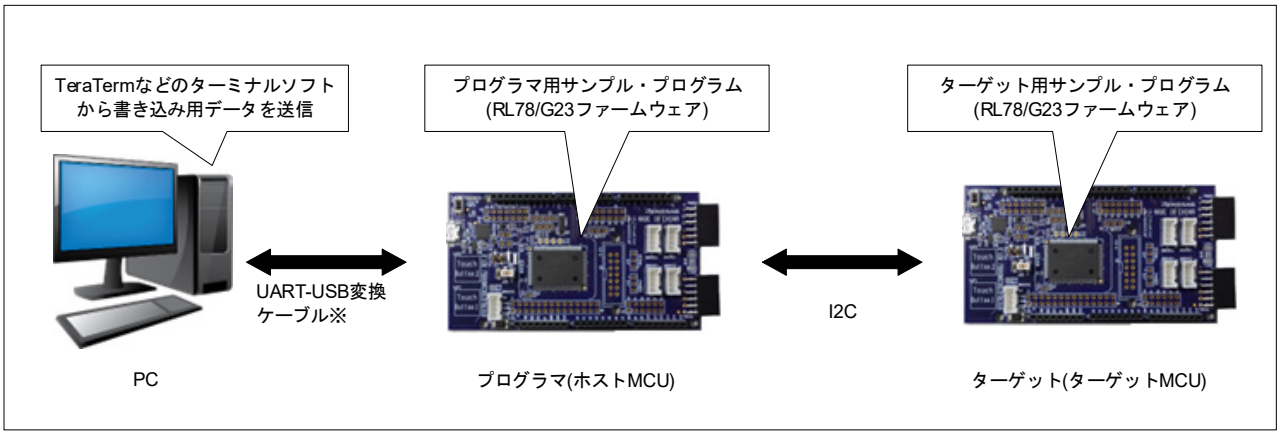
RL78/G23 搭載ボード(RL78/G23-128p Fast Prototyping Board)を 2 枚使用し、それぞれホスト MCU とターゲット MCU として割り当てます。

PC とホスト MCU を USB-UART 変換ケーブルで接続、ホスト MCU とターゲット MCU を I2C で接続して使用します。

- ホスト MCU (プログラマ)
  PC からターミナル・ソフトにより送信されたターゲット MCU の書き込み用データ(モトローラ S フォーマット)を UART で受信し、そのデータを I2C でターゲット MCU へ送信します。
  また、書き換え結果を PC へ UART で送信します。
- ターゲット MCU (ターゲット)
  ホスト MCU(プログラマ)から書き込み用データを I2C で受信し、そのデータを使用してセルフ・プログラミングによりフラッシュ・メモリの書き換えを行います。
  ブート・クラスタ 0 以外のコード・フラッシュとデータ・フラッシュの書き換えに対応します。
  ブート・クラスタ 1 の書き換えを行った場合は、ブート・スワップを実行します。

ホスト MCU とターゲット MCU、それぞれのサンプル・プログラムを提供します。

図 1-1　構成図



※ホスト MCU が搭載されている RL78/G23-128p Fast Prototyping Board の USB コネクタは RL78 COM port デバッグ・ツールとして使用するため、本サンプル・プログラムでは、別途 USB-UART 変換ケーブルを使用して PC と接続しています。

1.2　プロジェクト構成

本アプリケーションノートでは、プログラマ用とターゲット用のサンプル・プログラムを提供します。

ターゲット用サンプル・プログラムは、ブート・プログラムとユーザ・プログラムの 2 種類のプロジェクトから構成されています。

ブート・プログラムは、セルフ・プログラミングによりフラッシュ・メモリの書き換えを行うプログラムで、ブート・クラスタ 0 (0x00000 - 0x03FFF)に割り当てます。

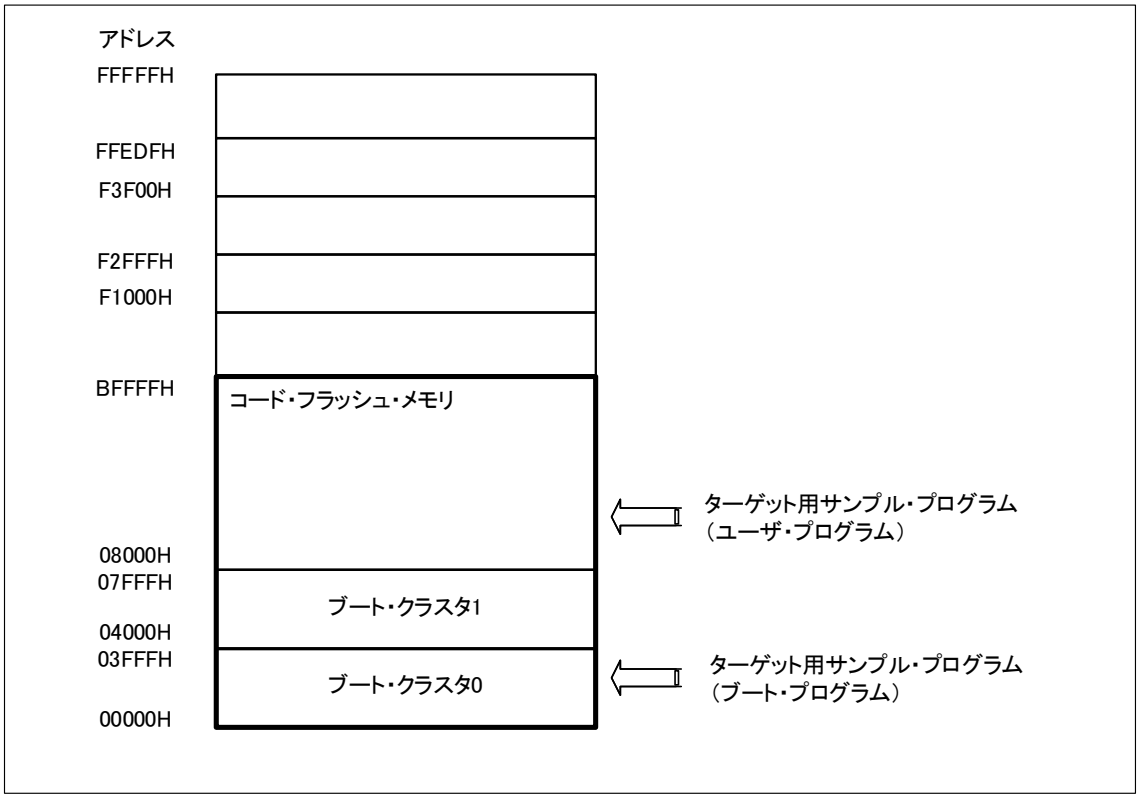
ユーザ・プログラムは、ユーザ処理を実装するプログラムとなっており、0x08000 番地以降に割り当てます。

以下の表 1-1 にプロジェクトの構成を示します。

表 1-1　プロジェクト構成

フォルダ名	説明
rl78g23_UART_To_I2C	プログラマ用サンプル・プログラム
rl78g23_RFD_with_I2C	ターゲット用サンプル・プログラム (ブート・プログラム)
rl78g23_FlashWriter_UserProgramSample	ターゲット用サンプル・プログラム (ユーザ・プログラム)

図 1-2　プログラム割り当て（ターゲット）



2.　開発環境

本アプリケーションノートのサンプル・プログラムは下記の条件で動作を確認しています。

表 2-1　動作確認条件

開発ツール		説明
MCU (プログラマ，ターゲット共通)		型番：R7F100GSN2DFB (内蔵メモリ：コード・フラッシュ 768KB，内蔵 RAM 48KB，データ・フラッシュ 8KB)
評価ボード (プログラマ，ターゲット共通)		RL78/G23-128p Fast Prototyping Board 型番：RTK7RLG230CSN000BJ 注意：本サンプル・プログラムは、HS モード(高速オンチップ・オシレータ 32MHz)で動作するため、VDD を 1.8V 以上に設定してください。
デバッグ・ツール		本サンプル・プログラムをデバッグする場合は、RL78/G23-128p Fast Prototyping Board の RL78 COM port デバッグ・ツールを使用。 E2 または E2 Lite 使用する場合は・RL78/G23-128p Fast Prototyping Board ユーザーズマニュアルを参照し、ジャンパ等の設定を行ってください。
CS+	統合開発環境	ルネサス エレクトロニクス製 CS+ for CC V8.14.00
	C コンパイラ	ルネサス エレクトロニクス製 CC-RL V1.15.01
e2studio	統合開発環境	ルネサス エレクトロニクス製 e2studio 2025-07 (25.07.0)
	C コンパイラ	ルネサス エレクトロニクス製 CC-RL V1.15.01
IAR	統合開発環境	IAR Systems 製 IAR Embedded Workbench for Renesas RL78 V5.20
	C コンパイラ	IAR Systems 製 IAR C/C++ Compiler for Renesas RL78 V 5.20.1
セルフ・プログラミング用ライブラリ		Renesas Flash Driver RL78 Type01

2.1　Renesas Flash Driver RL78 Type01

Renesas Flash Driver RL78 Type01 は、RL78/G2x のフラッシュ・メモリ内のデータを書き換えるためのソフトウェアです。

詳細は「Renesas Flash Driver RL78 Type01 ユーザーズマニュアル(R20UT4830)」を参照してください。

Renesas Flash Driver RL78 Type01 は、下記 URL から取得することができます。

https://www.renesas.com/ja/software-tool/code-flash-libraries-flash-self-programming-libraries

https://www.renesas.com/ja/software-tool/data-flash-libraries

3.　外部仕様

3.1　ホスト通信仕様

PC とプログラマ(RL78/G23-128p Fast Prototyping Board)間の通信仕様について説明します。

プログラマ(RL78/G23-128p Fast Prototyping Board)の P02/TXD1 と P03/RXD2、GND とホスト PC を USB-UART 変換ケーブルを用いて接続します。

本サンプル・プログラムでは PC とプログラマ(RL78/G23-128p Fast Prototyping Board)間の通信設定は以下としています。

表 3-1　ホスト通信設定

項目	設定
データ・ビット長[bit]	8
データ転送方向	LSB ファースト
パリティ設定	なし
転送レート[bps]	115,200
ストップビット[bit]	1
フロー制御	ソフトウェア(Xon/Xoff)

3.2　サンプル・プログラムの実行手順

3.2.1　動作環境の構築

以下に本サンプル・プログラムの動作環境の構築手順を示します。

① 「4.2 接続方法」を参照し、PC とプログラマ、ターゲットを接続します。

② 「1.2 プロジェクト構成」の各サンプル・プログラムをビルドし、Renesas Flash Programmer を使用して、ホスト MCU とターゲット MCU へファームウェアを書き込みます。
ターゲット MCU へ書き込むファイルについては、「6.5.1 ターゲット用プログラム・ファイルの作成」を参照してください。

3.2.2　サンプル・プログラムの実行とフラッシュ・メモリの書き換え

以下にターゲット MCU(RL78/G23)のフラッシュ・メモリの書き換え手順を示します。

① PC からターミナル・ソフトを起動し、「3.1 ホスト通信仕様」に従い通信設定を行います。

② ターゲット(RL78/G23-128p Fast Prototyping Board)のユーザスイッチ(SW)を押しながらプログラマとターゲットの電源(VDD)を投入します。SW を押しながらターゲットを起動することで、書き換えモードに移行しブート・プログラム側の処理が有効となります。

SW を押さない場合は、ユーザ・プログラムの処理が実行されます。書き換えモードの詳細については、「6.1 モード選択」を参照してください。

③ ターゲット MCU 起動後、ホスト MCU を起動(リセット)してください。

④ プログラマとターゲット間の接続確認が成功するとターミナル・ソフトに”Send an S-Record file:”が表示されます。
失敗した場合は、ターミナル・ソフトに”Target Not Found.”が表示されます。

⑤ ターミナル・ソフトを使用して、ユーザ・プログラムまたはブート・プログラムの書き込み用データ(モトローラ S フォーマット)を送信します。
使用するファイルについては、「6.5.1 ターゲット用プログラム・ファイルの作成」を参照してください。

⑥ 書き込みが成功した場合、ターミナル・ソフトに"Target Processes Ended."が表示されます。

その他のメッセージの仕様については、「3.4 メッセージ仕様」を参照してください。

3.3　LED 表示仕様

以下の表 3-2 にプログラマ、表 3-3 にターゲットの動作状態と LED 表示の関係を示します。

表 3-2　プログラマの LED 表示

動作状態	LED1	LED2
動作中	点灯	点灯
異常終了	消灯	消灯
正常終了	消灯	点灯

表 3-3　ターゲットの LED 表示

動作状態	LED1	LED2
ブート・プログラム動作中	点灯	点灯
ユーザ・プログラム動作中	点滅	消灯
異常終了	消灯	消灯
正常終了	消灯	点灯

3.4　メッセージ仕様

プログラマは文字列のメッセージを PC へ送信します。

表 3-4 にメッセージ一覧を示します。

表 3-4　メッセージの説明

メッセージ	説明
Target Processes Ended.	書き込み成功 書き込みが成功した場合のメッセージです。
S Record Error.	モトローラ S フォーマットが不正 送信するモトローラ S フォーマットが不正の場合に発生します。 モトローラ S フォーマットのデータレコードがアドレス昇順になっていない場合も本エラーが発生します。
I2C Error.	プログラマエラー プログラマ側で I2C 通信エラーがあった場合に発生します。
Target Not Found.	ターゲット通信エラー ホスト MCU が、ターゲット MCU との接続確認をできなかった場合。
Target Occurred an Error.	ターゲット書き込みエラー ターゲット書き込み処理でエラーがあった場合に発生します。

3.5　プログラマ・フローチャート

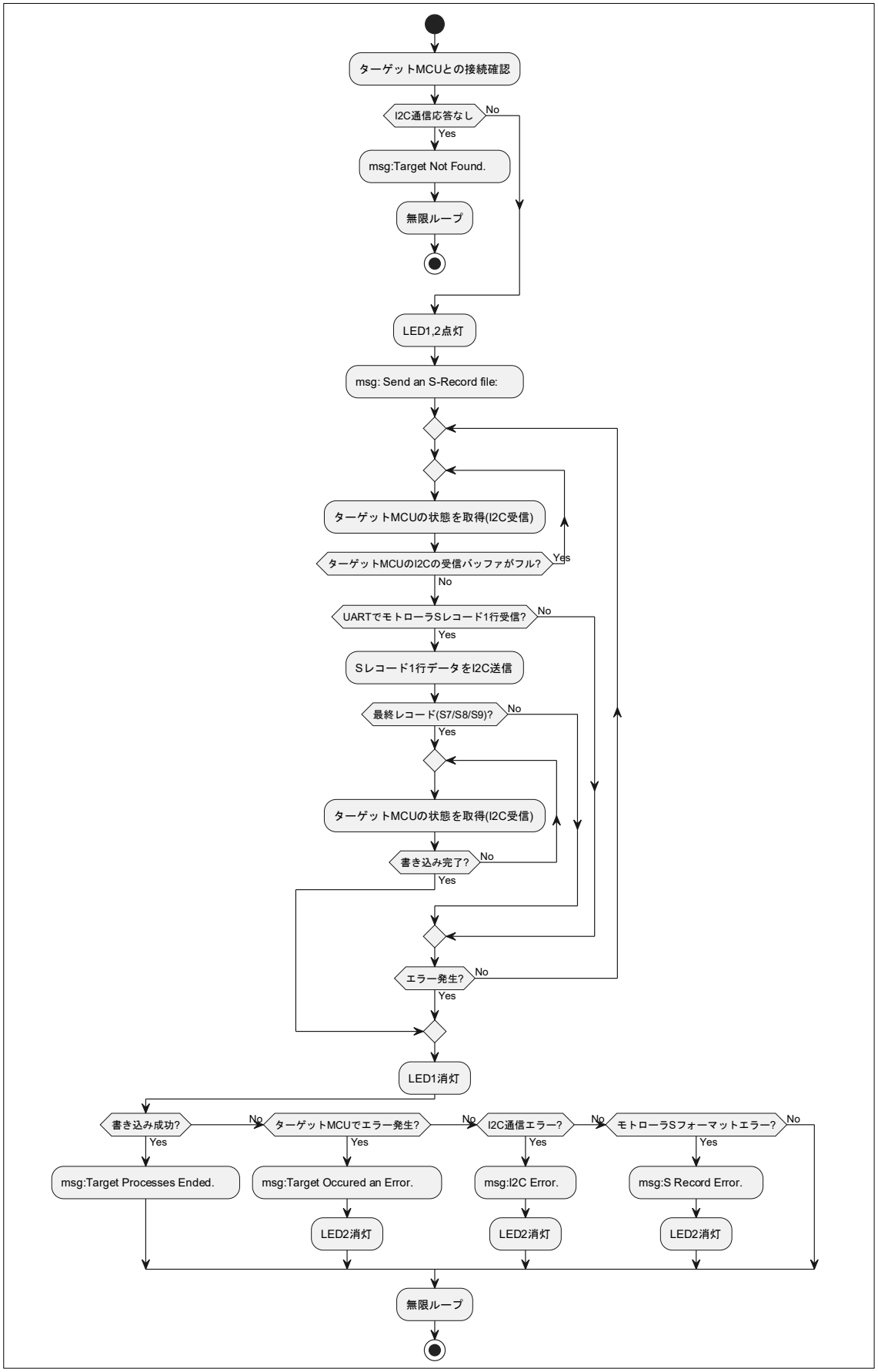
3.5.1　プログラマ・メイン処理

図 3-1 にプログラマのメイン処理の動作を示します。

PC から UART によりモトローラ S フォーマットのファイルを受信し、ターゲットへ I2C により S レコードデータを 1 行単位で送信します。ターゲットの状態を I2C 通信により取得して、ターゲットと同期をとりながら処理を行います。

また、フローチャートでは記載していませんが、PC との通信はソフトウェアフロー制御(Xon/Xoff)による UART 通信を行い、プログラマ側の受信バッファがフルになる場合は、PC からの送信を停止させる動作となっています。受信バッファに空きができたら、UART 通信を再開します。

図 3-1　プログラマのメイン処理



3.6　ターゲット・フローチャート

3.6.1　ターゲット・メイン処理

図 3-2 にターゲット(ブート・プログラム)のメイン処理の動作を示します。

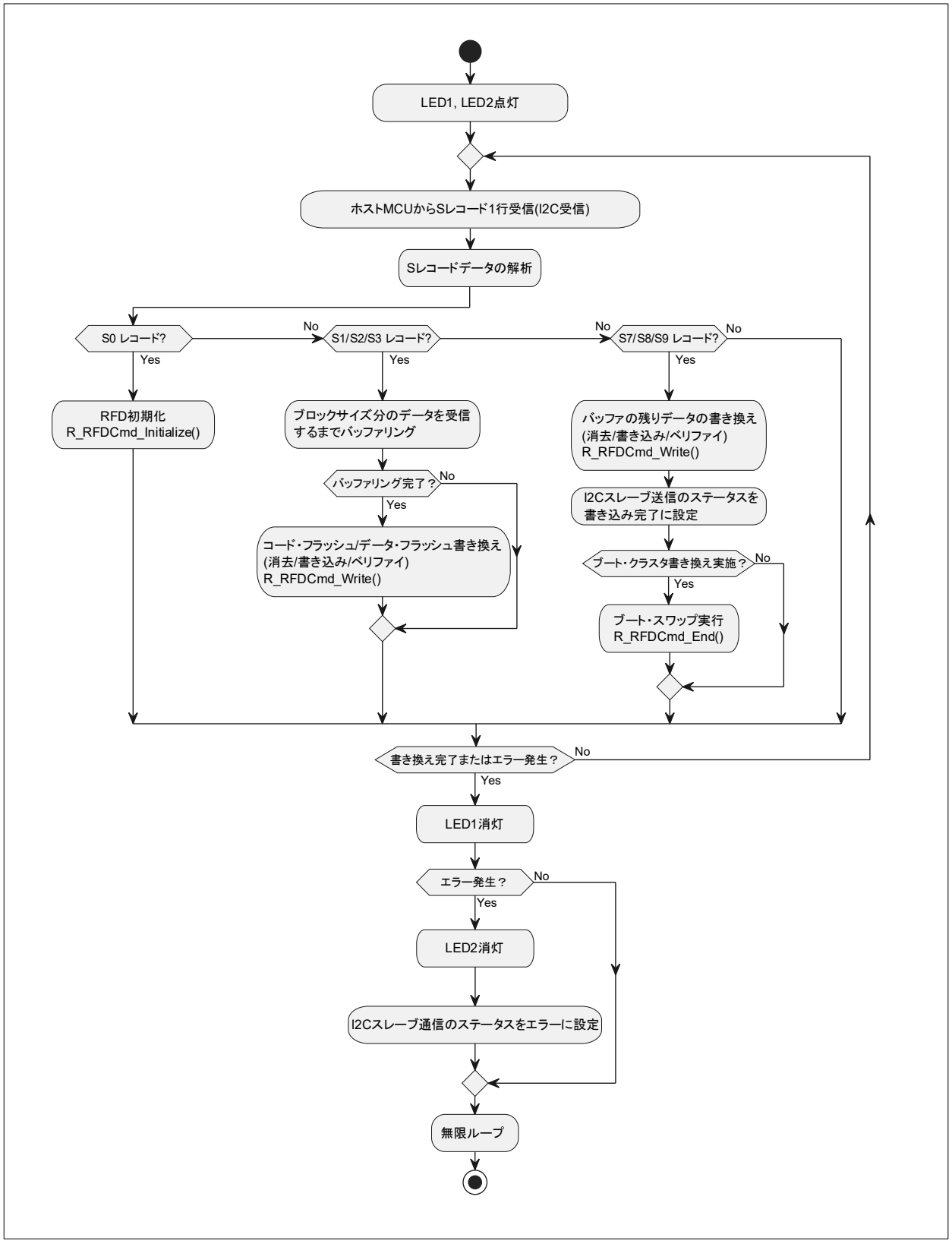
プログラマから I2C により S レコードデータを受信し、そのレコードタイプに応じて、初期化、書き換え、ブート・スワップ(ブート・クラスタ 1 書き換え時)を実行します。

S レコードファイルのデータ対象範囲は、ブート・クラスタ 1(0x04000 - 0x07FFF)を除くコード・フラッシュおよびデータ・フラッシュとなります。

ブート・クラスタ 1 に書き込む S レコードファイルのアドレスは 0x00000 - 0x03FFF の範囲としてください。（0x000C0-0x000C3 にオプション・バイト、0x000C4-0x000CD にオンチップ・デバッグ・セキュリティ ID を設定）ブート・プログラムはアドレスに 0x4000 のオフセットを加算してブート・クラスタ 1 に書き込みを行います。

ブート・クラスタ 1 の書き換え時は、最後にブート・スワップを実行するため、ブート・クラスタ 1 とブート・クラスタ 0 が入れ替わります。

図 3-2　ターゲットのメイン処理



4.　ハードウェア説明

4.1　ハードウェア構成例

図 4-1、図 4-2 に本アプリケーションノートで使用するプログラマとターゲットのハードウェア構成例を示します。

図 4-1　ハードウェア構成(プログラマ)

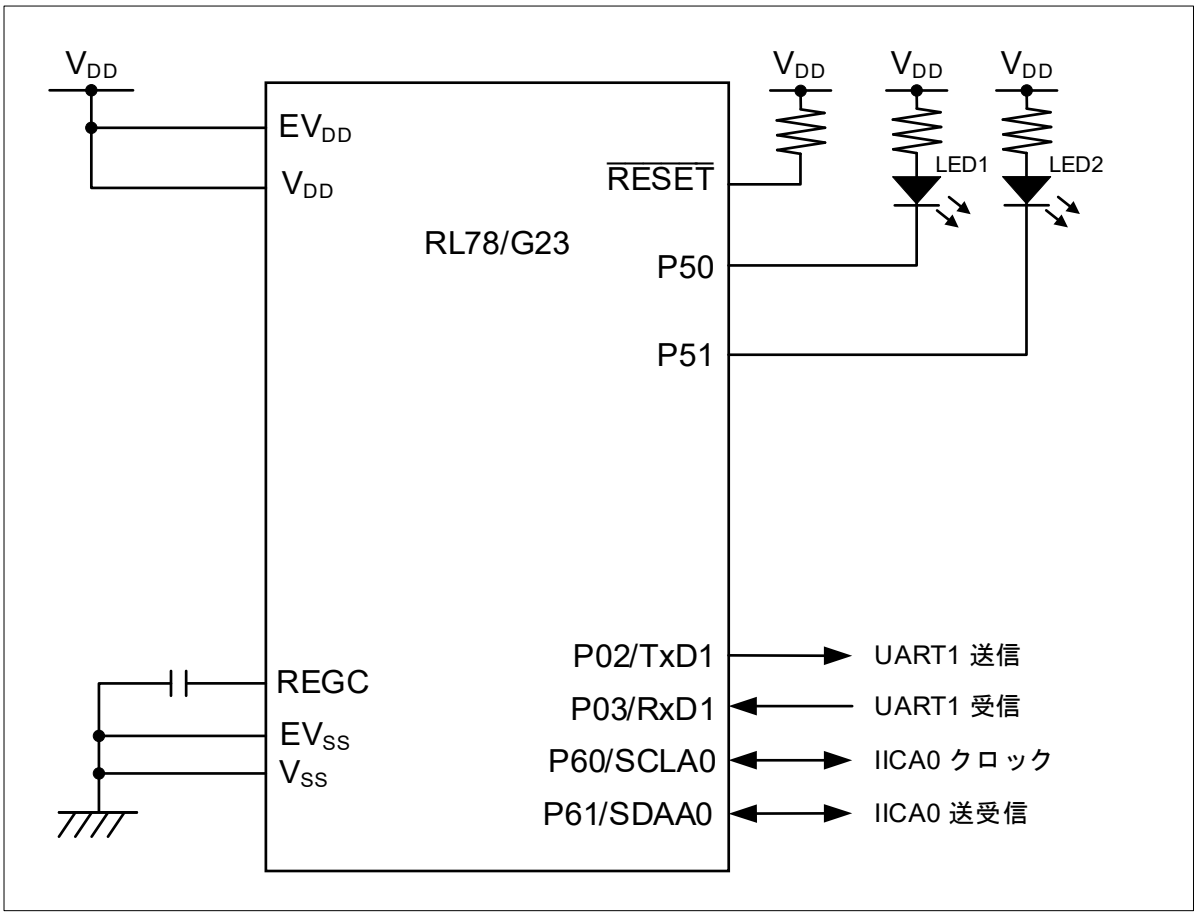
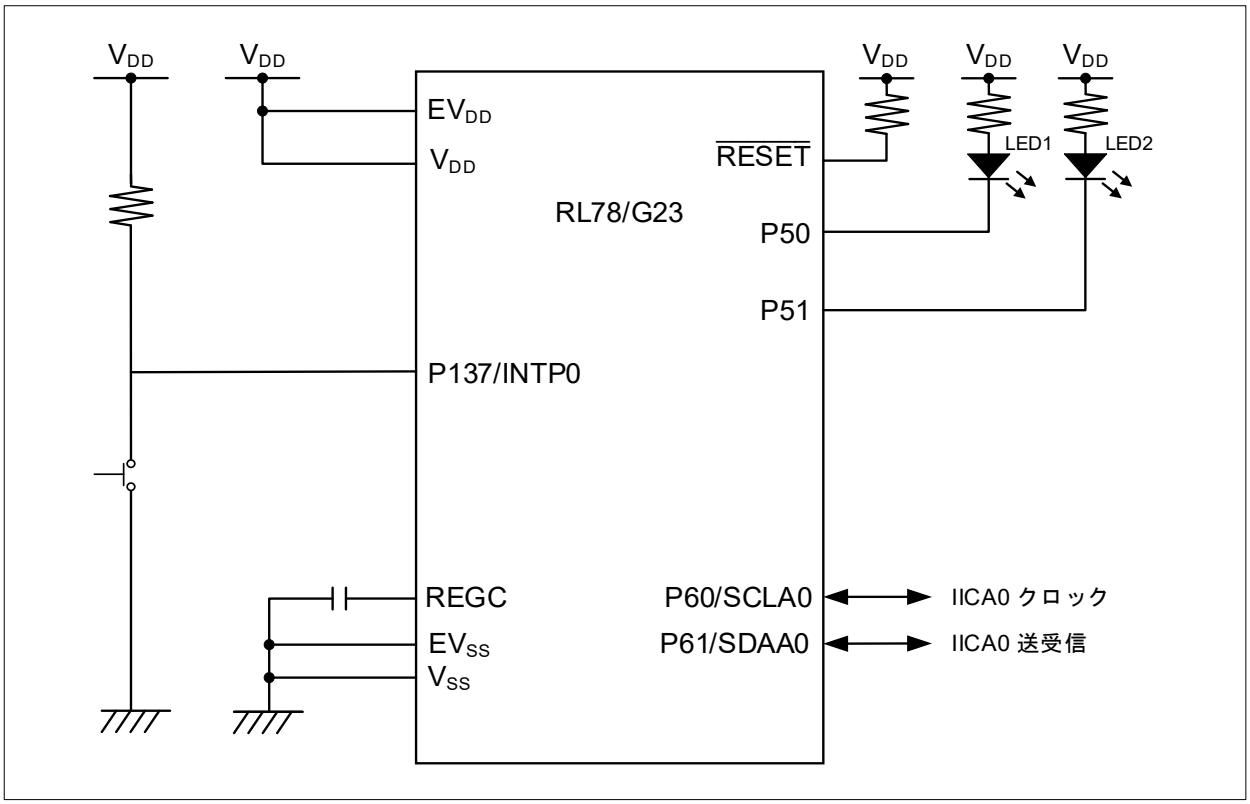


図 4-2　ハードウェア構成(ターゲット)



注意 1.　この回路イメージは接続の概要を示す為に簡略化しています。実際に回路を作成される場合は、端子処理などを適切に行い、電気的特性を満たすように設計してください (入力専用ポートは個別に抵抗を介して V_{DD} 又は V_{SS} に接続してください) 。

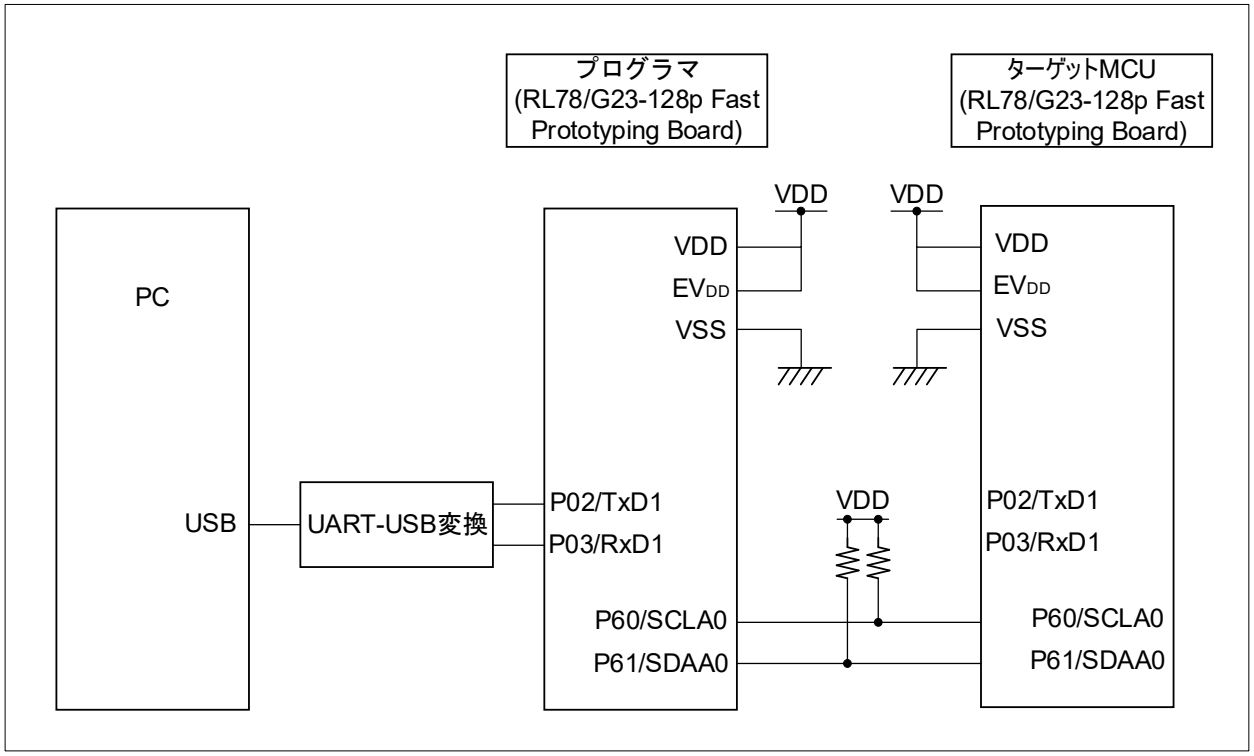
注意 2.　EV_{SS} で始まる名前の端子がある場合には V_{SS} に、EV_{DD} で始まる名前の端子がある場合には V_{DD} にそれぞれ接続してください。

注意 3.　V_{DD} は LVD0 にて設定したリセット解除電圧 (V_{LVD0}) 以上にしてください。

4.2　接続方法

PC とプログラマ、ターゲットの接続方法を以下に示します。

図 4-3　接続図



4.3　使用端子一覧

表 4-1、表 4-2 にプログラマとターゲットのサンプル・プログラムで使用する端子と機能を示します。

表 4-1　使用端子一覧(プログラマ)

端子名	入出力	機能
P02/TxD1	出力	ホスト PC 通信用送信端子(UART1)
P03/RxD1	入力	ホスト PC 通信用受信端子(UART1)
P60/SCLA0	出力	ターゲットインタフェース通信用クロック端子(IICA0)
P61/SDAA0	出力	ターゲットインタフェース通信用送受信端子(IICA0)
P50, P51	出力	LED1,LED2 への出力端子

表 4-2　使用端子一覧(ターゲット)

端子名	入出力	機能
P60/SCLA0	入力	ターゲットインタフェース通信用クロック端子(IICA0)
P61/SDAA0	入力	ターゲットインタフェース通信用送受信端子(IICA0)
P50, P51	出力	LED1,LED2 への出力端子
P137	入力	モード選択(書き換えモード/ユーザ・モード)で使用

注意　本アプリケーションノートは、使用端子のみを端子処理しています。実際に回路を作成される場合は、端子処理などを適切に行い、電気的特性を満たすように設計してください。

4.4　RL78/G23-128p Fast Prototyping Board の設定

E2 エミュレータ Lite や E2 エミュレータを使用してデバッグを行う場合は RL78/G23-128p Fast Prototyping Board で以下の設定が必要です。

- 「TOOL0_USB」、「RESET」、「T_RESET」のカットパターンをカットしてください。
- 「J15」、「J16」、「J19」のピンヘッダを 2-3 側にショートしてください。

VDD や EVDD の動作電圧の設定は「J20」で行います。

詳細は、「RL78/G23-128p Fast Prototyping Board ユーザーズマニュアル (R20UT4870)」を参照ください。

5.　プログラマ内部仕様

5.1　オプション・バイト設定(プログラマ)

表 5-1 にプログラマのサンプル・プログラムで使用するオプション・バイトの設定を示します。

表 5-1　プログラマのオプション・バイト設定

アドレス	設定値	内容
000C0H/040C0H	11101111B	ウォッチドッグ・タイマ動作停止
000C1H/040C1H	00111010B	LVD0 オフ
000C2H/040C2H	11101000B	HS モード、 高速オンチップ・オシレータ・クロック : 32MHz
000C3H/040C3H	10000100B	オンチップ・デバッグ許可

RL78/G23 のオプション・バイトは、ユーザ・オプション・バイト(000C0H – 000C2H)とオンチップ・デバッグ・オプション・バイト(000C3H)で構成されています。

5.2　セクション設定(プログラマ)

セクション設定はデフォルト設定(自動配置)を使用しています。

5.3　スマートコンフィグレータの設定(プログラマ)

プログラマのスマートコンフィグレータでの設定を以下に示します。

(1)　ポートの初期設定を行います。

- P50, P51 を出力モードに設定

(2)　インターバル・タイマ(16 ビット・カウンタ・モード)の初期設定を行います。

- 動作クロックを CK10 に、クロック・ソースを fCLK/2^4 に設定
- インターバル時間(上位 16 ビット)を 20ms に設定
- カウント完了割り込み設定(INTTM10)をレベル 3 に設定

(3)　シリアル・アレイ・ユニットの初期設定を行います。

- UART1 を使用 (P02 を TXD1 に、P03 を RXD1 に設定)
- 動作クロックを CK00 に、クロック・ソースを fCLK/2 に設定
- 転送モード設定をシングル転送モードに設定
- データ・ビット長設定を 8bit に設定
- データ転送方向設定を LSB に設定
- パリティ設定をパリティ・ビットなしに設定
- ストップビット長設定を 1bit に設定
- 送信データ・レベル設定を非反転に設定
- ボーレート設定を 115200bps に設定
- 送信完了割り込み設定(INTST1)をレベル 1 に設定
- 受信完了割り込み設定(INTSR1)をレベル 3 に設定
- 受信エラー割り込み設定(INTSRE1)をレベル 3 に設定
- コールバック機能設定の送信完了、受信完了、受信エラーを選択

(4)　I2C(マスタ・モード)の初期設定を行います。

- IICA0 を使用 (P60 を SCLA0、P61 を SDAA0 に設定)
- 動作クロックを fCLK/2 に設定
- 自局アドレス設定のアドレスを 16 に設定
- 動作モードを標準に、転送クロックを 100000bps に設定
- 通信完了割り込み設定(INTIICA0)をレベル 3 に設定
- コールバック機能設定のマスタ送信完了、マスタ受信完了、マスタ・エラーを選択
- コールバック拡張機能を選択

5.4　定義値(プログラマ)

SRECORD_TYPE は列挙型であり、モトローラ S フォーマットの S レコードデータの種類を定義します。

表 5-2 に SRECORD_TYPE の仕様を示します。

表 5-2　SRECORD_TYPE

定義値	説明
SRECORD_0	S0 レコード
SRECORD_1	S1 レコード
SRECORD_2	S2 レコード
SRECORD_3	S3 レコード
SRECORD_4	S4 レコード
SRECORD_5	S5 レコード
SRECORD_6	S6 レコード
SRECORD_7	S7 レコード
SRECORD_8	S8 レコード
SRECORD_9	S9 レコード
SRECORD_ERR	不正な S レコードデータ

5.5　関数仕様 (プログラマ)

5.5.1　関数一覧

表 5-3 にサンプル・プログラム(プログラマ)で使用する主な関数を示します。

表 5-3　関数一覧

関数名	概要
R_Get_SRecordType	Sレコードタイプの取得
R_Config_UART1_SendMessage	PCへのメッセージ送信
r_SendSRecord_I2C	ターゲットへデータ送信
r_Receive1Byte_I2C	ターゲットから1バイトデータ受信
r_Receive1Byte_I2C_Sparse	ターゲットから1バイトデータ受信 (20msウェイトあり)

5.5.2　関数説明

サンプル・プログラム(プログラマ)で使用する主な関数の仕様を示します。

R_Get_SRecordType	
概　要	S レコードタイプの取得
書　式	SRECORD_TYPE R_Get_SRecordType(uint8_t * srecord_str)
引　数	srecord_str: S レコード文字列(1 行)
戻り値	レコードタイプ
説　明	S レコード 1 行のデータを解析し、レコードタイプを取得します。

R_Config_UART1_SendMessage	
概　要	PC へのメッセージ送信
書　式	MD_STATUS R_Config_UART1_SendMessage(const char * p_msg, uint8_t len, uint8_t is_waiting)
引　数	p_msg: 表示メッセージ文字列へのポインタ len: 表示メッセージの長さ is_waiting: 送信完了待ちをするか否か
戻り値	MD_OK: 成功 MD_ERROR : 失敗
説　明	UART1 を使用して PC へメッセージを送信します。

r_SendSRecord_I2C	
概　要	ターゲットへデータ送信
書　式	MD_STATUS r_SendSRecord_I2C(uint8_t * p_buff, uint16_t length)
引　数	p_buff: 送信データを格納するバッファのポインタ length: 送信データのサイズ
戻り値	MD_OK: 成功 MD_ERROR : 失敗
説　明	IICA0 を使用して、ターゲット(スレーブ)へデータを送信します。

r_Receive1Byte_I2C	
概　要	ターゲットから 1 バイトデータ受信
書　式	MD_STATUS r_Receive1Byte_I2C(uint8_t * p_buff)
引　数	p_buff: 受信データを格納するバッファのポインタ
戻り値	MD_OK: 成功 MD_ERROR : 失敗
説　明	IICA0 を使用して、ターゲット(スレーブ)からデータを 1 バイト受信します。

r_Receive1Byte_I2C_Sparse	
概　要	ターゲットから 1 バイトデータ受信 (20ms ウェイトあり)
書　式	MD_STATUS r_Receive1Byte_I2C_Sparse(uint8_t * p_buff)
引　数	p_buff: 受信データを格納するバッファのポインタ
戻り値	MD_OK: 成功 MD_ERROR : 失敗
説　明	IICA0 を使用して、ターゲット(スレーブ)からデータを 1 バイト受信します。 ターゲットのステータス変化がない間、不要な I2C 通信を減らすため、受信処理の前に 20ms のウェイトを挿入します。

6.　ターゲット内部仕様

6.1　モード選択

ターゲット MCU は、ブート・プログラムのスタート・アップ・ルーチンで、RL78/G23-128p Fast Prototyping Board のユーザ・スイッチ(SW)の状態を確認し、SW が押されている場合(P137=Low)は書き換えモード(ブート・プログラムの処理)に移行し、SW が押されていない場合(P137=High)は、ユーザ・モード(ユーザ・プログラムの処理)に移行します。

6.2　オプション・バイト設定(ターゲット)

表 6-1 にターゲットのサンプル・プログラムで使用する、オプション・バイトの設定を示します。

ブート・プログラム側で設定します。

表 6-1　ターゲットのオプション・バイト設定

アドレス	設定値	内容
000C0H/040C0H	01101110B	ウォッチドッグ・タイマ動作停止
000C1H/040C1H	11111110B	LVD 検出電圧 : リセット・モード 立ち上がり時 TYP. 1.90 V (1.84 V ～ 1.95 V) 立ち下がり時 TYP. 1.86 V (1.80 V ～ 1.91 V)
000C2H/040C2H	11101000B	HS モード、 高速オンチップ・オシレータ・クロック : 32MHz
000C3H/040C3H	10000100B	オンチップ・デバッグ許可

RL78/G23 のオプション・バイトは、ユーザ・オプション・バイト(000C0H – 000C2H)とオンチップ・デバッグ・オプション・バイト(000C3H)で構成されています。

6.3　割り込みベクタの共有

ブート・プログラムとユーザ・プログラムは別プロジェクトとなっており、そのままでは割り込みベクタを共有できないため、ユーザ・プログラムの割り込みハンドラの関数をブート・プログラムから実行するための仕組みを用意しています。

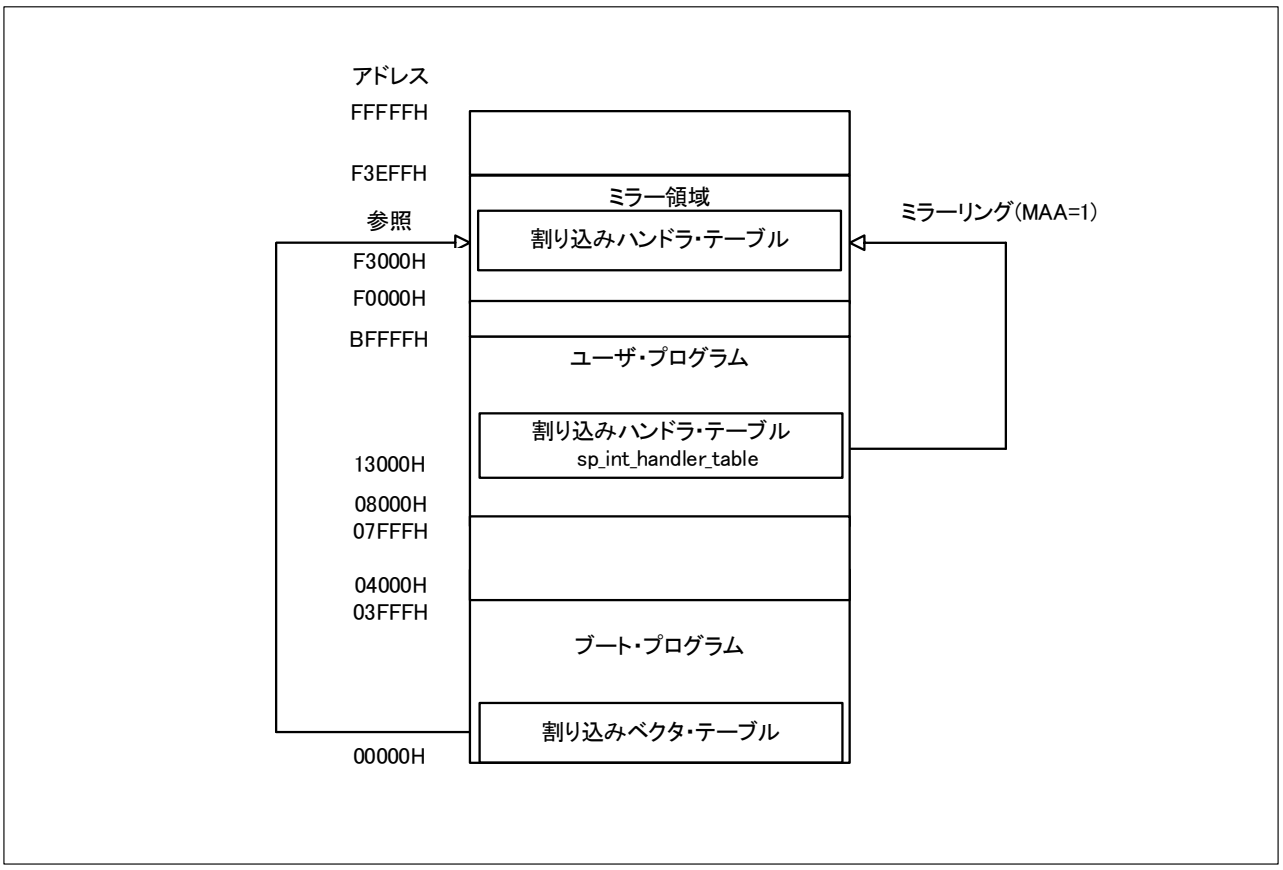
ユーザ・プログラムでは、ユーザ・プログラムで使用する割り込みハンドラを登録するための割り込みハンドラ・テーブルを固定番地(0x13000)に配置します。

また、ユーザ・プログラムのスタート・アップ・ルーチンで、PMC レジスタの MAA を 1 に設定し、0xF3000 番地に割り込みハンドラ・テーブルがミラーリングされるようにします。

上記によりブート・プログラムでは、ブート・プログラムの各割り込みハンドラで、0xF3000 番地にあるユーザ・プログラムの割り込みハンドラ・テーブルを参照して、対応するベクタ・アドレスにジャンプする処理を実装しています。

また、ブート・プログラムでは IICA0 の割り込み(INTIICA0)を使用しているため、ブート・プログラムの IICA0 の割り込みハンドラでは、MAA=0 の場合はブート・プログラム用の IICA0 の割り込み処理を実行し、MAA=1 の場合はユーザ・プログラムの割り込みハンドラ・テーブルを参照してユーザ・プログラムの IICA0 のベクタ・アドレスにジャンプする処理を実装しています。

図 6-1 割り込みベクタの共有



6.3.1　ユーザ・プログラムの割り込みハンドラの登録

ユーザ・プログラム用の割り込みハンドラ・テーブルは、ユーザ・プログラムのプロジェクト内の int_handlers.c で定義しています。

sp_int_handler_table へユーザ・プログラム側で使用する割り込みハンドラを追加してください。

sp_int_handler_table[0]には、ユーザ・プログラム開始アドレスを指定します。

サンプル・プログラムでは、LED 点滅処理で INTTM01 を使用しています。

図 6-2　割り込みハンドラ・テーブル

```
/***********************************************************************************************************************
Macro definitions
***********************************************************************************************************************/
#define USER_PROGRAM_START                  (0x8000)


/***********************************************************************************************************************
Static variables and functions
***********************************************************************************************************************/
#pragma section const USER_INT_HANDLERS
const int_handler_t sp_int_handler_table[64] =
{
    (int_handler_t)(USER_PROGRAM_START),    /* RST (Note: This must be a start point of your program.) */
    NULL,                                   /* INTDBG(do't set a handler here!) */
    NULL,                                   /* INTWDTI */
    NULL,                                   /* INTLVI */
    NULL,                                   /* INTP0  */
    NULL,                                   /* INTP1  */
    NULL,                                   /* INTP2  */
    NULL,                                   /* INTP3  */
(省略)
    R_Config_TAU0_1_interrupt,              /* INTTM01 */
    NULL,                                   /* INTTM14  */
    NULL,                                   /* INTTM15  */
    NULL,                                   /* INTTM16  */
    NULL,                                   /* INTTM17  */
    NULL,                                   /* No Use  */
    NULL,                                   /* No Use  */
    NULL,                                   /* BRK_I */
};
#pragma section
```

6.3.2　ブート・プログラムの割り込みハンドラ

ブート・プログラムでの各割り込みハンドラはユーザ・プログラムの割り込みハンドラ・テーブルを参照して対応するベクタ・アドレスへジャンプする処理を実装しています。

ブート・プログラムのプロジェクト内の int_handlers.c, int_handlers_tmpl.h で定義しています。

図 6-3　ブート・プログラムの割り込みハンドラ (int_handlers.c)

```
#define INT_VECT     INTWDTI
#define R_HANDLER    r_handler_INTWDTI_vect
#include "int_handlers_tmpl.h"

#define INT_VECT     INTLVI
#define R_HANDLER    r_handler_INTLVI_vect
#include "int_handlers_tmpl.h"

#define INT_VECT     INTP0
#define R_HANDLER    r_handler_INTP0_vect
#include "int_handlers_tmpl.h"

#define INT_VECT     INTP1
#define R_HANDLER    r_handler_INTP1_vect
#include "int_handlers_tmpl.h"

#define INT_VECT     INTP2
#define R_HANDLER    r_handler_INT2_vect
#include "int_handlers_tmpl.h"

(省略)

#define INT_VECT     INTTM16
#define R_HANDLER    r_handler_INTTM16_vect
#include "int_handlers_tmpl.h"

#define INT_VECT     INTTM17
#define R_HANDLER    r_handler_INTTM17_vect
#include "int_handlers_tmpl.h"

/* software break handler */
#define INT_VECT     0x7E
#define R_HANDLER    r_handler_BRK_vect
#include "int_handlers_tmpl.h"
```

図 6-4　ブート・プログラムの割り込みハンドラ (int_handlers_tmpl.h)

```
#ifdef R_HANDLER
#if INT_VECT == 0x7E
#pragma interrupt_brk R_HANDLER
#else
#pragma interrupt R_HANDLER(vect=INT_VECT)
#endif

static void __near R_HANDLER(void)
{
    int_handler_t __far * int_handler_table= INT_HANDLER_ADDRESS;

    if (int_handler_table[INT_VECT>>1])
    {
        int_handler_table[INT_VECT>>1]();
    }
}

#undef R_HANDLER
#undef INT_VECT

#endif
```

6.4　スマートコンフィグレータの設定(ブート・プログラム)

ブート・プログラムのスマートコンフィグレータでの設定を以下に示します。

(1)　ポートの初期設定を行います。

- P50, P51 を出力モードに設定

(2)　I2C 通信(スレーブ・モード)の初期設定を行います。

- IICA0 を使用 (P60 を SCLA0 に、P61 を SDAA0 に設定)
- 動作クロックを fCLK/2 に設定
- 自局アドレス設定のアドレスを 160 に設定
- 動作モードを標準に設定
- ウェイク・アップ機能設定をオフに設定
- 通信完了割り込み設定(INTIICA0)をレベル 3 に設定
- コールバック機能設定のスレーブ送信完了、スレーブ受信完了、スレーブ・エラーを選択

(3)　電圧検出回路の初期設定を行います。

- リセット・モードを選択
- リセット発生電圧(VLVD0)に 1.86V を選択

6.5　セクション設定(ターゲット)

表 6-2 にブート・プログラム、表 6-3 にユーザ・プログラムの CC-RL のセクション設定を示します。

- 「.」から始まるセクションはコンパイラの予約セクションです。
- 「RFD_」「SMP_」から始まるセクションは Renesas Flash Driver RL78 Type01(RFD)とそのサンプル・プログラムの仕様で規定されているセクションです。
- IAR は予約セクション名が異なりますが、基本的に同様のセクション構成になっています。

表 6-2　ブート・プログラム セクション一覧

セクション名	アドレス	説明
.text	0x000D8	コード部用セクション(near 領域配置)
.RLIB	(コンパイラ自動配置)	ランタイム・ライブラリ　コード用セクション
.SLIB	(コンパイラ自動配置)	標準ライブラリ　コード用セクション
.textf	(コンパイラ自動配置)	コード部用セクション(far 領域配置)
.constf	(コンパイラ自動配置)	ROM データ(far 領域配置)
RFD_DATA_n	(コンパイラ自動配置)	RFD 用データ
RFD_CMN_f	(コンパイラ自動配置)	RFD 用共通コード
RFD_CF_f	(コンパイラ自動配置)	RFD 用コード・フラッシュ用コード
RFD_DF_f	(コンパイラ自動配置)	RFD 用データ・フラッシュ用コード
RFD_EX_f	(コンパイラ自動配置)	RFD 用データ・フラッシュ用コード
SMP_CMN_f	(コンパイラ自動配置)	RFD サンプル 共通コード
SMP_CF_f	(コンパイラ自動配置)	RFD サンプル コード・フラッシュ用コード
SMP_CMN_n	(コンパイラ自動配置)	SMP 用共通コード
SMP_DF_f	(コンパイラ自動配置)	RFD サンプル データ・フラッシュ用コード
.data	(コンパイラ自動配置)	near 初期化データ用セクション(初期値あり)
.sdata	(コンパイラ自動配置)	初期化データ用セクション(初期値あり，saddr 配置変数)
.dataR	0xF3F00	near 初期化データ用セクション(初期値あり) (RAM 展開)
.bss	(コンパイラ自動配置)	データ領域用セクション(初期値なし，near 領域配置)
RFD_DATA_nR	(コンパイラ自動配置)	RFD 用データ(RAM 展開)
RFD_CMN_fR	(コンパイラ自動配置)	RFD 用共通コード(RAM 展開)
RFD_CF_fR	(コンパイラ自動配置)	RFD 用コード・フラッシュ用コード(RAM 展開)
RFD_EX_fR	(コンパイラ自動配置)	RFD エクストラ領域用コード(RAM 展開)
SMP_CMN_fR	(コンパイラ自動配置)	RFD サンプル 共通コード(RAM 展開)
SMP_CF_fR	(コンパイラ自動配置)	RFD サンプル コード・フラッシュ用コード (RAM 展開)
.sdataR	0xFFE20	初期化データ用セクション(初期値あり，saddr 配置変数)(RAM 展開)
.sbss	(コンパイラ自動配置)	データ領域用セクション(初期値なし，saddr 配置変数)

表 6-3　ユーザ・プログラム セクション一覧

セクション名	アドレス	説明
.text	0x08000	コード部用セクション(near 領域配置)
.RLIB	(コンパイラ自動配置)	ランタイム・ライブラリ　コード用セクション
.SLIB	(コンパイラ自動配置)	標準ライブラリ　コード用セクション
.textf	(コンパイラ自動配置)	コード部用セクション(far 領域配置)
.constf	(コンパイラ自動配置)	ROM データ(far 領域配置)
.data	(コンパイラ自動配置)	near 初期化データ用セクション（初期値あり）
.sdata	(コンパイラ自動配置)	初期化データ用セクション(初期値あり，saddr 配置変数)
USER_INT_HANDLERS_n	0x13000	ユーザ・プログラム用割り込みハンドラ・テーブル領域
.const	(コンパイラ自動配置)	ROM データ（near 領域配置）（ミラー領域内）
.dataR	0xF3F00	near 初期化データ用セクション(初期値あり)(RAM 展開)
.bss	(コンパイラ自動配置)	データ領域用セクション(初期値なし, near 領域配置)
.sdataR	0xFFE20	初期化データ用セクション(初期値あり，saddr 配置変数)(RAM 展開)
.sbss	(コンパイラ自動配置)	データ領域用セクション(初期値なし，saddr 配置変数)

6.5.1　ターゲット用プログラム・ファイルの作成

本サンプル・プログラムでは、ブート・スワップに対応するため、ブート・プログラムはアドレス 0x00000～0x03FFF、ユーザ・プログラムはアドレス 0x08000～0xBFFFF に割り当てています。

ブート・プログラムの場合は、ビルド設定によりコード・フラッシュの最終領域に OCD モニター用のデータが追加されるため、0x00000～0x03FFF のデータのみを使用する必要があります。

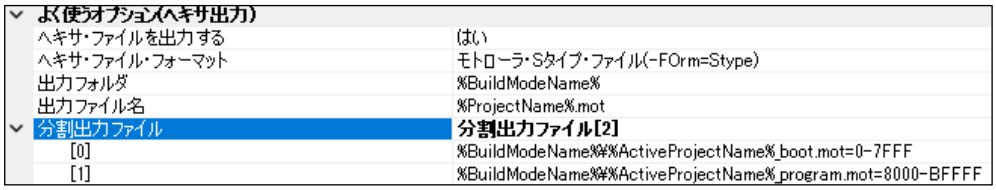
ユーザ・プログラムの場合は、セクション設定に関わらず 0x08000 未満にデータ(割り込みベクタ・テーブル、オプション・バイト、セキュリティ ID、OCD モニター)が自動生成されるため、これらを削除します。

CS+と e2studio では、CC-RL の分割出力ファイル機能により、上記の必要なデータのみを出力する設定を有効にしています。

ブート・プログラムは rl78g23_RFD_with_I2C_boot.mot, ユーザ・プログラムは rl78g23_FlashWriter_UserProgramSample_program.mot を使用してください。

図 6-5 ブート・プログラムの分割出力ファイル設定 (CS+の例)

図 6-6 ユーザ・プログラムの分割出力ファイル設定 (CS+の例)



IAR の場合は分割出力ファイル機能がないため、SRecordSplitter.bat を使用します。

(rl78g23_FlashWriter_UserProgramSample¥Workspace¥IAREW¥にあります)

コマンドプロンプトで、以下を実行することで、0x08000 以降のレコードのみが抽出されます。

表 6-4　SRecordSplitter.bat の使用方法

> ./SRecordSplitter.bat <元の S-Record ファイル> <新しい S-Record ファイル>

6.6　定義値(ターゲット)

SRECORD_DATA 構造体は、モトローラ S フォーマットの S レコード解析データを一時保存するために使用します。

表 6-5 に SRECORD_DATA 構造体の仕様を示します。

表 6-5　SRECORD_DATA 構造体

メンバ	説明
SRECORD_TYPE type	モトローラ S レコードのタイプ
uint32_t address	モトローラ S レコ―ドのアドレス
uint16_t size	モトローラ S レコードのサイズ
uint8_t data[SRECORD_DATA_SIZE]	モトローラ S レコードのデータ (SRECORD_DATA_SIZE=128)

6.7　関数仕様 (ブート・プログラム)

6.7.1　関数一覧

表 6-6 にサンプル・プログラム(ブート・プログラム)で使用する主な関数を示します。

表 6-6　ブート・プログラム関数一覧

関数名	概要
R_Decode_SRecord	Sレコードの解析
R_RFDCmd_Initialize	RFDの初期化
R_RFDCmd_Write	フラッシュ・メモリの書き換え
R_RFDCmd_End	ブート・スワップの実行

6.7.2　関数説明

サンプル・プログラム(ブート・プログラム)で使用する主な関数の仕様を示します

R_Decode_SRecord	
概　要	S レコードの解析
書　式	void R_Decode_SRecord(uint8_t * srecord_str, uint16_t len, SRECORD_DATA * srecord_data)
引　数	srecord_str: S レコード文字列(1 行) len: srecord_str の長さ srecord_data: S レコードの解析データ
戻り値	なし
説　明	S レコードの解析を行い、SRECORD_DATA 構造体に解析結果を設定します。

R_RFDCmd_Initialize	
概　要	RFD の初期化
書　式	e_rfd_ret_t R_RFDCmd_Initialize(void)
引　数	なし
戻り値	R_RFD_ENUM_RET_STS_OK: 成功 その他: 失敗
説　明	R_RFD_Init ()を実行し RFD の初期化を行います。

R_RFDCmd_Write	
概　要	フラッシュ・メモリの書き換え
書　式	R_RFDCmd_Write(uint32_t start_addr, uint16_t length, uint8_t __near * p_data)
引　数	start_addr: 書き込み開始アドレス length: データサイズ p_data: 書き込みデータ
戻り値	R_RFD_ENUM_RET_STS_OK: 成功 その他: 失敗
説　明	以下処理を行いフラッシュ・メモリのデータを書き換えます。 1. 割り込みベクタを RAM 上に設定 2. フラッシュ・メモリ制御モードをコード・フラッシュ・プログラミング・モードまたはデータ・フラッシュ・プログラミング・モードに設定 3. ブランク・チェックの実行 4. ブロック消去の実行 (非ブランクの場合) 5. 書き込みの実行 6. フラッシュ・メモリ制御モードを非書き換えモードに設定 7. ベリファイの実行 (書き込みデータと読み出しデータの比較) 8. 割り込みベクタを ROM に戻す

R_RFDCmd_End	
概　要	ブート・スワップの実行
書　式	e_rfd_ret_t R_RFDCmd_End(void)
引　数	なし
戻り値	R_RFD_ENUM_RET_STS_OK: 成功 その他:失敗
説　明	ブート・スワップを実行後、ソフトウェアリセットを行います。

6.7.3　注意事項

本サンプル・プログラムを、Board support package (bsp)のバージョン変更後にコード再生成を行うと、エラーとなる場合があります。新たに生成されたファイル内にある、cstart.asm に以下の記述を追加してください。

図 6-7　ブート・プログラムの cstart.asm

```
(省略)
;-------------------------------------------------------------------------------
;   startup
;-------------------------------------------------------------------------------
.SECTION .text, TEXT
_start:
;===============================================================================
; Jump to a user-program if SW is off(P137 == 1)
PORT13              .EQU  0xFFF0D
USER_INT_HANDLERS   .EQU  0x13000
        BF PORT13.7, $_jump_user_program_end
        MOV  ES, #HIGHW(USER_INT_HANDLERS)
        MOVW HL, #LOWW(USER_INT_HANDLERS)
        MOVW AX, ES:[HL+0]
        BR AX
_jump_user_program_end:
;===============================================================================

    ;---------------------------------------------
    ; setting register bank
    ;---------------------------------------------
$IFDEF BSP_CFG_ASM_RAM_GUARD_START_ADDRESS
    MOV !RAMSAR, #BSP_CFG_ASM_RAM_GUARD_START_ADDRESS
$ENDIF
(省略)
```

6.8　関数仕様 (ユーザ・プログラム)

6.8.1　関数一覧

表 6-7 にサンプル・プログラム(ユーザ・プログラム)で使用する主な関数を示します。

表 6-7　ユーザ・プログラム関数一覧

関数名	概要
main	メイン関数

6.8.2　関数説明

サンプル・プログラム(ユーザ・プログラム)で使用する主な関数の仕様を示します。

main	
概　要	メイン関数
書　式	void main(void)
引　数	なし
戻り値	なし
説　明	インターバル・タイマ(INTTM01)を使用して LED の点滅を行います。

6.8.3　注意事項

本サンプル・プログラムを、Board support package (bsp)のバージョン変更後にコード再生成を行うと、エラーとなる場合があります。新たに生成されたファイル内にある、cstart.asm でミラー領域の設定部分を以下のように変更してください。

図 6-8　ユーザ・プログラムの cstart.asm

```
(省略)

    ;---------------------------------------------------
    ; setting mirror area
    ;---------------------------------------------------
   ONEB    !PMC        ; mirror area = 10000-1FFFFH

    ;---------------------------------------------------
    ; setting the stack pointer
    ;---------------------------------------------------

(省略)
```

7. 参考ドキュメント

RL78/G23 ユーザーズマニュアル ハードウェア編 (R01UH0896)

Renesas Flash Driver RL78 Type01 ユーザーズマニュアル (R20UT4830)

Renesas Flash Driver RL78 Type01 SC 対応仕様(Code Flash) (R20AN0653)

Renesas Flash Driver RL78 Type01 SC 対応仕様 (Data Flash) (R20AN0654)

RL78 ファミリ Renesas Flash Driver RL78 Type01 SC 対応版 (Extra Area) (R20AN0655)

RL78 ファミリ Renesas Flash Driver RL78 Type01 SC 対応仕様(Common) (R20AN0656)

RL78/G23-128p Fast Prototyping Board ユーザーズマニュアル (R20UT4870)

(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	2025.08.06	—	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子(または外部発振回路)を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子(または外部発振回路)を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}(Max.)から V_{IH}(Min.)までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}(Max.)から V_{IH}(Min.)までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス(予約領域)のアクセス禁止
   リザーブアドレス(予約領域)のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス(予約領域)があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害(お客様または第三者いずれに生じた損害も含みます。以下同じです。)に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器(自動車、電車、船舶等)、交通制御(信号)、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム(生命維持装置、人体に埋め込み使用するもの等)、もしくは多大な物的損害を発生させるおそれのある機器・システム(宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等)に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害(当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。)から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為(「脆弱性問題」といいます。)によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報(データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等)をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24(豊洲フォレシア)

www.renesas.com

お問合せ窓口

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

www.renesas.com/contact/

商標について

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