RL78 ファミリ

Raspberry Pi によるフラッシュプログラマ（RL78 プロトコル A 編）

要旨

本アプリケーションノートは、RL78 プロトコル A に対応したマイクロコントローラのフラッシュ・メモリの書き込みを行う簡易プログラマのサンプル・プログラムについて説明します。

動作確認デバイス

RL78/G13

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 マイクロコントローラ(RL78 プロトコル A) プログラマ編 (R01AN0815)

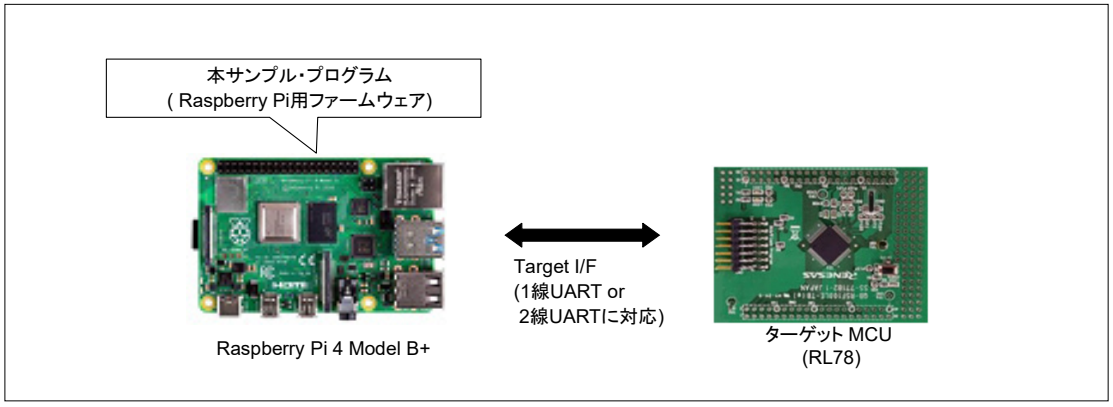
目次

1.　概要

本サンプル・プログラムは、RL78 マイクロコントローラ内蔵のフラッシュ・メモリの書き込みを行うための Raspberry Pi 用サンプル・プログラムであり、以下の特徴があります。

- 書き込み対象のマイクロコントローラ(ターゲット MCU)は、「RL78 プロトコル A」に対応した RL78 とします。
- RL78 プロトコル A のシリアルプログラミングにより書き込みを行います。
- Raspberry Pi 4 Model B+をフラッシュプログラマのハードウェアとして使用します。
- プログラムファイル(書き込み用データ)はモトローラ S フォーマットに対応します。

図 1-1　使用イメージ



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。フラッシュプログラマ(Raspberry Pi4 Model B+)は、PC によるリモート接続や、直接モニタなどの周辺機器を接続したスタンドアローンなどの接続方法があります。

表 2-1　動作確認条件

開発ツール	説明
フラッシュプログラマ	Raspberry Pi4 Model B+ (内蔵 RAM 4GB)
OS	Raspberry Pi OS 64-bit(version 5.10.17)
使用言語	C99
ソフトウェアビルド環境	gcc：8.3.0 (Raspbian 8.3.0-6+rpi1)
コンパイラ	make：GNU Make 4.2.1
共有ライブラリ	ldd：2.28(Debian GLIBC 2.28-10+rpi1)

注意　上記のバージョン以外では動作しない可能性があります。

3. Raspberry Pi の設定

3.1 Raspberry Pi 環境(config.txt)の設定

UART2～5 を使用できるようにするため、/boot/config.txt の中にある[all]の下に以下の項目を追記して、再起動します。

```
enable_uart=1
dtoverlay=uart1
dtoverlay=uart2
dtoverlay=uart3
dtoverlay=uart4
dtoverlay=uart5
```

3.2 ビルド環境の構築

ビルド環境を最新にしたい場合は、以下のコマンドを実行します。

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

以下のオプションで gcc と make のコマンドを確認することができます。

```
$ gcc --v
```

3.3 ビルド方法

ビルドをするには、makefile のあるディレクトリで以下のコマンドを実行します。

```
$ sudo make ALL　(ビルドを行う)
$ sudo make clean　(ビルドした実行バイナリを削除する)
```

4. 仕様

本サンプル・プログラムでは、フラッシュプログラマ（Raspberry Pi4 Model B+）で実行ファイル ”fp_a”を実行し、フラッシュプログラマ内のモトローラ S フォーマットファイル（書き込み用データ）をターゲット MCU に書き込みます。

4.1 オプション仕様

以下の仕様に従い、初期設定を行いターゲットの通信を行います。

- フラッシュプログラマは、指定されたオプションの設定で実行が成功した場合は、”OK”をターミナルに送信します。
- フラッシュプログラマは、指定されたオプションの設定で実行が失敗した場合は、”ERROR:XX”をターミナルに送信します。XX は 2 桁の 16 進数で表示されます。詳しくは表 4-2、表 4-3 を参照してください。

表 4-1 にオプションの詳細を示し、図 4-1 と図 4-2 にオプションの使用例を示します。

表 4-1 オプション仕様

ロングオプション	ショートオプション	設定	説明
--file=	-f	ファイル名.mot	S-Record ファイルを指定します。
--if=	-u	uart1	・uart1 1 線 UART(TOOL0)でターゲット MCU と通信を行います。 ・uart2 2 線 UART(TOOL0, TOOLTxD, TOOLRxD)でターゲット MCU と通信を行います。 本オプションを省略した場合は、uart1 として動作します。
		uart2	
--speed=	-b	115200	RL78 プロトコル A の Baud Rate Set コマンドで設定する通信速度(bps)を指定します。 本オプションを省略した場合は、115200 として動作します。
		250000	
		500000	
		1000000	
--vdd=	-d	x.x (10 進数, 整数 1 桁, 少数第 1 位)	RL78 プロトコル A の Baud Rate Set コマンドで設定する V_{DD} 印可電圧値(V)を指定します。 フラッシュプログラマとターゲット MCU に供給している V_{DD} の電圧値を設定してください。 本オプションを省略した場合は、3.3 として動作します。
--verify	-v	-	本オプションを指定すると、ベリファイを追加で実行します。
--checksum	-s	-	本オプションを指定すると、チェックサムを追加で実行します。

図 4-1　ロングオプションの使用例（実行ファイル名 fp_a）

```
>    sudo ./fp_a --file=test.mot --if=uart1 --vdd=3.3 --verify --checksum
OK:connect
OK:erase
OK:program,verify
OK:checksum
 code flash:xxxx
 data flash:xxxx
```

図 4-2　ショートオプションの使用例（実行ファイル名 fp_a）

```
>    sudo ./fp_a -ftest.mot -uuart1 -d3.3 -v -s
OK:connect
OK:erase
OK:program,verify
OK:checksum
 code flash:xxxx
 data flash:xxxx
```

4.2　エラーコード仕様

実行ファイルの実行が失敗した際、ターミナルに“Error:XX”の書式でエラーメッセージを表示します。XX が 2 桁の 16 進数の場合、エラーコードを参照してください。

表 4-2 と表 4-3 にエラーコードを示します。

表 4-2　エラーコードの説明(1/2)

エラーコード (16 進数)	説明
04	コマンド番号エラー ターゲット MCU から RL78 プロトコル A のステータス・コードのコマンド番号エラーを受信した場合のエラーです。
05	パラメーターエラー ターゲット MCU から RL78 プロトコル A のステータス・コードのパラメーターエラーを受信した場合のエラーです。
07	チェックサムエラー ターゲット MCU から RL78 プロトコル A のステータス・コードのチェックサムエラーを受信した場合のエラーです。
0F	ベリファイエラー ターゲット MCU から RL78 プロトコル A のステータス・コードのベリファイエラーを受信した場合のエラーです。
10	プロテクトエラー ターゲット MCU から RL78 プロトコル A のステータス・コードのプロテクトエラーを受信した場合のエラーです。
15	NACK ターゲット MCU から RL78 プロトコル A のステータス・コードの NACK を受信した場合のエラーです。
1A	消去エラー ターゲット MCU から RL78 プロトコル A のステータス・コードの消去エラーを受信した場合のエラーです。
1B	ブランクエラー ターゲット MCU から RL78 プロトコル A のステータス・コードのブランクエラーを受信した場合のエラーです。
1C	書き込みエラー ターゲット MCU から RL78 プロトコル A のステータス・コードの書き込みエラーを受信した場合のエラーです。

表 4-3　エラーコードの説明(2/2)

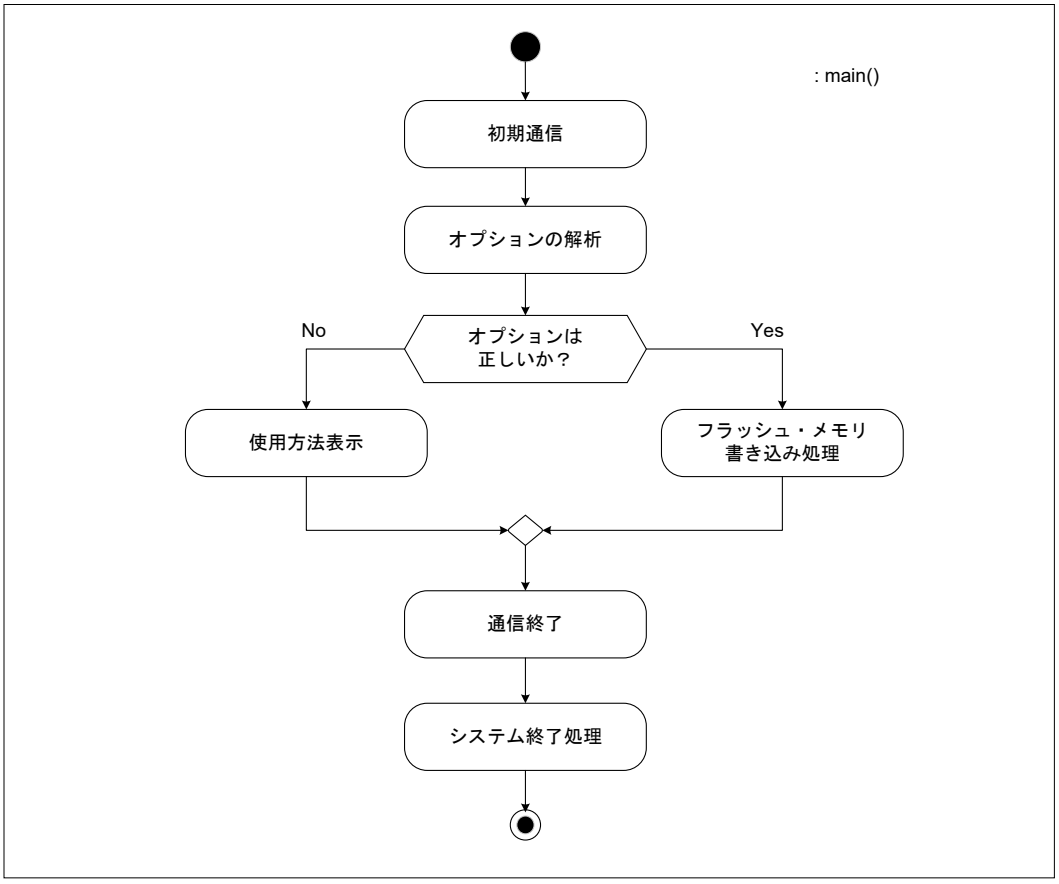
エラーコード (16 進数)	説明
FB	モトローラ S フォーマットのデータが不正 ターゲット MCU へモトローラ S フォーマットが不正の場合に発生します。 モトローラ S フォーマットのデータレコードがアドレス昇順になっていない場合も本エラーが発生します。
FC	ターゲット MCU 通信タイムアウト発生 フラッシュプログラマとターゲット MCU との通信でタイムアウトが発生した場合に発生します。
FE	コマンド通信データエラー ターゲット MCU から受信したパケットフォーマットが不正の場合に発生します。
FF	システムエラー 正常にプログラムが動作しなかった場合に発生します。

4.3　フローチャート

4.3.1　メインルーチン

図 4-3 にメインルーチンの動作を示します。

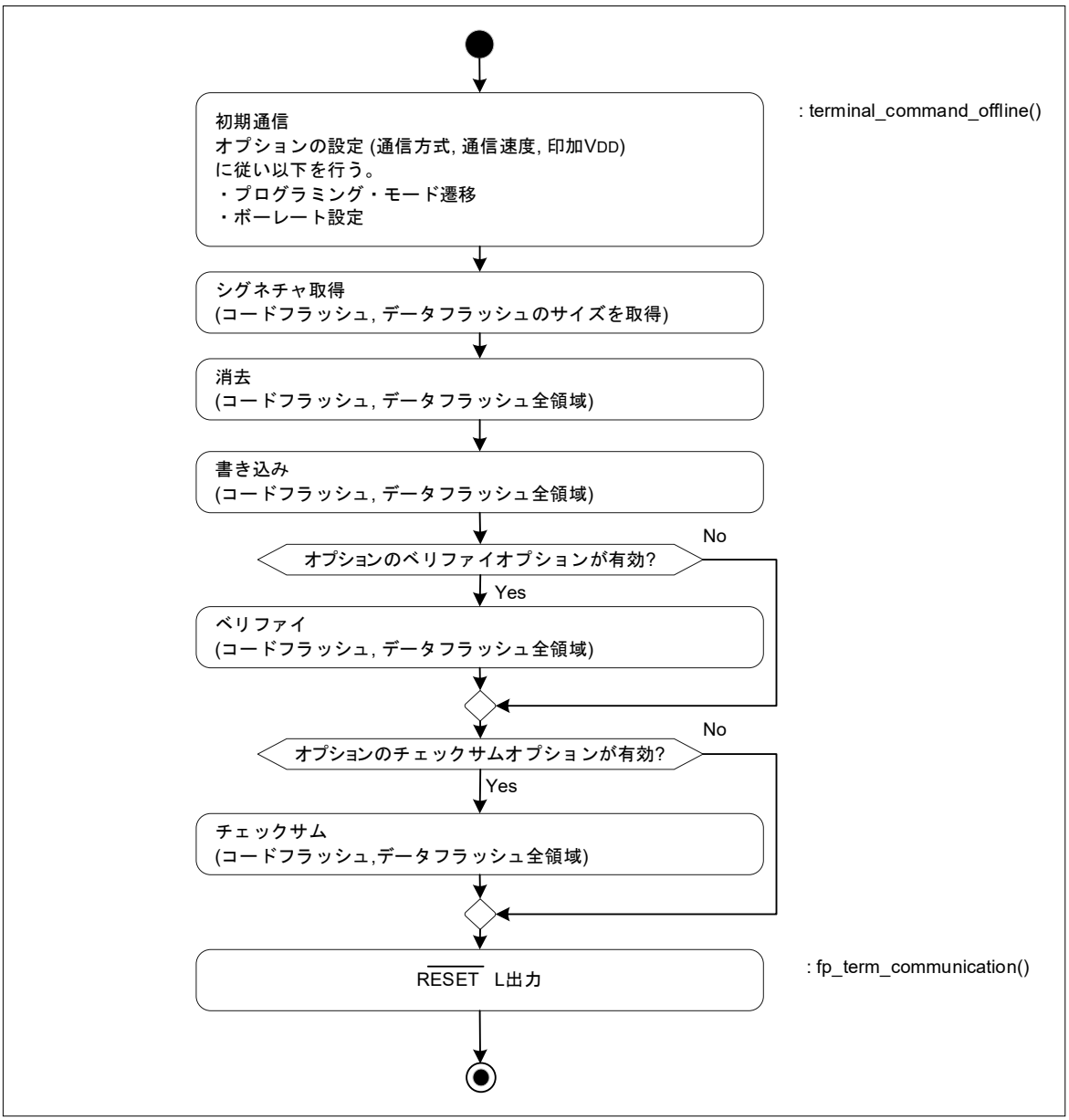
図 4-3　メインルーチン



4.3.2　フラッシュ・メモリ書き込み処理

図 4-4 にフラッシュ・メモリ書き込み処理の流れを示します。

図 4-4　フラッシュ・メモリ書き込み処理



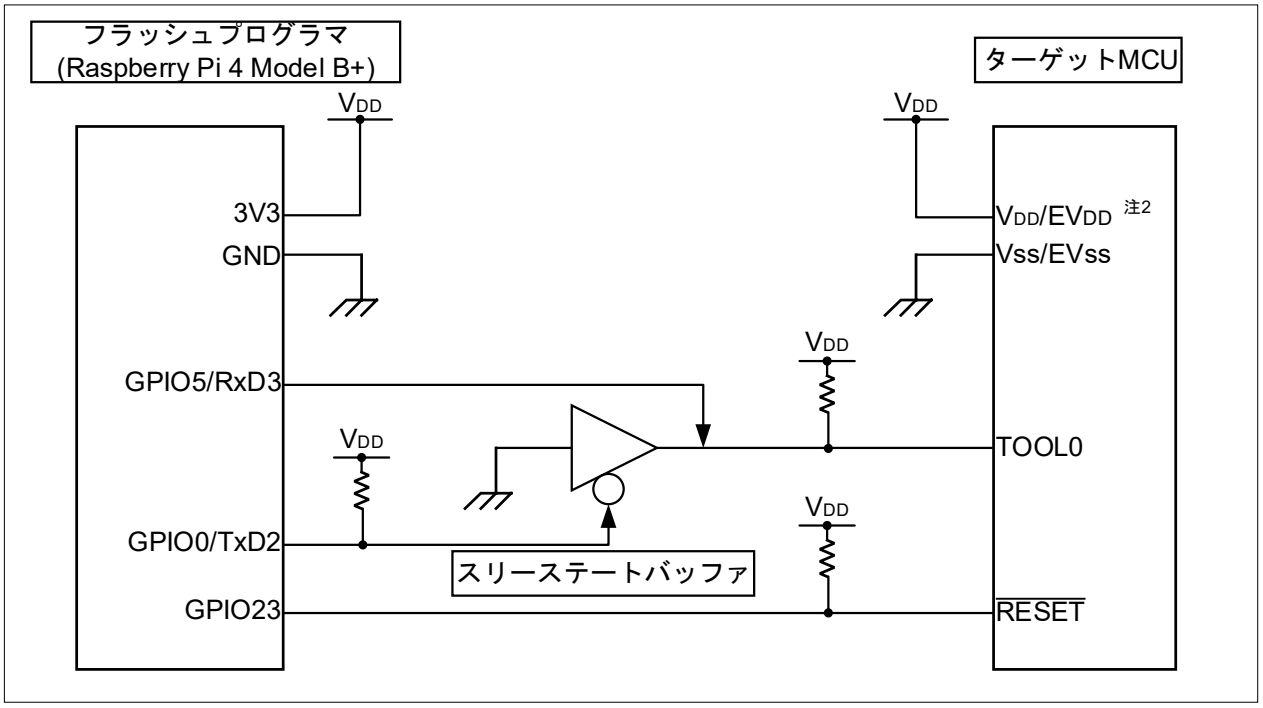
5. ハードウェア説明

5.1 ターゲットインタフェース仕様

フラッシュプログラマとターゲット MCU の接続方法を以下に示します。

5.1.1 1 線 UART

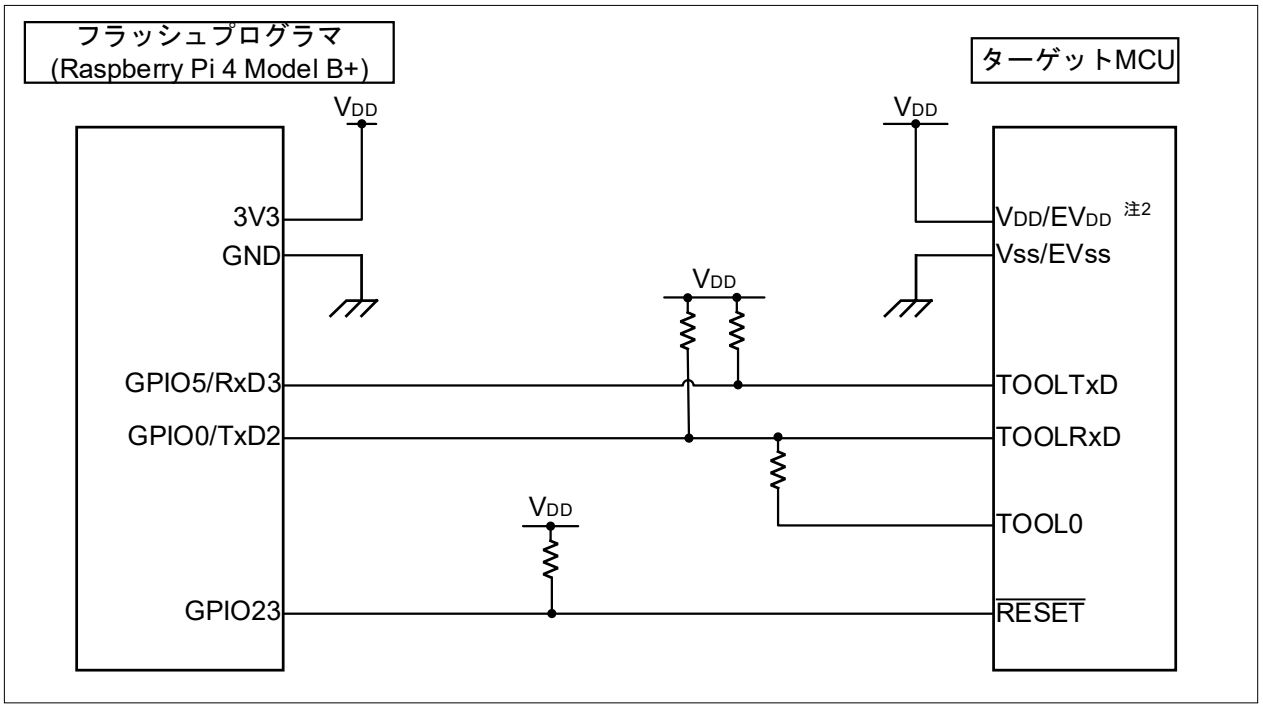
図 5-1 1 線 UART(V_{DD}=EV_{DD}) 注 1



注 1. V_{DD} は 3.3V の場合の接続例です。
注 2. V_{DD} と EV_{DD} が異なる場合は EV_{DD} に外部電源を供給する必要があります。

5.1.2　2 線 UART

図 5-2　2 線 UART(V_{DD}=EV_{DD})[注 1]



注 1.　V_{DD} は 3.3V の場合の接続例です。
注 2.　V_{DD} と EV_{DD} が異なる場合は EV_{DD} に外部電源を供給する必要があります。

5.2　使用端子一覧

表 5-1 にサンプル・プログラムで使用するフラッシュプログラマの端子と機能を示します。

表 5-1　使用端子一覧

端子名	入出力	機能
TxD2	出力	ターゲットインタフェース通信用送信端子(UART2) 注 1
RxD3	入力	ターゲットインタフェース通信用受信端子(UART3) 注 1
GPIO23	出力	ターゲット MCU の$\overline{\text{RESET}}$制御用端子

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。

注 1.　RL78 のプロトコル A の通信仕様では送信と受信のストップビットが異なります。Raspberry Pi OS の UART の通信設定では送信と受信のストップビットが共通になってしまうことから、送信と受信で使用する UART を分けています。

6. ソフトウェア説明

6.1 ファイル一覧

サンプル・プログラムで使用するファイルの一覧を示します。

表 6-1 に Raspberry Pi OS で提供されるファイルを示し、表 6-2 にサンプル・プログラムで提供するファイルを示します。

表 6-1 Raspberry Pi OS で提供されるファイル一覧

ディレクトリ	ファイル名	説明
/dev/	mem	メモリマップドI/Oファイル
/dev/	ttyAMA1またはttyAMA2[注1]	シリアル通信ポートファイル UART2に対応
/dev/	ttyAMA2またはttyAMA3[注1]	シリアル通信ポートファイル UART3に対応
/boot/	config.txt firmware/config.txt	RPi4のconfig設定ファイル

注1. OS のバージョンにより異なります。

表 6-2 サンプル・プログラムで提供するファイル一覧

ディレクトリ	ファイル名	説明
./	fp_a	makeで作成される、プログラム実行ファイル (2.開発環境で作成したものを添付しています。) 別の場所からコピーして使用する場合は、以下のコマンドで実行権限の付与が必要になることがあります。 $ chmod a+x <ファイル名>"
./	makefile	makefileのサンプル (makeコマンドの実行手順を記述したテキストファイル)
./	main.c	メイン関数処理
common/	protocol_a.c	プロトコルAコマンド処理
common/	terminal_com.c	ターミナルコマンド処理
common/	utility.h	ユーティリティ関数処理
driver/	config_driver.c config_driver.h	システムの初期化機能処理
driver/	config_gpio.c config_gpio.h	GPIO用のデバイス・ドライバ
driver/	config_systemtimer.c config_systemtimer.h	システム・タイマーのデバイス・ドライバ
driver/	config_uart.c config_uart.h	UART用のデバイス・ドライバ

6.2　関数一覧

表 6-3 と表 6-4 にサンプル・プログラムで使用する主な関数を示します。

表 6-3　関数一覧(1/2)

関数名	概要	ソースファイル
main	メイン関数	main.c
read_arguments	オプション引数の解析	main.c
system_init	システム初期設定処理	config_driver.c
system_term	システム終了処理	config_driver.c
config_gpio_create	GPIOレジスタのメモリマップドI/O取得	config_gpio.c
config_gpio_destroy	GPIOレジスタのメモリマップドI/O破棄	config_gpio.c
config_gpio_p23_output_start	GPIO23をOutput設定にする	config_gpio.c
config_gpio_p23_output_stop	GPIO23を初期設定に戻す	config_gpio.c
config_gpio_p0_txd2_start	GPIO0をTXD2設定にする	config_gpio.c
config_gpio_p0_txd2_stop	GPIO0をOutput設定にする	config_gpio.c
config_gpio_control_reset	GPIO23($\overline{\text{RESET}}$)のHI/LO制御	config_gpio.c
config_gpio_control_tool0	GPIO0(TOOL0)のHI/LO制御	config_gpio.c
config_systemtimer_create	SystemTimerレジスタのメモリマップドI/Oを取得	config_systemtimer.c
config_systemtimer_destroy	SystemTimerレジスタのメモリマップドI/Oを破棄	config_systemtimer.c
config_systemtimer_get_count	SystemTimerのカウンタ値を取得	config_systemtimer.c
config_systemtimer_wait_ms	ms単位でウェイト	config_systemtimer.c
config_systemtimer_wait_us	us単位でウェイト	config_systemtimer.c
config_uart_create	UARTレジスタのメモリマップドI/Oを取得	config_uart.c
config_uart_destroy	UARTレジスタのメモリマップドI/Oを破棄	config_uart.c
config_uart2_start	UART2の初期設定	config_uart.c
config_uart2_stop	UART2の設定破棄	config_uart.c
config_uart3_start	UART3の初期設定	config_uart.c
config_uart3_stop	UART3の設定破棄	config_uart.c
config_uart2_send	TXD2からデータ送信	config_uart.c
config_uart2_send_with_wait	TXD2からデータ送信(byte間ウェイトあり)	config_uart.c
config_uart3_receive	RXD3からデータ受信	config_uart.c
config_uart23_set_baudrate	UART2,UART3のボーレート設定	config_uart.c
config_uart2_set_send_wait_time	config_uart2_send_with_waitのウェイト時間設定	config_uart.c
fp_cmd_reset_a	Resetコマンド送信処理	protocol_a.c
fp_cmd_verify_a	Verifyコマンド送信処理	protocol_a.c
fp_cmd_erase_a	Block Eraseコマンド送信処理	protocol_a.c
fp_cmd_program_a	Programmingコマンド送信処理	protocol_a.c
fp_cmd_baudrate_a	Baud Rate Setコマンド送信処理	protocol_a.c

表 6-4　関数一覧(2/2)

関数名	概要	ソースファイル
fp_cmd_checksum_a	Checksumコマンド送信処理	protocol_a.c
fp_cmd_signature_a	Silicon Signatureコマンド送信処理	protocol_a.c
fp_initial_communication	通信開始の初期処理(モード引き込み)	protocol_a.c
fp_get_signature	Silicon Signatureコマンドを実行し、各パラメーターを取得する	protocol_a.c
terminal_command_init	各パラメーターの初期化	terminal_com.c
terminal_command_init_dev	デバイス依存の各パラメーターを初期化	terminal_com.c
terminal_command_offline	Flash書き換え処理を実行	terminal_com.c

6.3　関数仕様

サンプル・プログラムで使用する主な関数の仕様を示します。

read_arguments	
概　要	オプション引数の解析
書　式	static uint8_t read_arguments(st_command_data_t * cmd, int argc, char * argv[])
引　数	st_command_data_t * cmd: オプション設定情報 int argc: オプション設定数 char * argv[]: オプション引数
戻り値	0: 正常 1: 異常(オプションが正しく記載されていない)
説　明	引数の com_data を読み取る処理を行います。

system_init	
概　要	システム初期設定処理
書　式	void system_init(void)
引　数	なし
戻り値	なし
説　明	システムの初期設定を行います。

system_term	
概　要	システム終了処理
書　式	void system_term (void)
引　数	なし
戻り値	なし
説　明	システムの終了処理を行います。

config_gpio_create	
概　要	GPIO レジスタのメモリマップド I/O 取得
書　式	void config_gpio_create (int32_t mem_fd)
引　数	int32_t mem_fd: MMIO ファイルディスクリプタ
戻り値	なし
説　明	GPIO レジスタのメモリマップド I/O を取得し、GPIO ポートの設定を行います。

config_gpio_destroy	
概　要	GPIO レジスタのメモリマップド I/O 破棄
書　式	void config_gpio_destroy (void)
引　数	なし
戻り値	なし
説　明	GPIO レジスタのメモリマップド I/O を破棄する処理を行います。 また GPIO ポートをプログラム実行前の状態に戻します。

config_gpio_p23_output_start	
概　要	GPIO23 を Output 設定にする
書　式	void config_gpio_p23_output_start (void)
引　数	なし
戻り値	なし
説　明	Raspberry Pi の GPIO23 を Output 設定にする処理を行います。

config_gpio_p23_output_stop	
概　要	GPIO23 を初期設定に戻す
書　式	void config_gpio_p23_output_stop (void)
引　数	なし
戻り値	なし
説　明	Raspberry Pi の GPIO23 を初期設定に戻す処理を行います。

config_gpio_p0_txd2_start	
概　要	GPIO0 を TXD2 設定にする
書　式	void config_gpio_p0_txd2_start(void)
引　数	なし
戻り値	なし
説　明	Raspberry Pi の GPIO0 を TxD2 設定にする処理を行います。

config_gpio_p0_txd2_stop	
概　要	GPIO0 を Output 設定にする
書　式	void config_gpio_p0_txd2_stop(void)
引　数	なし
戻り値	なし
説　明	Raspberry Pi の GPIO0 を Output 設定にする処理を行います。

config_gpio_control_reset	
概　要	GPIO23(RESET)の HI/LO 制御
書　式	oid config_gpio_control_reset(uint8_t enabled)
引　数	uint8_t enabled: リセット情報(0: リセット解除 1: リセット)
戻り値	なし
説　明	Raspberry Pi の GPIO23(RESET)の HI/LO 制御を行います。 config_gpio_control_reset(0)で RESET 解除状態となり、RESET信号は HI となります。 config_gpio_control_reset(1)で RESET 状態となり、RESET信号は LO となります。

config_gpio_control_tool0	
概　要	GPIO0(TOOL0)の HI/LO 制御
書　式	void config_gpio_control_tool0(uint8_t enabled)
引　数	uint8_t enabled: GPIO0 の HI/LO 情報(0: LO 1: HI)
戻り値	なし
説　明	Raspberry Pi の GPIO0(TOOL0)の HI/LO 制御を行います。

config_systemtimer_create	
概　要	SystemTimer レジスタのメモリマップド I/O を取得
書　式	void config_systemtimer_create(int32_t mem_fd)
戻り値	int32_t mem_fd: MMIO ファイルディスクリプタ
引　数	なし
説　明	SystemTimer レジスタのメモリマップド I/O を取得する処理を行います。

config_systemtimer_destroy	
概　要	SystemTimer レジスタのメモリマップド I/O を破棄
書　式	void config_systemtimer_destroy(void)
引　数	なし
戻り値	なし
説　明	SystemTimer レジスタのメモリマップド I/O を破棄する処理を行います。

config_systemtimer_get_count	
概　要	SystemTimer のカウンタ値を取得
書　式	uint64_t config_systemtimer_get_count(void)
引　数	なし
戻り値	SystemTimer のカウント値
説　明	SystemTimer のカウンタ値を取得する処理を行います。

config_systemtimer_wait_ms	
概　要	ms 単位でウェイト
書　式	void config_systemtimer_wait_ms(const uint16_t wait_count)
引　数	const uint16_t wait_count: ウェイトカウント値[ms]
戻り値	なし
説　明	ms 単位でウェイトする処理を行います。

config_systemtimer_wait_us	
概　要	us 単位でウェイト
書　式	void config_systemtimer_wait_us(const uint16_t wait_count)
引　数	const uint16_t wait_count: ウェイトカウント値[us]
戻り値	なし
説　明	us 単位でウェイトする処理を行います。

config_uart_create	
概　要	UART レジスタのメモリマップド I/O を取得
書　式	void config_uart_create(int32_t mem_fd)
引　数	int32_t mem_fd: MMIO ファイルディスクリプタ
戻り値	なし
説　明	UART レジスタのメモリマップド I/O を取得する処理を行います。

config_uart_destroy	
概　要	UART レジスタのメモリマップド I/O を破棄
書　式	void config_uart_destroy(void)
引　数	なし
戻り値	なし
説　明	UART レジスタのメモリマップド I/O を破棄する処理を行います。

config_uart2_start	
概　要	UART2 の初期設定
書　式	void config_uart2_start(void)
引　数	なし
戻り値	なし
説　明	UART2 の初期設定を行います。(Raspberry Pi4 の場合、本関数で設定したストップビットが送受信共通になってしまうため、本サンプルでは UART2 を送信としています。)

config_uart2_stop	
概　要	UART2 の設定破棄
書　式	void config_uart2_stop(void)
引　数	なし
戻り値	なし
説　明	UART2 の設定を破棄する処理を行います。

config_uart3_start	
概　要	UART3 の初期設定
書　式	void config_uart3_start(void)
引　数	なし
戻り値	なし
説　明	UART3 の初期設定を行います。(Raspberry Pi4 の場合、本関数で設定したストップビットが送受信共通になってしまうため、本サンプルでは UART3 受信としています。)

config_uart3_stop	
概　要	UART3 の設定破棄
書　式	void config_uart3_stop(void)
引　数	なし
戻り値	なし
説　明	UAR3 の設定を破棄する処理を行います。

config_uart2_send	
概　要	TXD2 からデータ送信
書　式	e_md_status_t config_uart2_send(uint8_t * const tx_buf, uint16_t tx_num)
引　数	uint8_t * const tx_buf: 転送データ uint16_t tx_num: 送信データサイズ
戻り値	MD_OK: 正常 MD_ARGERROR: 引数エラー MD_TXERROR: 送信エラー
説　明	TXD2 からデータを送信します。データが長すぎる場合は、可能な限り書き込みを行い、残りのデータを改めて書き込みをします。これを全データが送れるまで繰り返します。 送信エラーがあった場合、write()関数は負の値を返すため、これを利用してエラー検知しています。

config_uart2_send_with_wait	
概　要	TXD2 からデータ送信(byte 間ウェイトあり)
書　式	e_md_status_t config_uart2_send_with_wait(uint8_t * const tx_buf, uint16_t tx_num)
引　数	uint8_t * const tx_buf: 送信データ uint16_t tx_num: 送信データサイズ
戻り値	MD_OK: 正常 MD_ARGERROR: 引数エラー MD_TXERROR: 送信エラー
説　明	TXD2 からデータを送信します。1byte 送信ごとにウェイトを入れ、全データが送れるまで繰り返します。

config_uart3_receive	
概　要	RXD3 からデータ受信
書　式	e_md_status_t config_uart3_receive(uint8_t * const rx_buf, uint16_t rx_num, uint16_t timeout_ms, uint8_t is_echobacked, uint16_t * p_top_pos)
引　数	uint8_t * const rx_buf: 受信データ uint16_t rx_num: 受信データサイズ uint16_t timeout_ms: タイムアウト時間 uint8_t is_echobacked: 1 線 UART のエコーバックの除去 uint16_t * p_top_pos: 受信データの先頭データ
戻り値	MD_OK: 正常 MD_ARGERROR: 引数エラー MD_RXERROR: 受信エラー MD_RXTIMEOUT: 受信タイムアウトエラー
説　明	RXD3 からデータ受信を受信します。読んだデータが不足している場合(全データを受信しきれていない場合など)は、可能な限り読み取り、改めて読み取ります。これを全データが読めるまで繰り返します。 最後まで受信できなかった場合は、どこかの段階で select()に失敗し、タイムアウトが発生します。

config_uart23_set_baudrate	
概　要	UART2,UART3 のボーレート設定
書　式	void config_uart23_set_baudrate(e_uart_baudrate_t baudrate)
引　数	e_uart_baudrate_t baudrate: ボーレート
戻り値	なし
説　明	UART2,UART3 のボーレート設定を行います。

config_uart2_set_send_wait_time	
概　要	config_uart2_send_with_wait のウェイト時間設定
書　式	void config_uart2_set_send_wait_time(uint8_t freq)
引　数	uint8_t freq : ターゲット周波数
戻り値	なし
説　明	送信後の待ち時間の設定を行います。

fp_cmd_reset_a	
概　要	Reset コマンド送信処理
書　式	uint8_t fp_cmd_reset_a(void)
引　数	なし
戻り値	0: 正常終了 0 以外: 異常終了 (4.2 エラーコード仕様を参照)
説　明	RL78 プロトコル A の Reset コマンドを実行します。

fp_cmd_verify_a	
概　要	Verify コマンド送信処理
書　式	uint8_t fp_cmd_verify_a(const uint32_t start, const uint32_t end, const uint8_t * data)
引　数	const uint32_t start: ベリファイ開始アドレス const uint32_t end: ベリファイ終了アドレス const uint8_t * data: ベリファイ比較用データ
戻り値	0: 正常終了 0 以外: 異常終了 (4.2 エラーコード仕様を参照)
説　明	RL78 プロトコル A の Verify コマンドを実行します。

fp_cmd_erase_a	
概　要	Block Erase コマンド送信処理
書　式	uint8_t fp_cmd_erase_a(const uint32_t addr)
引　数	const uint32_t addr: 消去ブロックアドレス
戻り値	0: 正常終了 0 以外: 異常終了 (4.2 エラーコード仕様を参照)
説　明	RL78 プロトコル A の Block Erase コマンドを実行します。

fp_cmd_program_a	
概　要	Programming コマンド送信処理
書　式	uint8_t fp_cmd_program_a(const uint32_t start, const uint32_t end, const uint8_t * data)
引　数	const uint32_t start: 書き込み開始アドレス const uint32_t end: 書き込み終了アドレス const uint8_t * data: 書き込みデータ
戻り値	0: 正常終了 0 以外: 異常終了 (4.2 エラーコード仕様を参照)
説　明	RL78 プロトコル A の Programming コマンドを実行します。

fp_cmd_baudrate_a	
概　要	Baud Rate Set コマンド送信処理
書　式	uint8_t fp_cmd_baudrate_a(const e_uart_speed_t baudrate, const uint16_t vdd, uint8_t * frq, uint8_t * fpm)
引　数	const UART_SPEED baudrate: 通信ボーレート UART_SPEED_DEFAULT: 115200 bps UART_SPEED_250000: 250000 bps UART_SPEED_500000: 500000 bps UART_SPEED_1000000: 1000000 bps const uint16_t vdd: V_{DD} 印可電圧 [100 mV] uint8_t * frq: CPU 動作周波数 [MHz] (ターゲット MCU から取得) uint8_t * fpm: フラッシュ書き換えモード(ターゲット MCU から取得)
戻り値	0: 正常終了 0 以外: 異常終了 (4.2 エラーコード仕様の項を参照)
説　明	RL78 プロトコル A の Baud Rate Set コマンドを実行します。

fp_cmd_checksum_a	
概　要	Checksum コマンド送信処理
書　式	uint8_t fp_cmd_checksum_a(const uint32_t start, const uint32_t end, uint16_t * checksum)
引　数	const uint32_t start: チェックサム開始アドレス const uint32_t end: チェックサム終了アドレス const uint16_t * checksum: 取得したチェックサムデータ
戻り値	0: 正常終了 0 以外: 異常終了 (4.2 エラーコード仕様を参照)
説　明	RL78 プロトコル A の Checksum コマンドを実行します。

fp_cmd_signature_a	
概　要	Silicon Signature コマンド送信処理
書　式	uint8_t fp_cmd_signature_a(uint8_t * dvc, uint8_t * dev, uint32_t * cfe, uint32_t * dfe, uint8_t * fwv)
引　数	uint8_t * dvc: デバイス機能コード(ターゲット MCU から取得) uint8_t * dev: デバイス名(ターゲット MCU から取得) uint32_t * cfe: コード・フラッシュ領域最終アドレス(ターゲット MCU から取得) uint32_t * dfe: データ・フラッシュ領域最終アドレス(ターゲット MCU から取得) uint8_t * fwv: ブートファームウェアバージョン(ターゲット MCU から取得)
戻り値	0: 正常終了 0 以外: 異常終了 (4.2 エラーコード仕様を参照)
説　明	RL78 プロトコル A の Silicon Signature コマンドを実行します。

fp_initial_communication	
概　要	通信開始の初期処理(モード引き込み)
書　式	uint8_t fp_initial_communication(st_command_data_t * command)
引　数	st_command_data_t * command: コマンド設定情報
戻り値	0: 正常終了 0 以外: 異常終了 (4.2 エラーコード仕様を参照)
説　明	RL78 プロトコル A に従い、ターゲット MCU との初期通信を行います。 リセット解除、モード情報送信、Baud Rate Set コマンド、Reset コマンドを実行します。

fp_get_signature	
概　要	Silicon Signature コマンドを実行し、各パラメーターを取得する
書　式	uint8_t fp_get_signature(st_command_data_t * command)
引　数	st_command_data_t * command: コマンド設定情報
戻り値	0: 正常終了 0 以外: 異常終了 (4.2 エラーコード仕様を参照)
説　明	ターゲット MCU のシグネチャ情報を取得し、引数の com_data に設定します。

terminal_command_init	
概　要	各パラメーターの初期化
書　式	void terminal_command_init(st_command_data_t * com_data)
引　数	st_command_data_t * com_data: コマンド設定情報
戻り値	なし
説　明	引数の com_data を初期化します。

terminal_command_init_dev	
概　要	デバイス依存の各パラメーターを初期化
書　式	void terminal_command_init_dev(st_command_data_t * com_data)
引　数	st_command_data_t * com_data: コマンド設定情報
戻り値	なし
説　明	引数の com_data のシグネチャ情報を初期化します。

terminal_command_offline	
概　要	Flash 書き換え処理を実行
書　式	void terminal_command_init(st_command_data_t * com_data)
引　数	st_command_data_t * com_data: コマンド設定情報
戻り値	なし
説　明	ファイルからの S-Record データ読み出し、Flash 書き換え処理を実行します。

7. 参考ドキュメント

RL78 マイクロコントローラ(RL78 プロトコル A) シリアルプログラミング編 (R01AN0815)

(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	2024/2/29	—	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。