Renesas Synergy™
電子錠ドア（Electronic Door Lock）

R30AN0275JJ0110
Rev.1.10
2017.4.27

要旨

本書では、Renesas Synergy™ を使ったアプリケーション開発事例として、電子錠ドアの試作版アプリケーション、及び Bluetooth Low Energy (BLE) 通信を使ったドアのモニタシステムについて説明します。本書で主に説明する内容は、アプリケーションの仕様、準備手順、ハードウェアとソフトウェアの構成、及び電子錠ドアの LCD 画面の作成・制御方法（Appendix）であり、各スレッド内の詳細な処理については説明しません。本書は、以下のソフトウェア環境に基づいて説明しています。

表 1-1　ソフトウェア環境

Renesas Synergy™ Software Package (SSP)	v1.2.0
Renesas Synergy™ Standalone Configurator (SSC)	v5.3.1.002
e²studio ISDE	v5.3.1.002
IAR Embedded Workbench® for Renesas Synergy™ (IAR EW for Synergy)	v7.71.1

動作確認デバイス

DK-S7G2

SK-S7G2

目次

1. はじめに

1.1 概要

Renesas Synergy™ プラットフォームは、組込みシステム開発の複雑化、コスト増加、開発期間の長期化といった問題を解決するために提案されたプラットフォームです。その中で、Renesas Synergy™ Software Package（以下、SSP）では、RTOS、HAL ドライバ、ソフトウェア・フレームワークを動作保証した形で提供しているため、ユーザは、これらを利用することで、アプリケーションの開発に集中することができます。本書では、実際のシステム開発事例として、電子錠ドアのアプリケーション開発、及び電子錠ドアと BLE で通信し電子錠ドアの状態を表示するモニタの開発事例について説明します。電子錠ドアは、Renesas Synergy™ の開発ボードに各種センサやデバイスを接続し、ドアの開閉制御部分を省略した試作版を開発するという位置付けです。

1.2 参考文献

[1] Renesas, “Renesas Synergy™ Platform Renesas Bluetooth MCU (RL78/G1D) (R12AN0056EU0100 Rev. 01.00)”.

[2] Renesas, “RL78/G1D モジュール ファームウェア ユーザーズマニュアル (R01UW0160JJ0100 Rev. 1.00)”

[3] Max Botix, “LV-MaxSonar®-EZ1™ Series High Performance Sonar Range Finder”.

[4] Analog Devices, “Digital Accelerometer ADXL345 Data Sheet (Rev.E)”.

[5] Express Logic, “GUIX User Guide (Version 5)”.

[6] Express Logic, “GUIX Studio User Guide (V5 for Windows)”.

2. 仕様

図 2-1 にデバイス構成を示します。電子錠ドア（Electronic Door Lock）は DK-S7G2 上で動作し、PmodMaxSONAR™（距離計測用の超音波センサ）、PmodACL™（衝撃検知用の加速度センサ）、LCD（時刻やメッセージなどの表示用、及びユーザ入力用）、そして RL78/G1D（BLE 通信用）で構成されます。また、モニタ（Monitor）は SK-S7G2 上で動作し、LED（通信状態表示用）、LCD（ドアの状態表示用）、及び RL78/G1D で構成されます。

本アプリケーションにおいて、S7G2 は rBLE API（RL78/G1D を制御するための API）を用いて RL78/G1D を制御します。また、データ通信には Cycling Speed and Cadence Profile を用います。

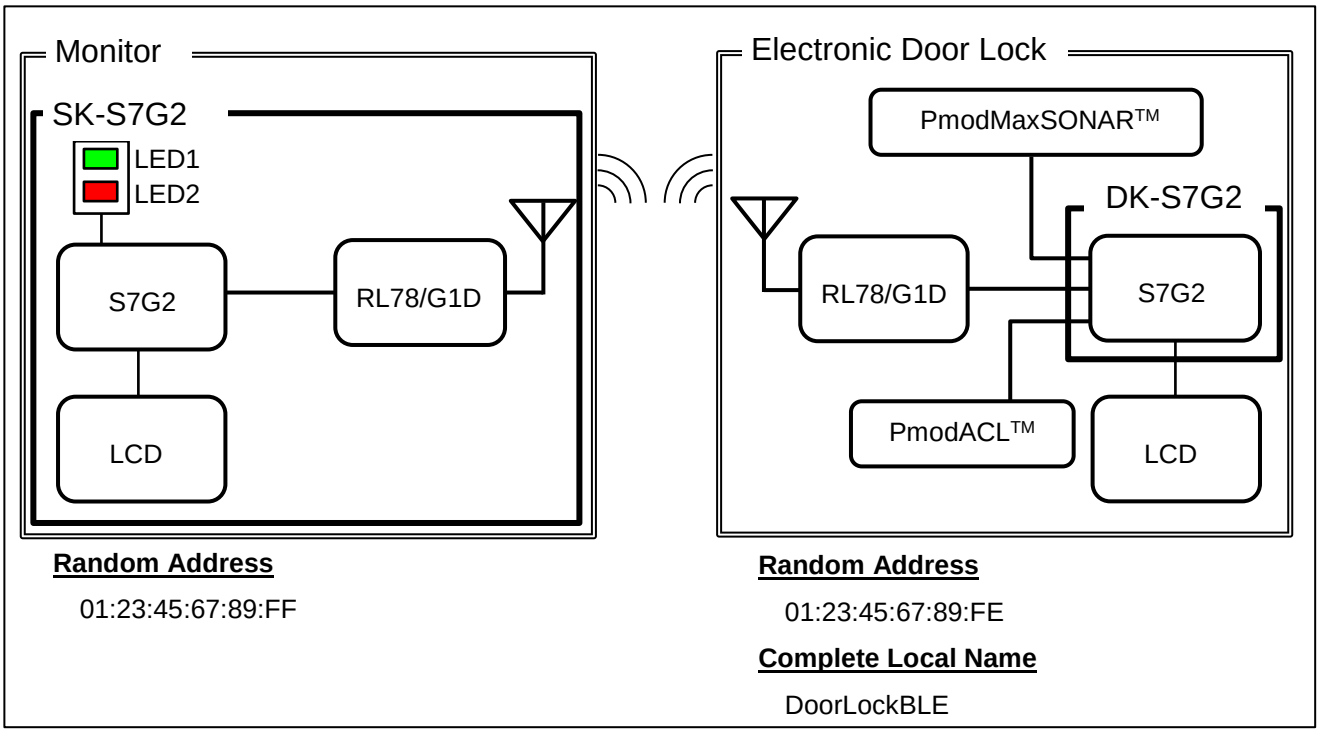

図 2-1 デバイス構成

サンプルプログラムの概要は以下の通りです。

- 電子錠ドア（DK-S7G2）
  - 電源投入後、LCD に図 2-2 の時刻設定画面を表示し、時刻の入力を待つ（HH:MM:SS 形式）。また、自身のランダムアドレスを「01:23:45:67:89:FE」に変更した後、Advertising Packet 中に Complete Local Name として「DoorLockBLE」を設定し、Broadcast を開始する。
  - 時刻を入力し Enter を押すと（もしくは時刻を入力せずに Enter を押して時刻設定がスキップされると）、図 2-3 のパスコード設定画面を表示し、パスコードの入力を待つ（4 桁の数字）。
  - パスコードを入力し Enter を押すと、図 2-4 の画面を表示し、その後 LCD 画面を消灯する。
  - 超音波センサと人(物)との距離が 30cm 以内になると、LCD 画面を点灯して図 2-5 のパスコード入力画面を表示し、パスコードの入力を待つ。また、パスコード画面を表示する際にテンキー内のキー配置をランダムに変える。更に、モニタとの BLE 通信が確立している場合は（モニタの LED1 が点灯している場合）、モニタに向けて 1 秒毎に「時刻、超音波センサから取得した距離、状態」を通知する（Notification）。

    ※この通知は、20 秒間継続。

- 正しいパスコードを入力し、Enter を押した場合、LCD にドアの解錠に成功したことを示すメッセージを表示し（図 2-6）、その後 LCD を消灯する。また、間違ったパスコードを入力した場合、LCD にドアの解錠に失敗したことを示すメッセージを表示し（図 2-7）、その後 LCD を消灯する。
- 電子錠ドアに衝撃を与えた場合は、LCD に警告メッセージを表示し（図 2-8）、その後 LCD を消灯する。この時、パスコード入力時と同様にモニタに向けて 1 秒毎に「時刻、超音波センサから取得した距離、状態」を通知する（Notification）。

  ※この通知は、20 秒間継続。

- モニタ（SK-S7G2）
  - 電源投入後、自身の Random Address を「01:23:45:67:89:FF」に変更する。
  - Random Address を変更後、Device 検索を行ない、Complete Local Name が「DoorLockBLE」のデバイス（電子錠ドア）を発見した場合、接続を開始する。
  - 接続完了後、プロファイル（CSCP）の有効化と Notification の許可を行い、LED1 を点灯する。
  - 電子錠ドアを発見できなかった場合、もしくは接続が失敗した場合は、LED2 を点灯する。
  - 電子錠ドアから Notification を受信する度に、その内容を LCD に表示する。

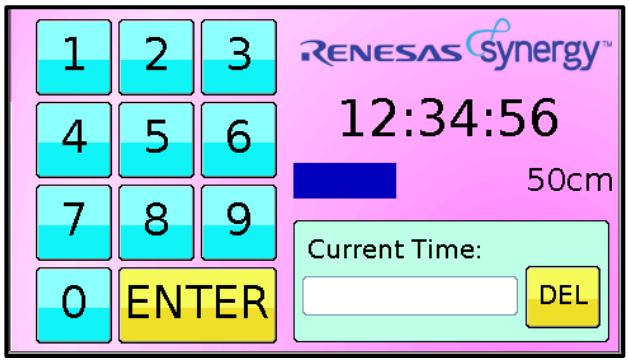

図 2-2　時刻設定画面

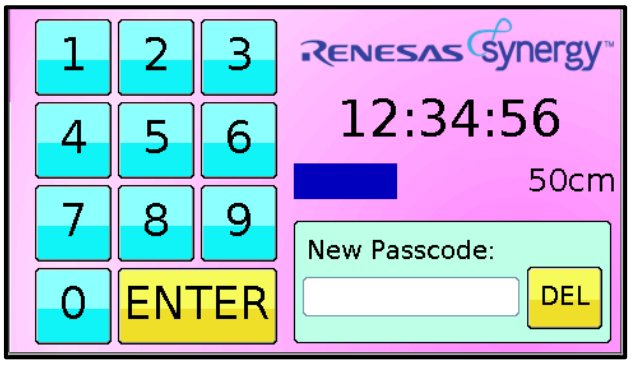

図 2-3　パスコード設定画面

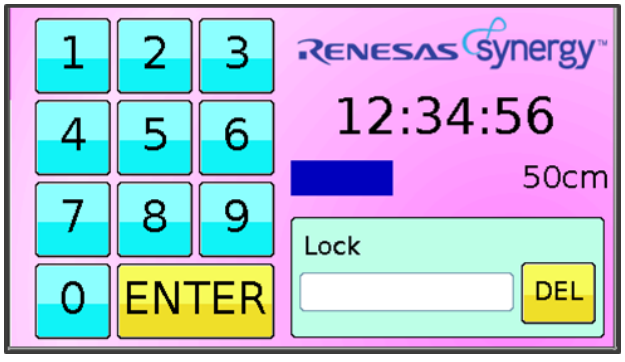

図 2-4　施錠画面

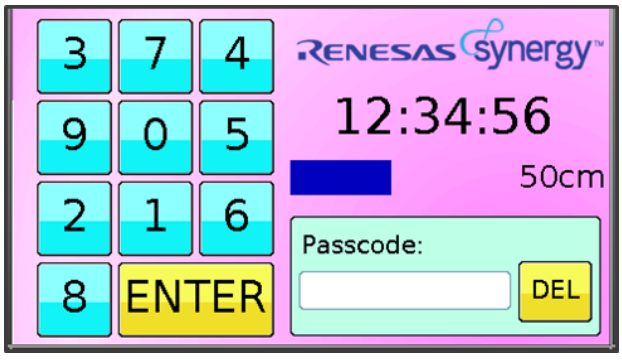

図 2-5　パスコード入力画面

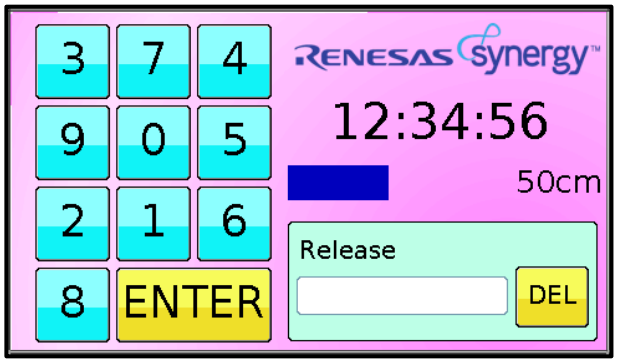

図 2-6　解錠成功画面

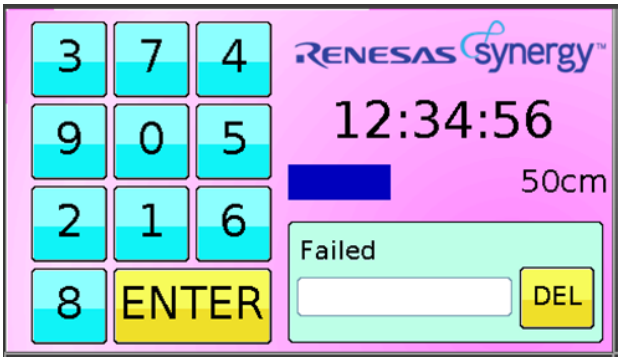

図 2-7　解錠失敗画面

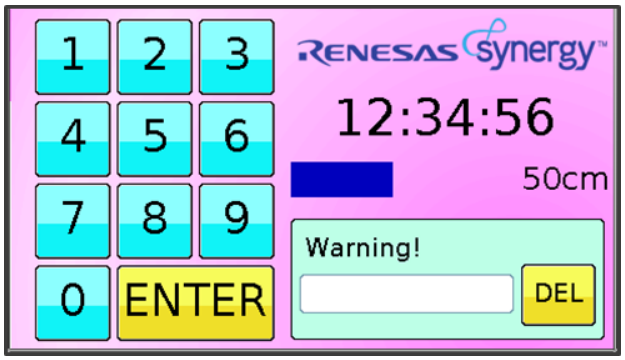

図 2-8　警告画面

3. 準備手順

3.1 RL78/G1D 用ファームウェアの準備

付属の RL78_G1D_IM(CSCP,CPP).hex を SK-S7G2 上の RL78/G1D 及び DK-S7G2 に接続する RL78/G1D 評価モジュールへ書き込んでください。

3.1.1 SK-S7G2 上の RL78/G1D へのファームウェア書き込み手順

参考文献 [1] の 13.1 Appendix 1 を参考にして、SK-S7G2 上の RL78/G1D へファームウェアを書き込んでください。

3.1.2 DK-S7G2 に接続する RL78/G1D 評価モジュールへのファームウェア書き込み手順

RL78/G1D 評価モジュールに E1 エミュレータを接続し、Renesas Flash Programmer *1 を使用してファームウェアを書き込んでください。

*1 本アプリケーションノートでは、Renesas Flash Programmer V3.02.01 の無償版を利用しています。詳細は以下の URL を参照してください。

https://www.renesas.com/ja-jp/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html

Renesas Flash Programmer の操作方法は以下の通りです。

(1) RL78/G1D に E1 エミュレータを接続した状態で、Renesas Flash Programmer を起動し、[ファイル] → [新しいプロジェクトを作成(C)…]を選択します。そして、マイクロコントローラに「RL78」を指定し、プロジェクト名に任意のプロジェクト名を指定します。また、E1 エミュレータからの電源供給は環境に合わせて設定してください（図 3-1）。

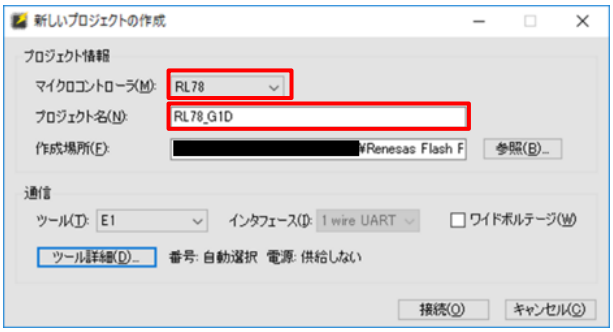

図 3-1 プロジェクトの作成

(2) 操作タブ内のプログラムファイルに上記ファームウェアを設定します（図 3-2）。

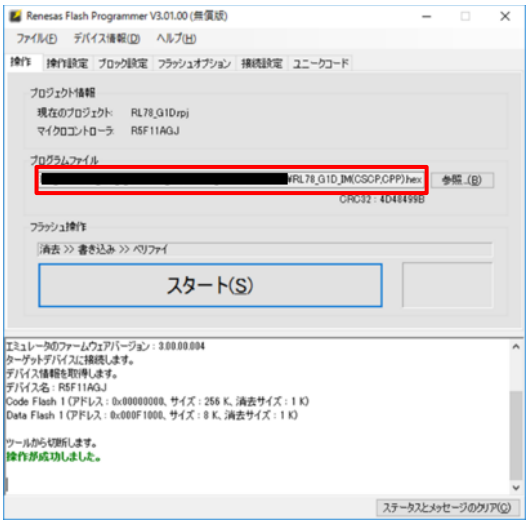

図 3-2 書き込み対象のファームウェアの設定

(3) 操作設定タブにて、消去オプションがブロック選択消去であることを確認します（図 3-3）。また、ブロック設定タブにて、図 3-4 の通り[Code Flash1]の[Block254]と[Block255]のチェックを外します。詳細は参考文献[2]を参照してください。

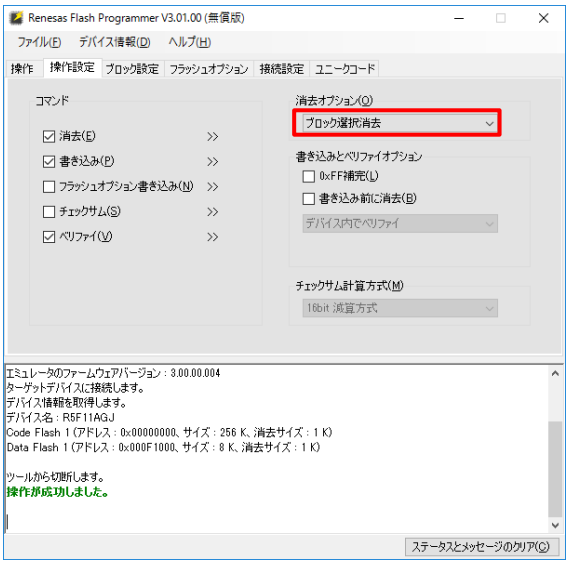

図 3-3 操作設定タブ

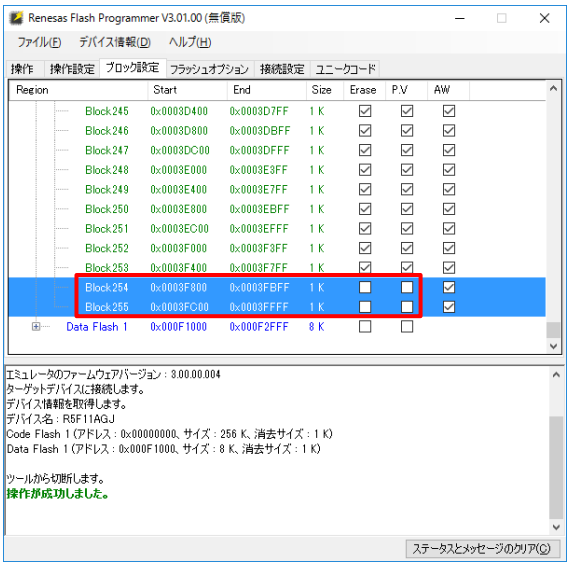

図 3-4 ブロック設定タブでの設定

(4) 操作タブにて、[スタート]を押し、プログラムを書き込みます。書き込みが正常に終了すると、[正常終了]と表示されます（図 3-5）。

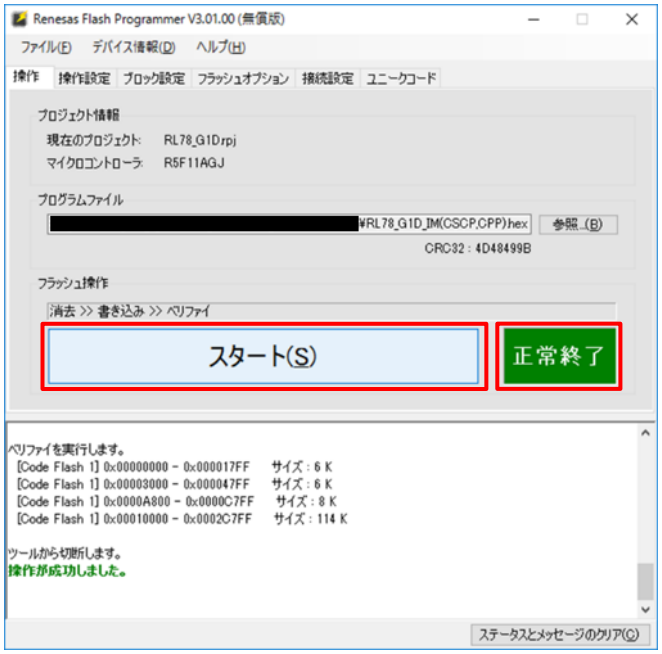

図 3-5 ファームウェアの書き込み

3.2 各デバイスとの接続例

図 3-6 に接続例を示します。PMODA の上段に PmodMAXSONAR™ を、PMODB に RL78/G1D 評価モジュールを、PMODC に PmodACL™ を、そして DK-S7G2 付属の LCD を J102 に接続します。RL78/G1D 評価モジュールを PMODB に接続する際は、別途アダプタ基板等を用意する必要があります。また、BAT1 に CR1220 ボタン電池を装着することで、AC 電源アダプタからの電源供給が行われない場合でも RTC へ電源供給することが可能です。PmodMAXSONAR™ と PmodACL™ については、参考文献[3], [4]を参照してください。

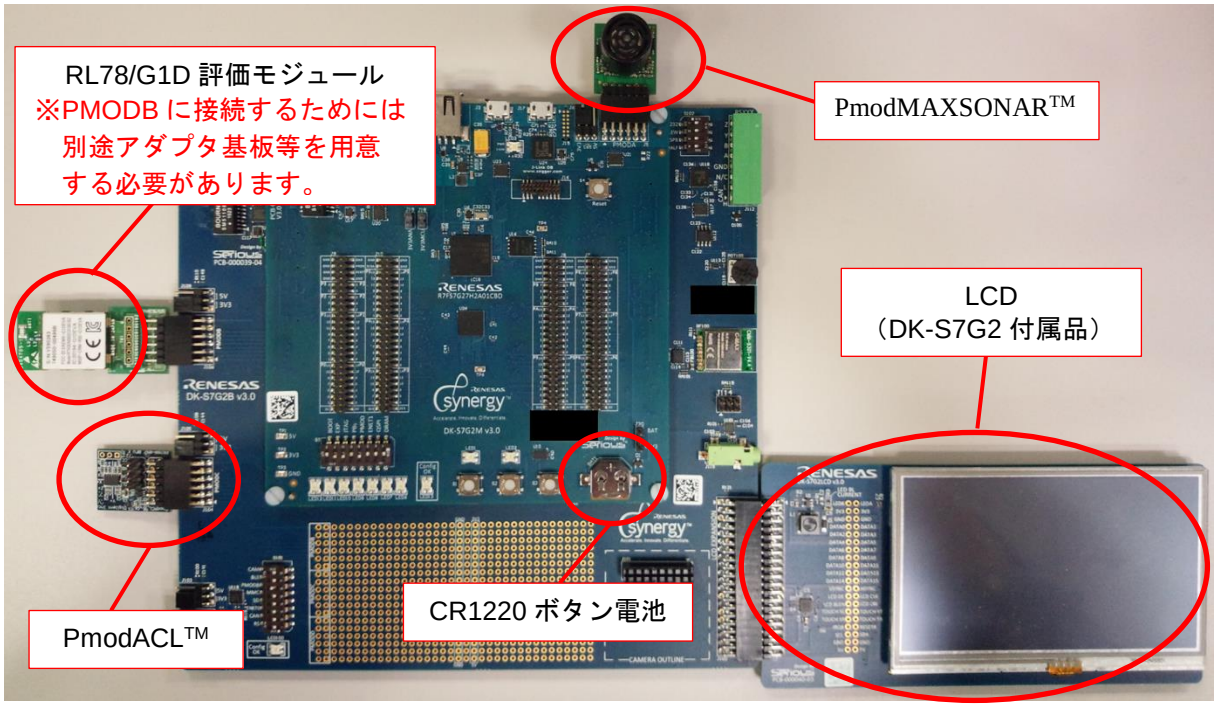

図 3-6 接続例

3.3 ソフトウェアのインストール

電子錠ドアとモニタそれぞれに対応するプロジェクトを表 3-1 に従ってインポートしてください。

表 3-1 プロジェクトファイル

電子錠ドア（DK-S7G2）	ElectronicDoorLock_Door_DK_S7G2.zip
モニタ（SK-S7G2）	ElectronicDoorLock_Monitor.zip

4. ハードウェア構成

4.1 ブロック図

図 4-1 に電子錠ドア（DK-S7G2）、図 4-2 にモニタ（SK-S7G2）のブロック図を示します。

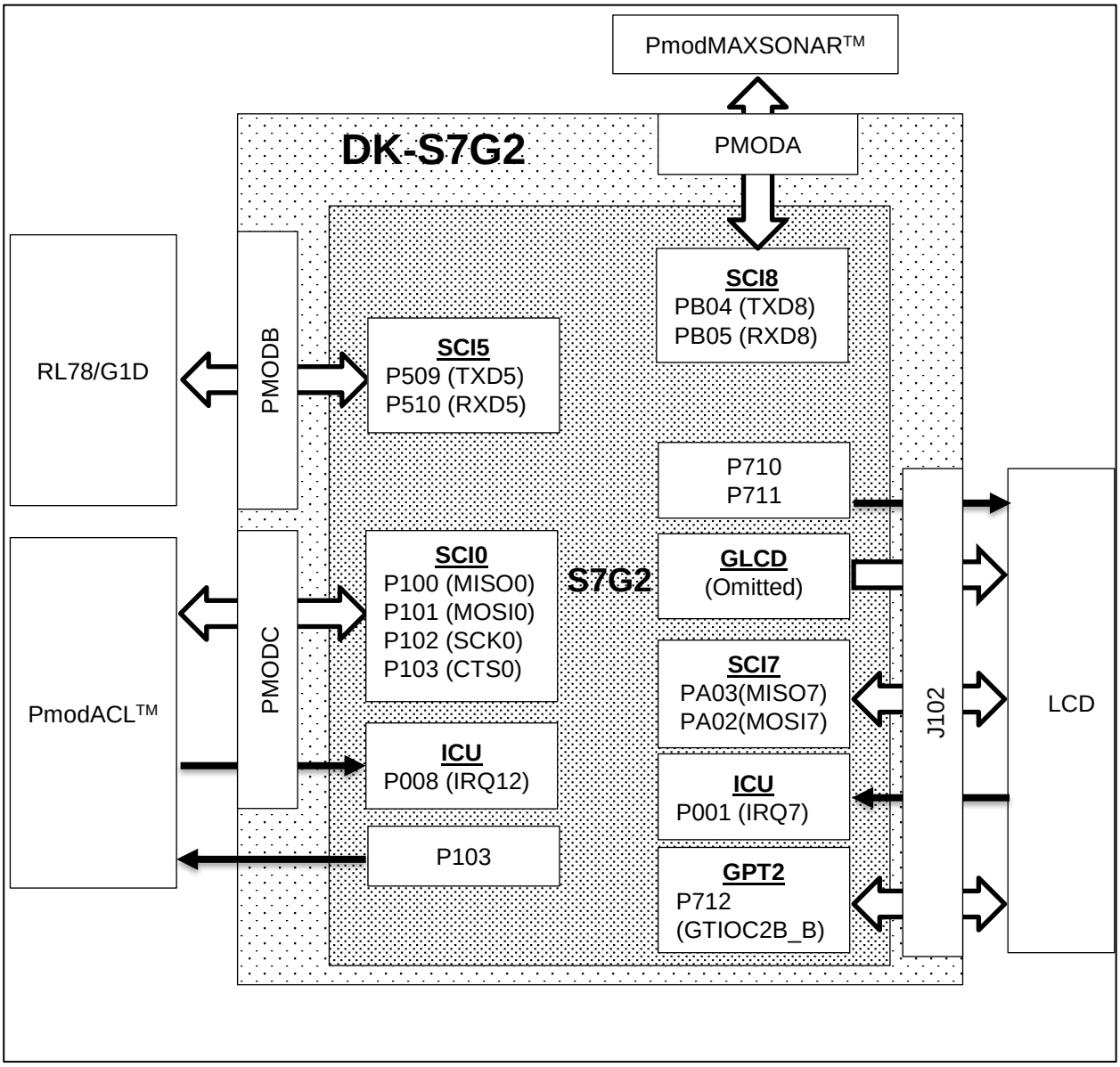

図 4-1 電子錠ドアのブロック図

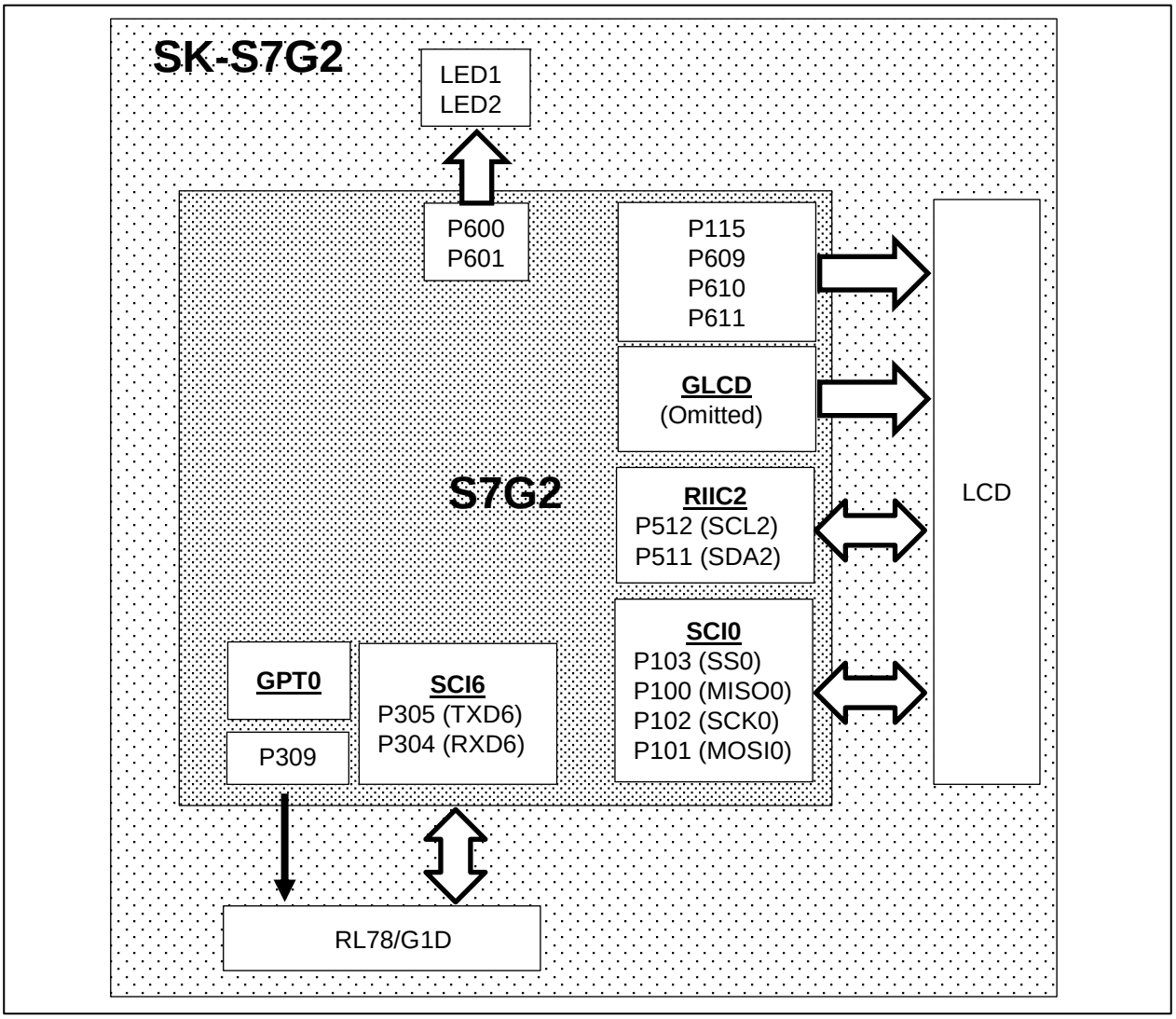

図 4-2　モニタのブロック図

4.2　DK-S7G2 の DIP スイッチ設定

本アプリケーションの動作に必要な DK-S7G2 ボードのスイッチ設定を表 4-1、表 4-2 に示します。

表 4-1　DK-S7G2M スイッチ S5 設定

番号	名称	設定*1
S5-1	DRAM	OFF
S5-2	QSPI	OFF
S5-3	ENET1	OFF
S5-4	PMOD	ON
S5-5	PBs	OFF
S5-6	JTAG	ON
S5-7	EXP	ON
S5-8	BOOT	OFF

*1 グレーは任意

表 4-2　DK-S7G2B スイッチ S101 設定

番号	名称	設定*1
S101-1	RS	OFF
S101-2	CAN	OFF
S101-3	ENET0	OFF
S101-4	SD	OFF
S101-5	MMC	OFF
S101-6	PMODB	ON
S101-7	BLE	OFF
S101-8	CAM	OFF

*1 グレーは任意

5. ソフトウェア構成

5.1 スレッド構成

電子錠ドア／モニタ上で動作するスレッドを表 5-1 に示します。これらのスレッドは、ユーザがプロジェクト作成時に追加するスレッドであり、使用するフレームワークなどで作成されるスレッドは含みません。各スレッドに対する設定値を表 5-2 に示します。また、図 5-1 に電子錠ドア、図 5-2 にモニタのスレッド構成を示します。

表 5-1 スレッド一覧

スレッド名		スレッドの機能
電子錠ドア		
	Main Thread	各スレッドからのメッセージ通信を行い、電子錠ドアとしての機能を構成する。
	BLE Thread	RL78/G1D を制御し、モニタとの接続確立や Notification の通知を行う。
	GLCD Thread	LCD を制御し、時刻や距離、テンキーやメッセージの表示を行う。また、ユーザからの入力があった場合は、押されたキーを Main Thread へ通知する。
	Acl Thread	加速度センサを制御し、衝撃を検知した場合は Main Thread へ衝撃があったことを通知する。
	Sonar Thread	近接センサを制御し、Main Thread へ測定した距離を通知する。
	Timer Thread	各スレッドが設定した時間を超過した場合に、対応するスレッドへタイマがオーバフローしたことを通知する。
	RTC Thread	RTC を制御し、1 秒毎に Main Thread へ時刻を通知する。
モニタ		
	BLE Thread	RL78/G1D を制御し、電子錠ドアとの接続確立や Notification の受信を行う。また、Log Thread へ各処理のログを通知する。
	GLCD Thread	LCD を制御し、BLE Thread が受信した電子錠ドアの状態と Log Thread が保持しているログ情報を LCD へ表示する。
	Log Thread	BLE Thread からのメッセージをログ情報として保持する。

表 5-2 各スレッドの設定値

Name		Symbol	Stack size (bytes)	Priority	Auto Start	Time slicing Intervals (ticks)
電子錠ドア						
	Main Thread	main_thread	1024	2	Enabled	1
	BLE Thread	ble_thread	1024	1	Enabled	1
	GLCD Thread	glcd_thread	1024	2	Enabled	1
	Acl Thread	acl_thread	1024	2	Enabled	1
	KeyPad Thread	keypad_thread	1024	2	Enabled	1
	Sonar Thread	sonar_thread	1024	2	Enabled	1
	Timer Thread	timer_thread	1024	2	Enabled	1
	RTC Thread	rtc_thread	1024	2	Enabled	1
モニタ						
	BLE Thread	ble_thread	1024	1	Enabled	1
	GLCD Thread	glcd_thread	2048	2	Enabled	1
	Log Thread	log_thread	1024	3	Enabled	1

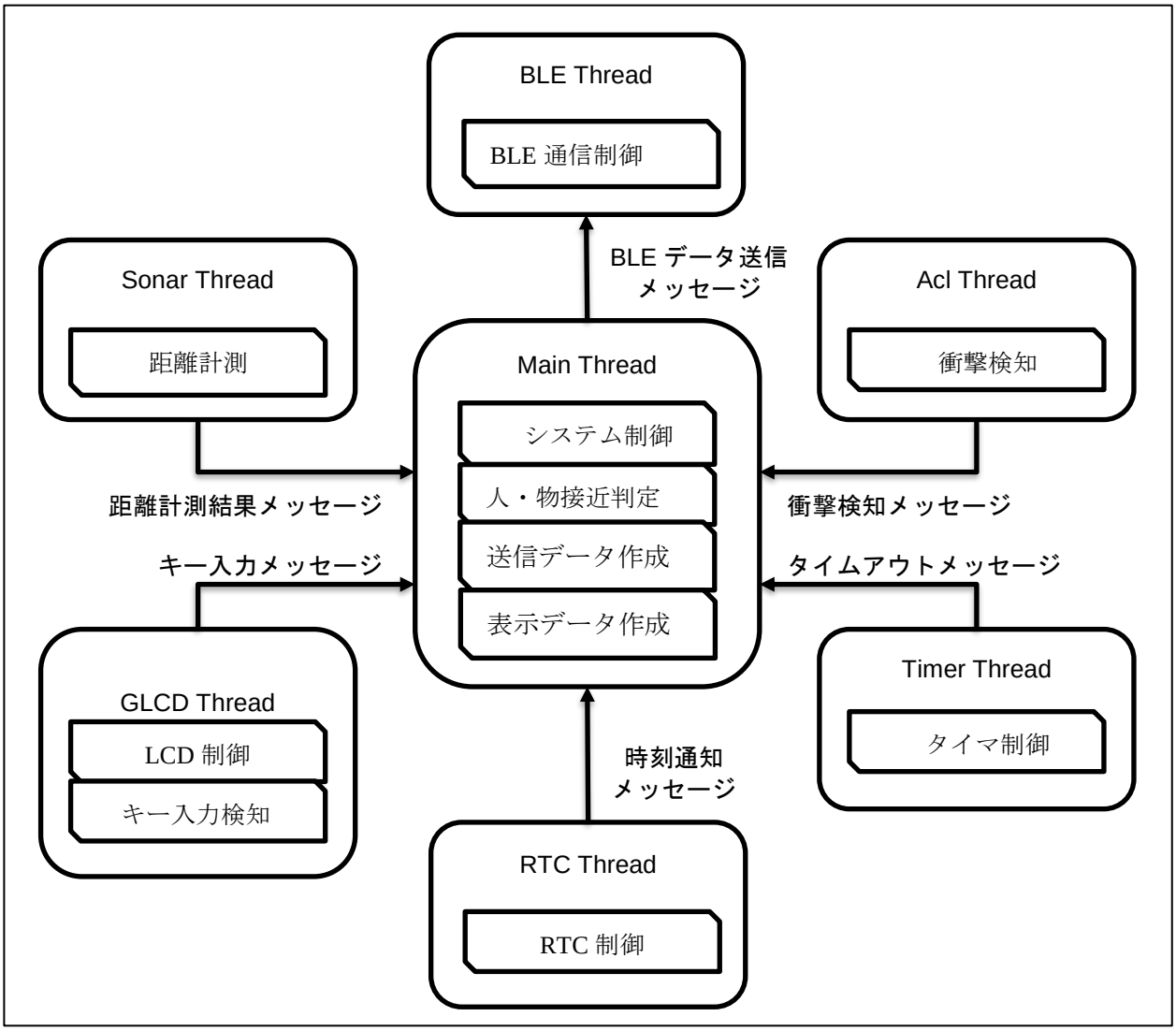

図 5-1　電子錠ドアのスレッド構成

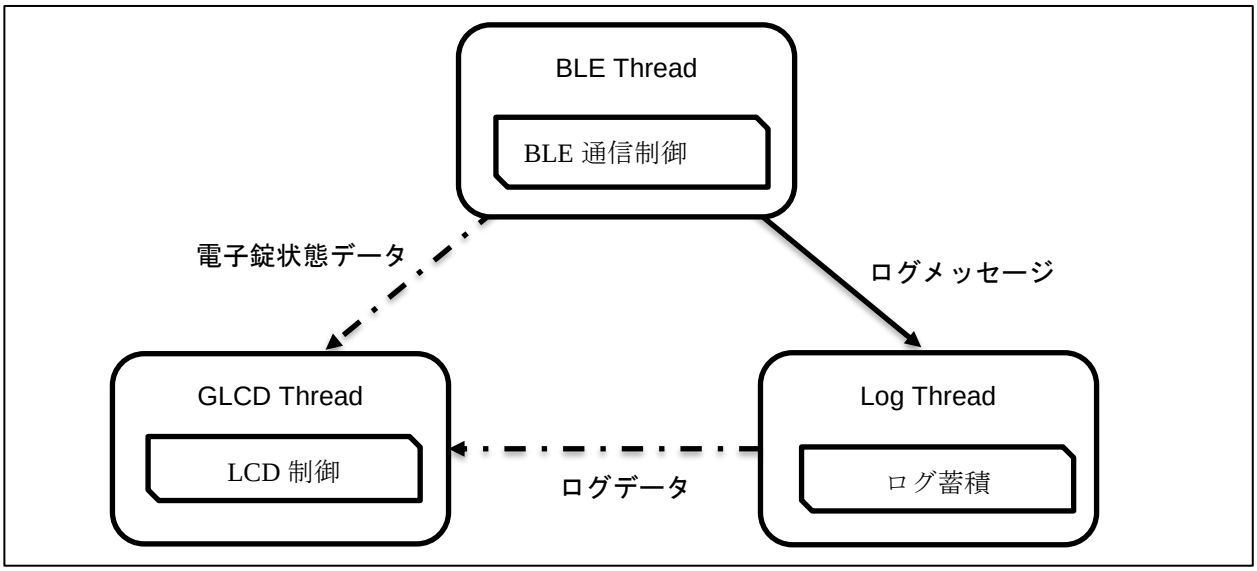

図 5-2　モニタのスレッド構成

5.2 電子錠ドア上で動作するスレッドの機能概要

5.2.1 Main スレッド

システムを初期化した後、Messaging Framework を用いて、各スレッドと通信を行い電子錠ドアの動作を制御します。送受信されるメッセージと各スレッドとの対応を図 5-3 に示します。また、Main Thread の各状態で各メッセージを受信した時の処理と次の状態への遷移について表 5-3 に示します。

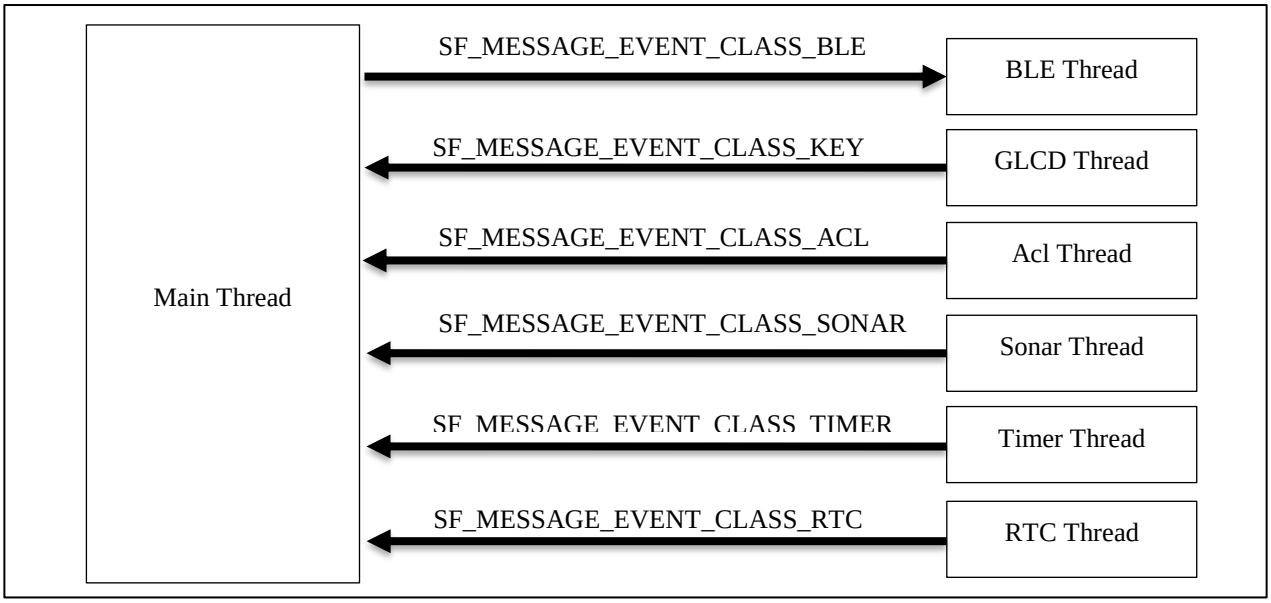

図 5-3 送受信されるメッセージと各スレッドとの対応

表 5-3 電子錠ドアの状態遷移表

State	Timeout	Event Class				
"Message" *1	(sec) *2	SONAR	KEY	TIMER	ACL	RTC *8
SET_CAL *"Current Time:"*	-	-	SET_PC	-	-	-
SET_PC *"New Passcode:"*	-	-	LOCK_ENT *3 *4		-	-
LOCK_ENT *"Lock"*	2	-	-	LOCKING	-	-
LOCKING *"(LCD 消灯)"*	-	PC_INP	-	-	WARN *4	-
PC_INP *"Passcode:"*	10	-	UNLOCK *5 PC_NG *6 - *7	LOCKING	-	-
PC_NG *"Failed"*	2	-	-	LOCKING	-	-
UNLOCK *"Release"*	5	-	-	LOCK_ENT	-	-
WARN *"Warning!"*	3	-	-	LOCKING	-	-

*1 遷移時に表示するメッセージ。
*2 遷移時に設定するタイマのタイムアウト値。
*3 設定パスコード長分の数字が入力された場合。それ以外は状態遷移なし。
*4 BLE メッセージを送信する回数を設定する。
*5 パスコードが正しい場合。
*6 パスコードが不正の場合。
*7 入力長が設定パスコード長未満の場合。
*8 状態遷移なし。BLE メッセージを送信する回数が設定されている場合は、その回数分 RTC メッセージを受信する毎に BLE スレッドに対して BLE メッセージを送信する。

5.2.2 BLE スレッド

rBLE API を用いて、RL78/G1D を制御します。起動後、Random Address の設定や Broadcast の開始を行います。モニタとの接続が確立した状態で、Main スレッドから SF_MESSAGE_EVENT_CLASS_BLE メッセージを受信した場合は、メッセージに設定された情報を基に Notification（CSC Measurement）をモニタへ送信します（図 5-4）。また、CSC Measurement に設定される各データの内容は表 5-4 の通りです。

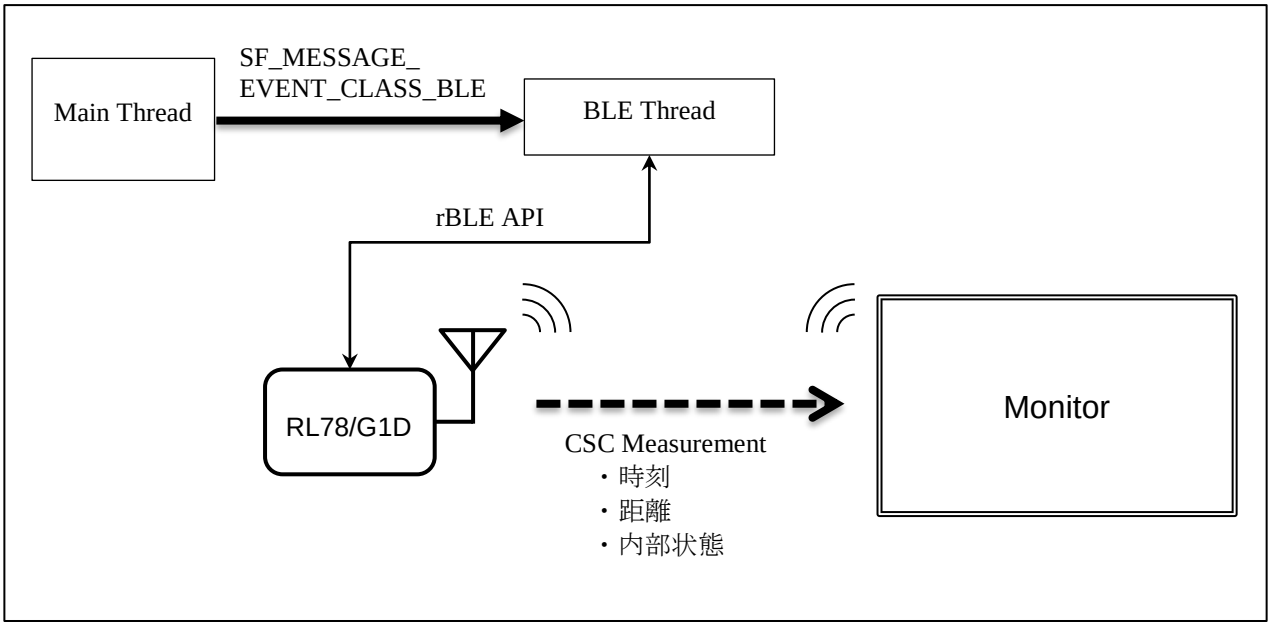

図 5-4 BLE スレッドが受信するメッセージ

表 5-4 CSC Measurement の内容

Fields	Value
Flags	0x03 (Wheel/Crank Revolution Data ともに有効化)
Cumulative Wheel Revolutions	時刻（BCD 形式：HHMMSS）
Last Wheel Event Time	距離（0x0000～0x0064：0cm～100cm）
Cumulative Crank Revolutions	状態 0x0001: SET_CAL 0x0002: SET_PC 0x0003: LOCK_ENT 0x0004: LOCKING 0x0005: PC_INP 0x0006: PC_NG 0x0007: UNLOCK 0x0008: WARN
Last Crank Event Time	0x0000 (未使用)

5.2.3 GLCD スレッド

GUIX と Touch Panel Framework を用いて、DK-S7G2 付属の LCD を制御します。キー入力を検出した場合は、その検出したキーを SF_MESSAGE_EVENT_CLASS_KEY メッセージに格納して Main スレッドに送信します。また、Main スレッドから取得した時刻や距離を基に LCD 画面を更新します（図 5-5）。

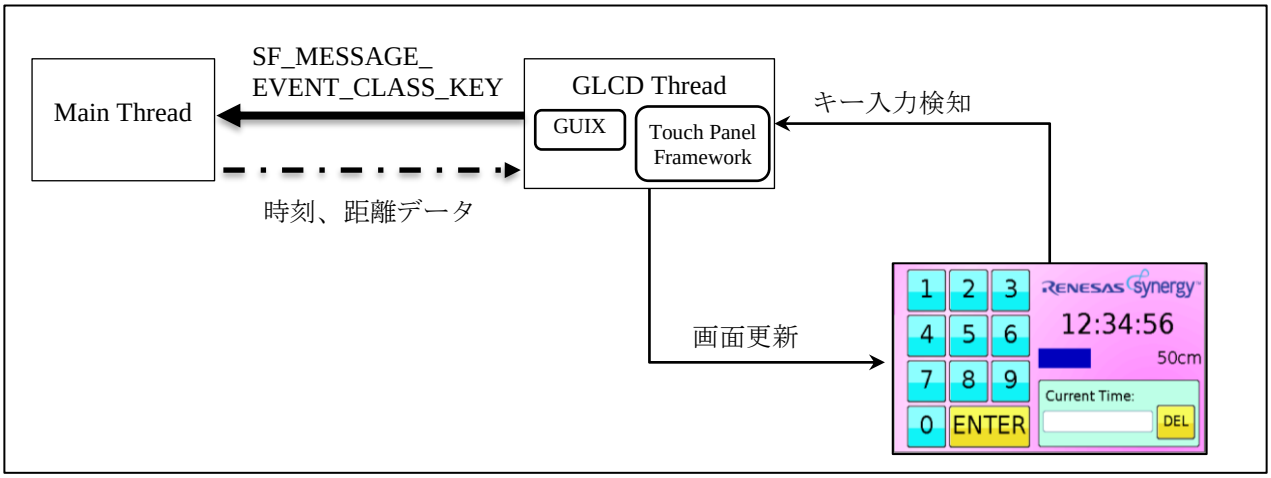

図 5-5 GLCD スレッドの機能概要

5.2.4 Acl スレッド

加速度センサ（PmodACL™）を制御して、衝撃を検知します。起動時に以下の初期化を行います。

- 加速度センサのキャリブレーション
- タップ検出条件の設定
- タップ検出時の割り込み出力設定（SINGLE_TAP を INT2 に出力）

その後、加速度センサからのタップ発生を IRQ12 の割り込みで待ち、割り込みが発生すると、SF_MESSAGE_EVENT_CLASS_ACL イベントのメッセージを出力します（図 5-6）。

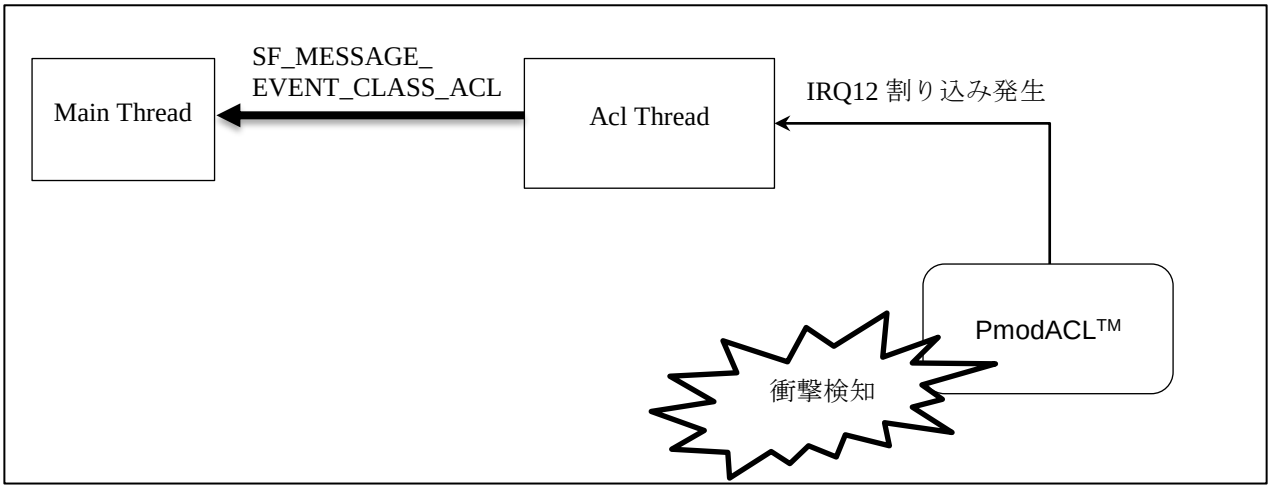

図 5-6 Acl スレッドの機能概要

5.2.5 Sonar スレッド

超音波センサ（PmodMAXSONAR™）を制御し、電子錠ドア - 人・物間の距離を測定します。超音波センサから 50msec 毎に UART で送信される測定値を受信し、その測定値を SF_MESSAGE_EVENT_CLASS_SONAR メッセージに格納して Main スレッドへ送信します（図 5-7）。

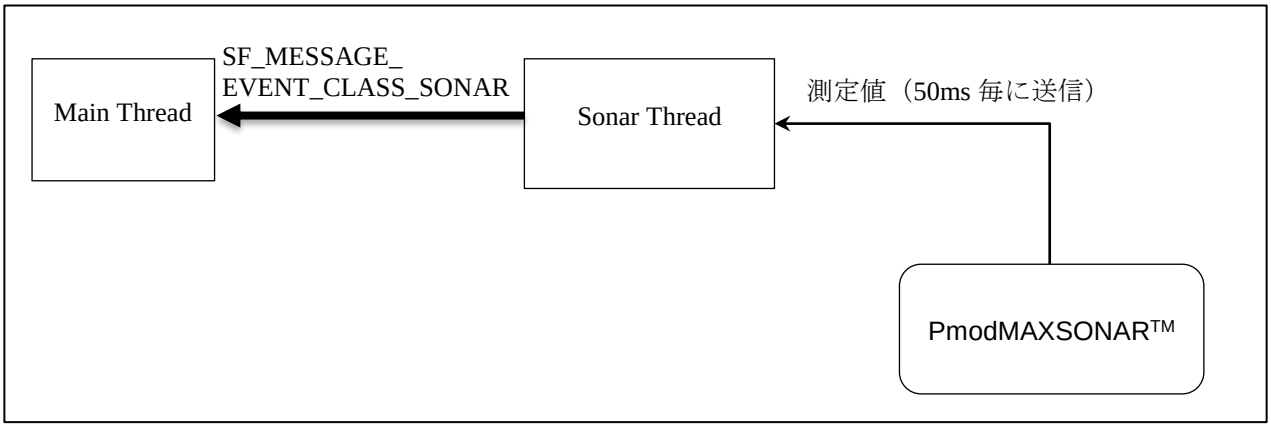

図 5-7 Sonar スレッドの機能概要

5.2.6 Timer スレッド

Main スレッドにより設定された時間（秒）のタイマを起動し、そのタイマがオーバフローすると、SF_MESSAGE_EVENT_CLASS_TIMER メッセージを Main スレッドへ送信します（図 5-8）。

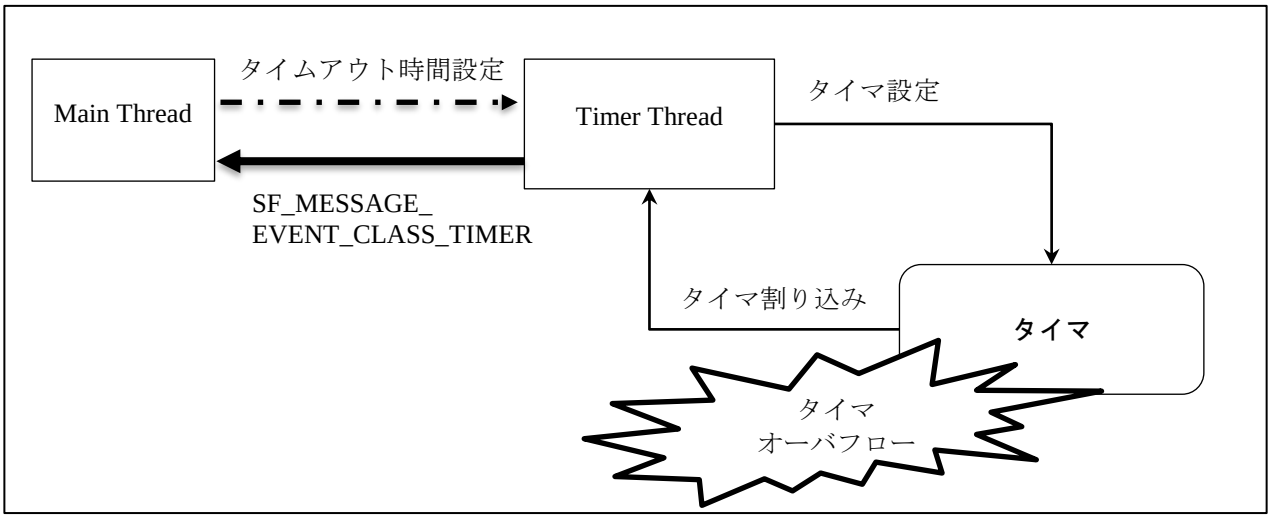

図 5-8 Timer スレッドの機能概要

5.2.7 RTC スレッド

RTC を制御し、1 秒経過する毎に取得した時刻を SF_MESSAGE_EVENT_CLASS_RTC メッセージに格納して Main スレッドへ送信します（図 5-9）。

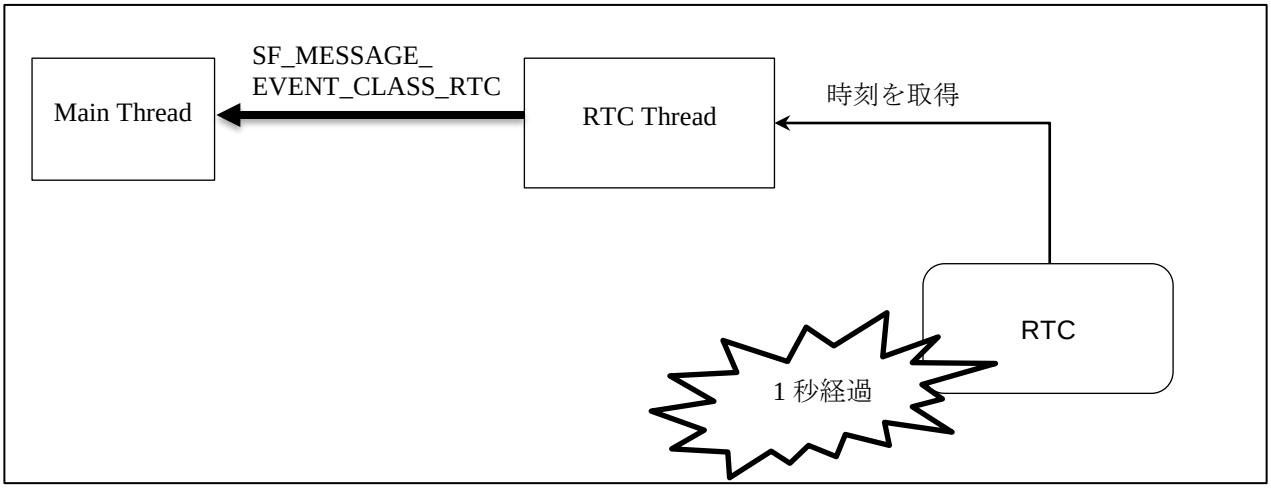

図 5-9 RTC スレッドの機能概要

5.3 モニタ上で動作するスレッド

5.3.1 BLE スレッド

rBLE API を用いて、RL78/G1D を制御します。起動後、Random Address の変更や Device 検索を行ない、Complete Local Name が「DoorLockBLE」のデバイス（電子錠ドア）を発見した場合、接続を開始します。接続完了後、プロファイル（CSCP）の有効化と Notification の許可を行います。また、電子錠ドアから Notification を受信した場合は、そのデータから時刻、距離、状態を取り出し保持します。保持したデータは GLCD スレッドが LCD 画面を更新する際に用います。また、rBLE API を使用する毎に、その内容を SF_MESSAGE_EVENT_CLASS_LOG メッセージに格納し、Log スレッドへ送信します（図 5-10）。

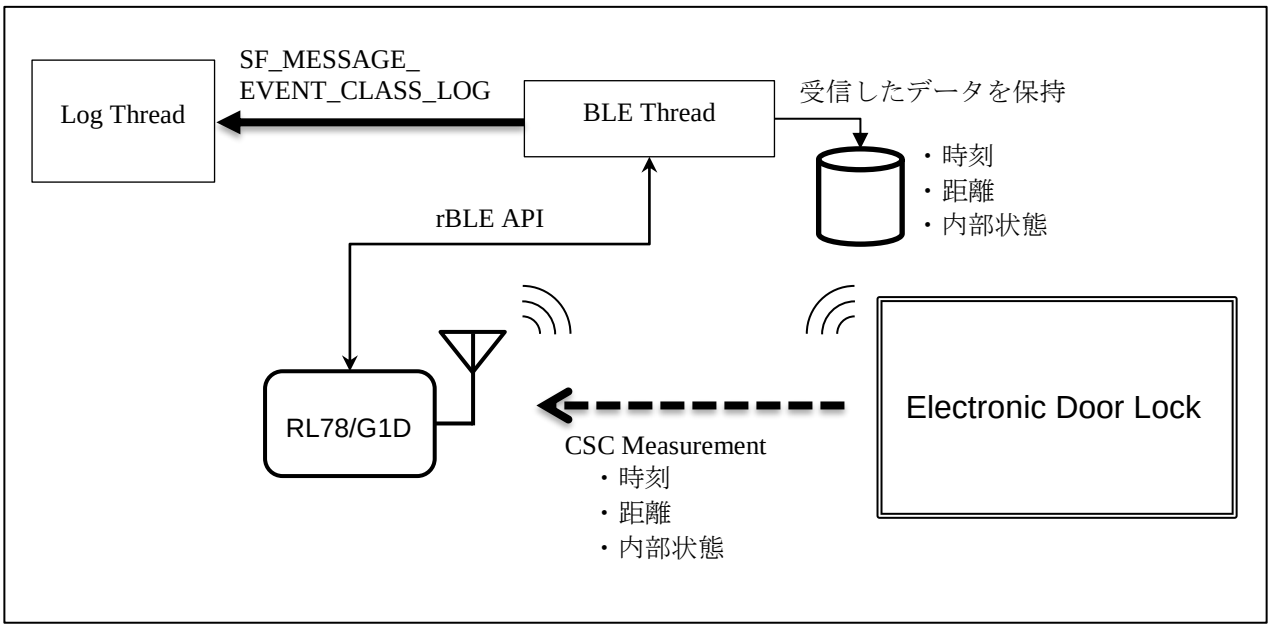

図 5-10 BLE スレッドの機能概要

5.3.2 GLCD スレッド

GUIX を用いて、SK-S7G2 に実装されている LCD を制御します。BLE スレッドから取得した時刻や距離、状態、また Log スレッドから取得したログデータを基に LCD 画面を更新します（図 5-11）。

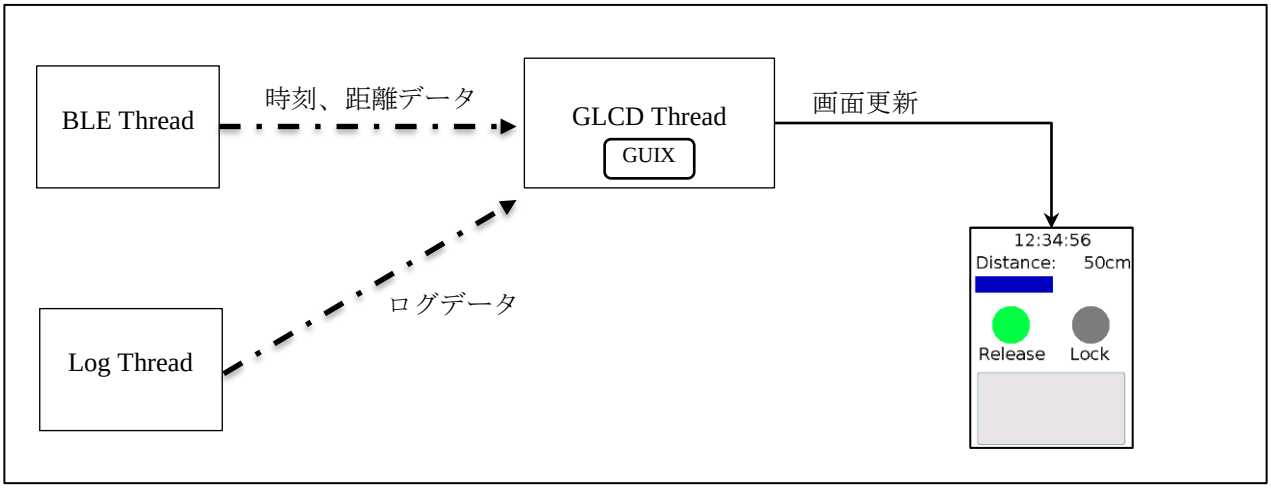

図 5-11 GLCD スレッドの機能概要

5.3.3 Log スレッド

BLE スレッドから SF_MESSAGE_EVENT_CLASS_LOG メッセージを受信すると、そのメッセージに格納されているログを保持します。このログは GLCD スレッドが LCD の画面を更新する際に用います（図 5-12）。

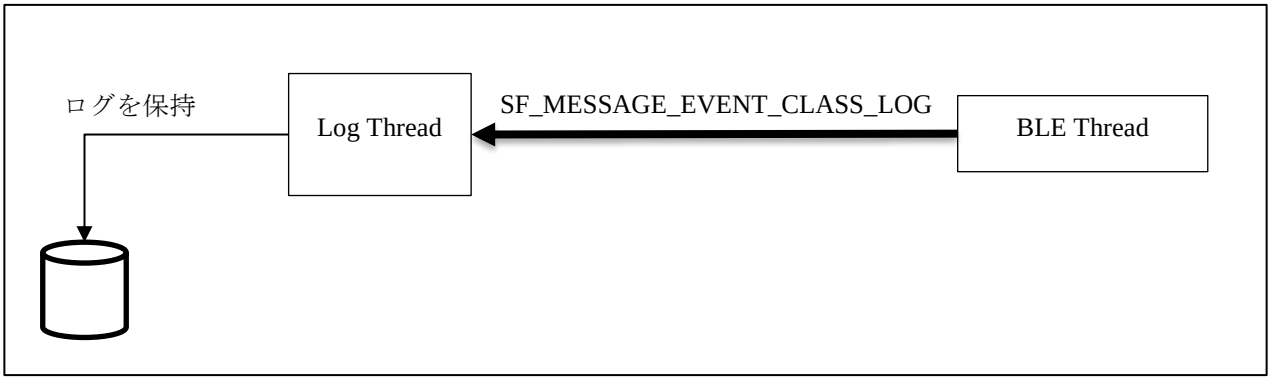

図 5-12 Log スレッドの機能概要

6. Appendix

6.1 電子錠ドアの GUI 画面作成、制御方法

本章では、電子錠ドアの LCD に表示している GUI 画面を構成する以下の 4 つの Widget（図 6-1）について、GUIX Studio (v5.3.2.2)での作成方法と、GUIX API を用いて制御する方法を説明します。詳細は参考文献[6], [7]を参照してください。

- Text Button
- Progress Bar
- Prompt
- Single Line Input

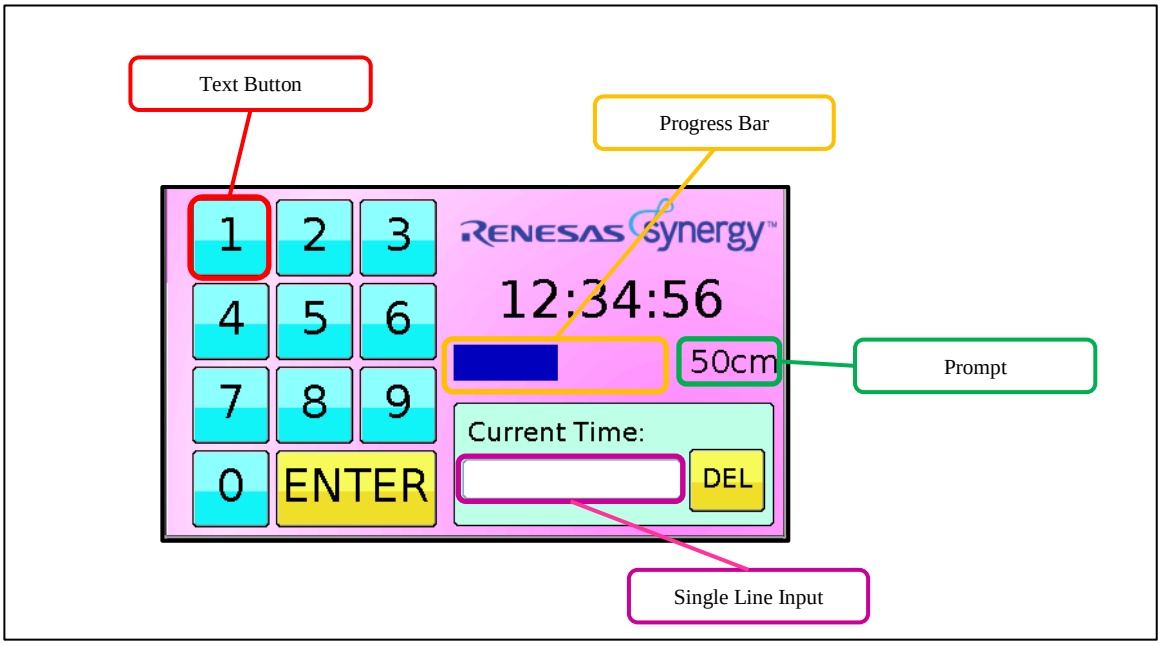

図 6-1 本アプリケーションノートで説明する Widget

本章では、以下に格納されている GUIX Studio プロジェクトを用いて説明します。このプロジェクトは図 6-1 内で赤枠、黄枠、緑枠、紫枠で囲まれた Widget を削除したものです。次節からこれらを作成し、アプリケーションから制御する方法を説明します。

ElectronicDoorLock_Door_DK_S7G2¥guix_studio_example¥EDL_display¥

6.1.1 Text Button

本節では、テンキーの「1」の作成とアプリケーションからの制御方法について説明します。

まず、Project View 内の main_screen（テンキーなどを表示する Window）を右クリック、[Insert] → [Button] → [Text Button]を選択し、Text Button を作成します。次に、図 6-2 のようにマウスでテンキーの位置へ移動し、サイズを変更します。もしくは、Properties View 内で Left, Top, Width, Height を指定することでも可能です（表 6-1 参照）。

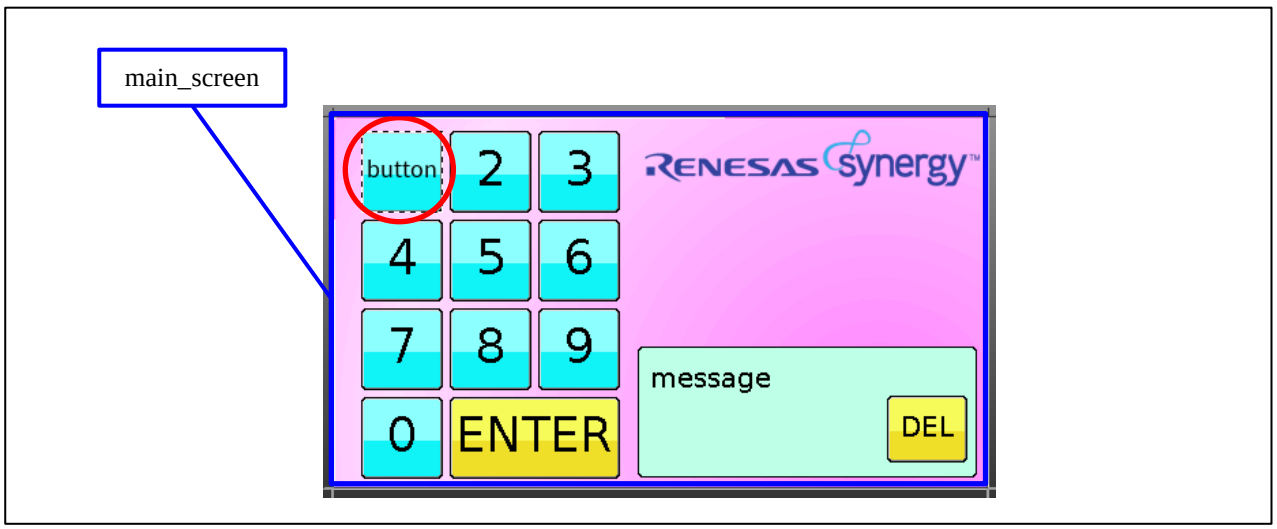

図 6-2 Text Button の作成

次に Properties View で表 6-1 のように設定します。Widget Name はアプリケーションから制御する際に利用し、button_1 と設定することで main_screen.main_screen_button_1 として作成したボタンにアクセスできるようになります（main_screen Window 内に button_1 という Widget Name で Text Button を作成したことで、このような変数が定義されます。他の Widget でも同様です）。また、Widget Id についても同様にアプリケーションが作成したボタンとして識別するための ID になります。ボタンの色については、Normal fill（ボタンの下半分）と Selected fill（ボタンの上半分）で指定します。

表 6-1 Text Button の Properties View

Widget Name	button_1
Widget Id	ID_BUTTON_1
Left	20
Top	10
Width	60
Height	60
Normal fill	BTN_LOWER
Selected fill	BTN_UPPER
Text	1
Font	Font_NumericKeypad

次に、ボタンが押された際のイベントを受け取る方法を図 6-3 に示します。図中の赤字部分のように、GX_SIGNAL マクロを用いて、第 1 引数には作成したボタンの Widget Id、第 2 引数にはイベント種類（GX_EVENT_CLICKED）を指定することで、ボタンが押された際のイベントを受け取ることができます。図中のイベントハンドラは、main_screen の Properties View 内の Event Function で指定した関数になります。

glcd_event_handler.c:

```
UINT MainScreenEventHandler (GX_WINDOW * widget, GX_EVENT * event_ptr)
{
    /* Omitted */
    switch (event_ptr->gx_event_type)
    {
        /* Omitted */
        case GX_SIGNAL(ID_BUTTON_1, GX_EVENT_CLICKED):
            /* Omitted */
            break;
```

図 6-3　ボタンが押された際のイベント

6.1.2　Progress Bar

本節では、超音波センサの距離を表示するための Progress Bar の作成とアプリケーションからの制御方法について説明します。

まず、Project View 内の main_screen を右クリック、[Insert] → [Indicator] → [Progress Bar]を選択し、Progress Bar を作成します。次に、図 6-4 のようにマウスで移動し、サイズを変更します。

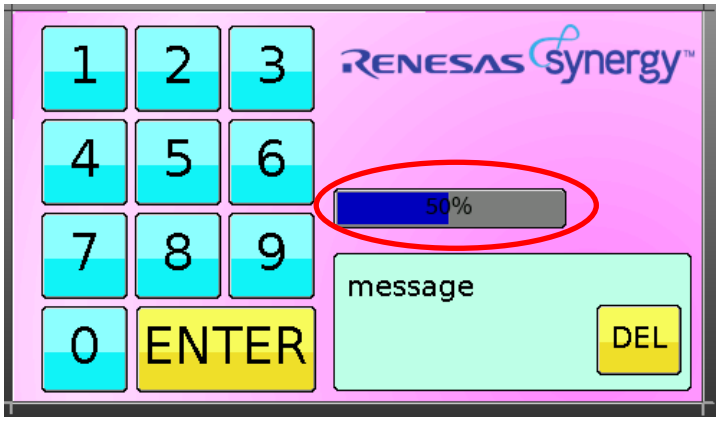

図 6-4　Progress Bar の作成

次に Properties View で表 6-2 のように設定します。Border に No Border を設定することで、Progress Bar の枠を非表示にし、Transparent のチェックを付けることで、Progress Bar 内の灰色の部分が透過色になります。また、Show Text、Show Percentage のチェックを外すことで、パーセンテージ表示を削除することができます。また、Min Value には 0 を、Max Value には 100 を設定することで、Progress Bar の値の範囲を 0～100 に設定します。

表 6-2　Progress Bar の Properties View

Widget Name	progress_bar_distance
Border	No Border
Transparent	チェックを付ける
Show Text	チェックを外す
Show Percentage	チェックを外す
Min Value	0
Max Value	100

次に作成した Progress Bar の値（青色の部分：Current Value）をアプリケーションから変更する方法を図 6-5 に示します。図中(glcd_event_handler.c)の赤字部分のように、gx_progress_bar_value_set()関数を用いて、第 1 引数には作成した Progress Bar へのポインタ、第 2 引数には設定したい値を指定することで、Progress Bar の値を変更することができます。本アプリケーションでは、第 2 引数に超音波センサが計測した距離を 0～100（単位は cm）の範囲の値で返す get_sonar_distance()の戻り値を設定しているため、Progress Bar の値は超音波センサが計測した距離（単位：cm）に応じて変更されることになります。また、この処理は GX_EVENT_TIMER イベント契機で実行されます。GX_EVENT_TIMER イベントを発生されるためには、gx_system_timer_start()関数を用いて、第 1 引数には、Widget Control Block へのポインタ、第 2 引数にはタイマの ID、第 3 引数には最初のタイムアウトが発生するまでの間隔（Tick）、第 4 引数には 2 回目以降の周期的なタイムアウトが発生する間隔（Tick）を指定します。本アプリケーションでは、第 3,4 引数に 25 を設定しており、GUIX の Tick は 20ms に設定していることから、タイムアウトは 25 [tick] × 20 [ms/tick] = 500 [ms]毎に発生します。

glcd_event_handler.c:

```
UINT MainScreenEventHandler (GX_WINDOW * widget, GX_EVENT * event_ptr)
{
    /* Omitted */
    switch (event_ptr->gx_event_type)
    {
        /* Omitted */
        case GX_EVENT_TIMER:
            /* Omitted */
            retval = gx_progress_bar_value_set(
                        &main_screen.main_screen_progress_bar_distance,
                        get_sonar_distance());
```

glcd_thread_entry.c:

```
void glcd_thread_entry (void)
{
    /* Omitted */
    status = gx_system_timer_start(p_first_screen,
                                   GUIX_TIMER_ID_DRAW,
                                   GUIX_TICKS_FOR_500MS,
                                   GUIX_TICKS_FOR_500MS);
```

図 6-5 Progress Bar の値の変更方法

6.1.3 Prompt

本節では、超音波センサの距離を数値として表示するための Prompt の作成とアプリケーションからの制御方法について説明します。

まず、Project View 内の main_screen を右クリック、[Insert] → [Text] → [Prompt]を選択し、Progress Bar を作成します。次に、図 6-6 のようにマウスで移動し、サイズを変更します。

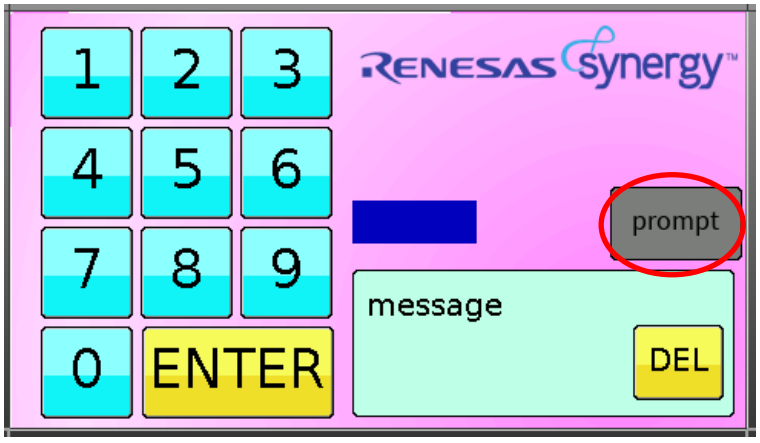

図 6-6 Prompt の作成

次に Properties View で表 6-3 のように設定します。Border に No Border を設定することで、Prompt の枠を非表示にし、Transparent のチェックを付けることで、Prompt 内の灰色の部分が透過色になります。また、Accept Focus のチェックを外すことで、この Prompt を選択できないようにします（チェックが付いている場合、選択されるとテキストの色が Selected Text Color で指定された色に従って変更されます）。

表 6-3 Progress Bar の Properties View

Widget Name	prompt_distance
Border	No Border
Transparent	チェックを付ける
Accept Focus	チェックを外す
Text	50cm
Font	Font_PromptDistance

次に作成した Prompt の文字列をアプリケーションから変更する方法について説明します。図 6-7 の赤字部分のように、gx_prompt_text_set()関数を用いて、第 1 引数には作成した Prompt へのポインタ、第 2 引数には設定したい文字列へのポインタ（本アプリケーションでは、str_distance[]に文字列が設定されています）を指定することで、Prompt に表示する文字列を変更することができます。

glcd_event_handler.c:

```
UINT MainScreenEventHandler (GX_WINDOW * widget, GX_EVENT * event_ptr)
{
    /* Omitted */
        retval = gx_prompt_text_set(
                    &main_screen.main_screen_prompt_distance,
                    str_distance);
```

図 6-7 プロンプトの文字列の変更方法

6.1.4 Single Line Input

本節では、時刻やパスコードを入力するための Single Line Input の作成とアプリケーションからの制御方法について説明します。

まず、Project View 内の main_screen を右クリック、[Insert] → [Text] → [Single Line Input]を選択し、Single Line Input を作成します。次に、図 6-8 のようにマウスで移動し、サイズを変更します。

図 6-8　Single Line Input の作成

次に Properties View で表 6-4 のように設定します。Accept Focus のチェックを外すことで、この Single Line Input を選択できないようにします。また、Single Line Input 内の色を変更するため、Normal fill に WHITE を設定します。また、Single Line Input に表示する文字列の色を変更するため、Normal Text Color に TEXT を設定します。

表 6-4　Single Line Input の Properties View

Widget Name	text_input
Accept Focus	チェックを外す
Normal fill	WHITE
Font	Font_TextInput
Normal Text Color	TEXT

次に作成した Single Line Input の文字列をアプリケーションから変更する方法を図 6-9 に示します。Single Line Input に文字列を挿入するためには、図中の 1 つ目の赤字部分のように、gx_single_line_text_input_character_insert()関数を用いて、第 1 引数には作成した Single Line Input へのポインタ、第 2 引数には表示したい文字列、第 3 引数に文字列の長さを指定します。また、Single Line Input 内の文字列から 1 文字削除するためには、図中の 2 つ目の赤字部分のように、glcd_input_character_backspace()関数を用いて、第 1 引数に Single Line Input へのポインタを指定します。また、Single Line Input の文字列全てを削除したい場合は、図中の 3 つ目の赤字部分のように、gx_single_line_text_input_buffer_clear()関数を用いて、第 1 引数に Single Line Input へのポインタを指定します。

glcd_event_handler.c:

```
void glcd_input_character_insert (char *str)
{
    gx_single_line_text_input_character_insert(
        &main_screen.main_screen_text_input,
        (GX_UBYTE *)str,
        strlen(str));
}

void glcd_input_character_backspace (void)
{
    gx_single_line_text_input_backspace(&main_screen.main_screen_text_input);
}

void glcd_input_clear (void)
{
    gx_single_line_text_input_buffer_clear(&main_screen.main_screen_text_input);
}
```

図 6-9　Single Line Input に表示する文字列の操作方法

ホームページとサポート窓口<website and support,ws>

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2016.11.22	-	初版
1.10	2017.4.27	-	SSP1.2.0 に対応 IAR EW for Synergy に対応

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. 未使用端子の処理
   【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】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. リザーブアドレス（予約領域）のアクセス禁止
   【注意】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、将来の機能拡張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。
4. クロックについて
   【注意】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。
5. 製品間の相違について
   【注意】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、内部ROM、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、ノイズ輻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。お客様の機器・システムの設計において、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には、お客様の責任において行ってください。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、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、全部または一部を問わず、改造、改変、複製、その他の不適切に使用しないでください。かかる改造、改変、複製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。
5. 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、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、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できません。たとえ、意図し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。
6. 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。
7. 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。
8. 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。
9. 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。また、当社製品および技術を、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。
   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。
10. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
11. 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。
12. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

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その総株主の議決権の過半数を直接または間接に保有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.3.0-1　2016.11)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口　http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/