RX ファミリ
グラフィック LCD コントローラモジュールを用いた TFT-LCD パネルへの画像表示サンプルプログラム
Firmware Integration Technology

R01AN3509JJ0100
Rev.1.00
2017.10.02

要旨

本アプリケーションノートでは、グラッフィク LCD コントローラ(以下、GLCDC)および、その FIT モジュールを使用した TFT-LCD パネル(以下、LCD パネル)への画像表示方法について説明します。

対象デバイス

・RX65N グループ、RX651 グループ　　ROM 容量：1.5MB ～ 2MB

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

関連ドキュメント

Firmware Integration Technology ユーザーズマニュアル(R01AN1833)
ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)
e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)
CS+に組み込む方法 Firmware Integration Technology (R01AN1826)
Renesas e^2 studio スマート・コンフィグレータ ユーザーガイド(R20AN0451)

目次

1. 仕様

本アプリケーションノートのサンプルプログラムでは、GLCDC を使用して下記の動作を行います。

- 表示画像の切り替え　　：読み込んだ画像の切り替えができます。
- アルファブレンドの変更　　：表示画像の透明度の変更ができます。
- 輝度/コントラスト/ガンマの調整　：各設定値の変更ができます。

また、本サンプルプログラムで使用する LCD パネルはタッチ入力に対応しており、タッチ操作や RSK のスイッチを使用してサンプルプログラムを制御します。

図 1.1 に本アプリケーションノートの動作概要を示します。

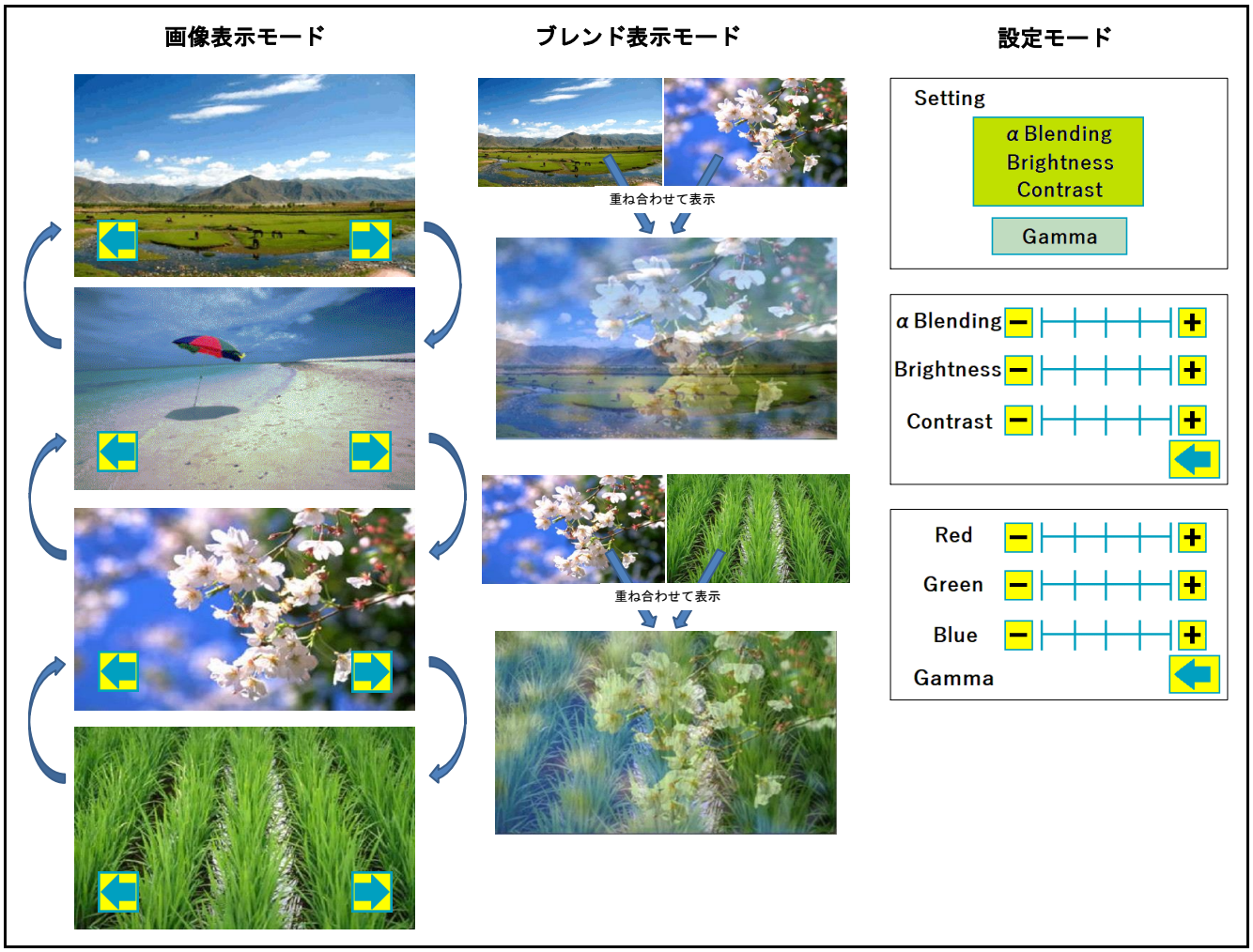


図 1.1 動作概要

サンプルプログラムは、下記の 3 つの画面モードで動作します。

■　画像表示モード

画像を 1 枚ずつ表示します。パネル上の矢印をタッチすることで、次の画像に切り替わります。

■　ブレンド表示モード

画像表示している画像のブレンド表示(上位層の画像の透過)を行います。パネル上をタッチすることで次の画像とのブレンド表示を行います。なお、アルファ値は「設定モード」で設定できます。

■　設定モード

「画像表示モード」および「ブレンド表示モード」の輝度/コントラスト/ガンマを調整することができます。それぞれ 5 段階で変更できます。また、アルファ値の調整は「ブレンド表示モード」のみ有効となります。設定モードの背景には、「画像表示モード」で表示している画像が使用されます。

上記機能を実現するため、本アプリケーションノートで使用する周辺機能を表 1.1 に示します。

表 1.1　使用する周辺機能と用途

周辺機能	用途
GLCDC	画面表示と制御
RSPI チャネル 1	SerialFlash との通信
SCI(簡易 I^2C モード) チャネル 7	LCD パネルのタッチコントローラとの通信
CMT チャネル 0	タッチ検出周期のタイマ

1.1　表示画像の構成

GLCDC は、グラフィック 1、グラフィック 2、バックグラウンド画面の 3 層構造で情報を持っており、それらの情報をもとに加工を行い、1 枚の画像として LCD パネルに出力しています。グラッフィク 1 とグラフィック 2 には各々画像を指定でき、表示させる層も動的に切り替えることができます。また、アルファブレンドの機能により、特定の色を透過し、重ね合わせた画像を生成する機能もあります。

本サンプルプログラムではその機能を使用し、図 1.2 で示すように画像を重ね合わせて出力しています。

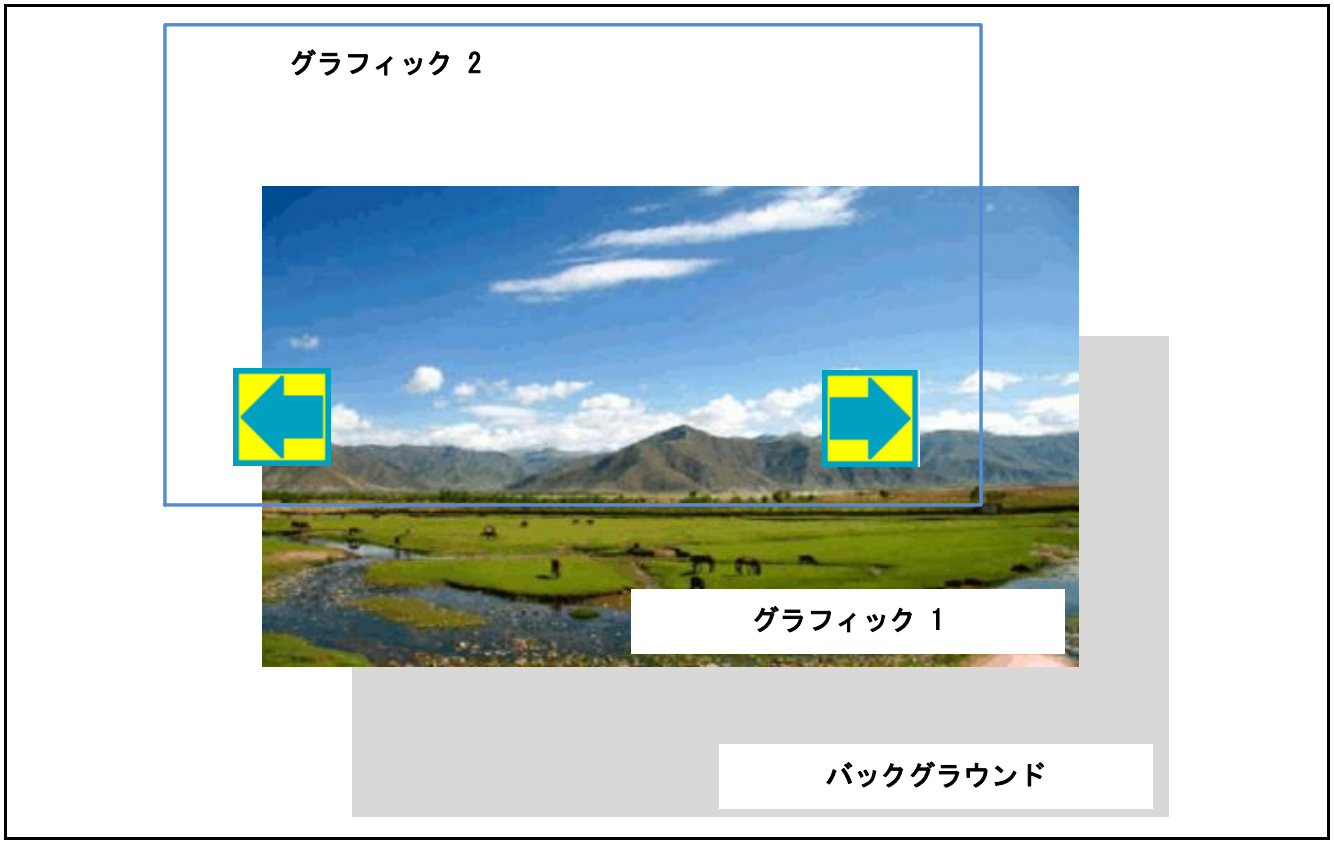


図 1.2　表示画像の構成

1.2　使用する画像データ

本サンプルプログラムでは、表 1.2 で示す画像を使用しています。画像データは基本的に SerialFlash へ格納しており、動作開始時および必要に応じて都度、画像データを SerialFlash から読み出します。

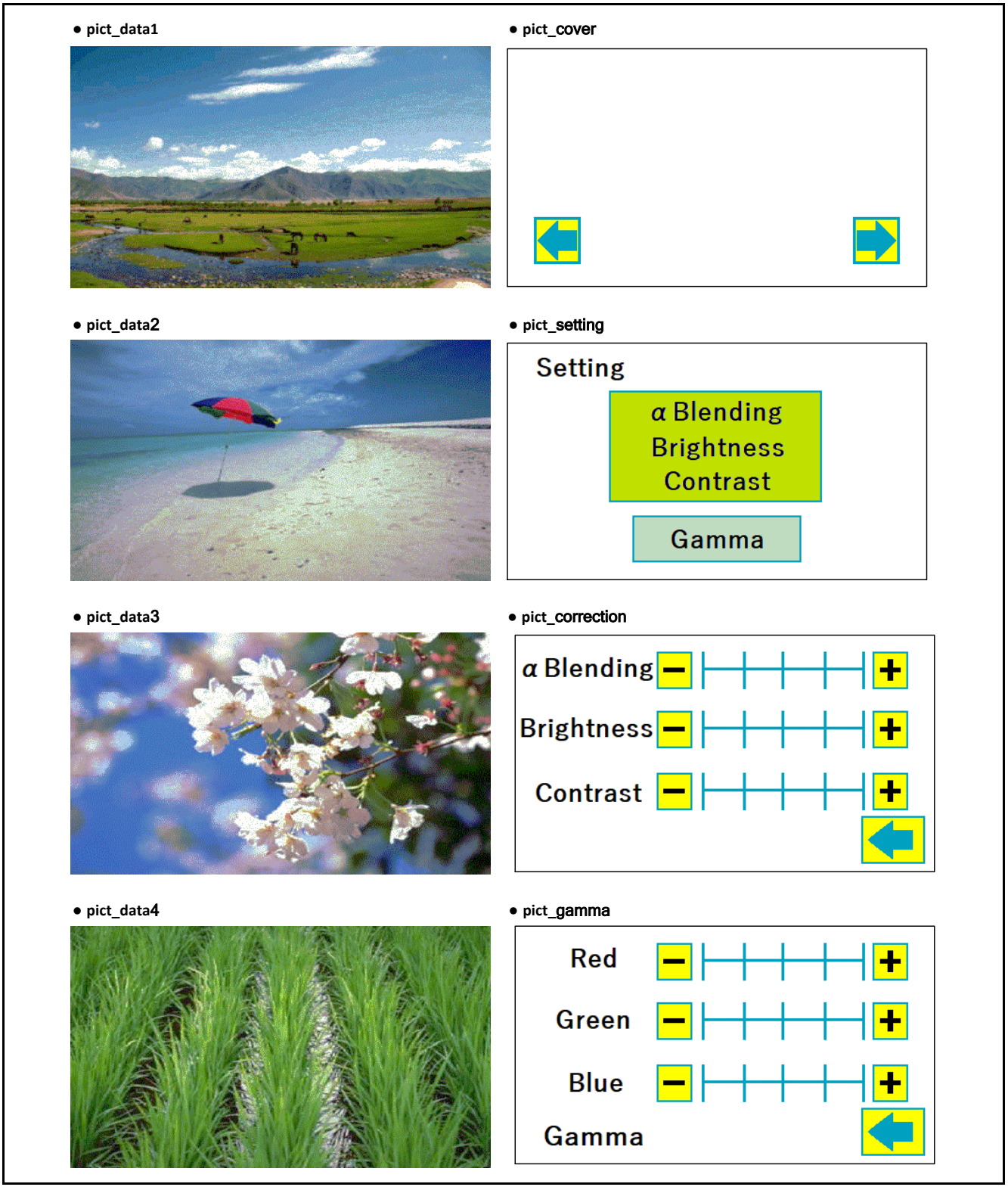


図 1.3　使用する画像

表 1.2　使用する画像の情報

ファイル名	画像サイズ(byte)	画像形式	説明
pict_data1.bmp	114,422	BMP (256 色)	表示画像 1
pict_data2.bmp	114,422		表示画像 2
pict_data3.bmp	114,422		表示画像 3
pict_data4.bmp	114,422		表示画像 4
pict_cover.bmp	114,422		重ね合わせ画像 (切り替えボタン)
pict_setting.bmp	114,422		重ね合わせ画像 (設定選択画面)
pict_correction.bmp	114,422		重ね合わせ画像 (輝度・コントラスト設定画面)
pict_gamma.bmp	114,422		重ね合わせ画像 (ガンマ設定画面)
合計サイズ	915,376	-	

画像の形式は BMP (Windows Bitmap Image) を使用します。

画像の大きさは 448×253 としており、256 色(8bit)で作成しています。

GLCDC で表示させる画像の横幅は、1 ピクセルあたりのバイトサイズ×画像の横幅(ピクセル) が 64 バイトで割り切れる幅を設定する必要があります。BMP(256 色)の画像形式では、1 ピクセルあたりのサイズが 8 ビットになるため、画像の横幅を 448 ピクセル (448 / 64 バイト = 7)としています。

1.3　タッチ制御

本サンプルプログラムで使用する LCD パネルにはタッチコントローラが搭載されており、このコントローラと通信を行ってタッチ入力の情報を取得します。RX65N をマスタ、タッチコントローラをスレーブとして I^2C プロトコルで通信を行います。

定期的にタッチコントローラとの通信を行い、タッチ入力の情報を取得して、画面上のボタンが押されたかどうかを判定します。

押されたボタンに応じて、表示画像の切り替え、アルファ値/輝度/コントラスト/ガンマの調整を行います。

2. 動作確認条件

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

項目	内容
使用マイコン	R5F565NEDDFC (RX65N グループ)
動作周波数	メインクロック: 24MHz PLL: 240MHz (メインクロック 1 分周 10 逓倍) システムクロック(ICLK): 120MHz (PLL 2 分周) 周辺モジュールクロック A(PCLKA): 120MHz (PLL 2 分周) 周辺モジュールクロック B(PCLKB): 60MHz (PLL 4 分周) LCD パネルクロック(LCD_CLK): 10MHz (PLL 24 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 e^2 studio Version　6.0.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V.2.07.00
	コンパイルオプション -lang = c99 <画像表示用プロジェクト> セクションに RIMAGE (0x00800000) を追加 <SerialFlash 書き込み用プロジェクト> セクションに IMAGE (0xFFE80000) を追加 リンカに binary オプションを追加 (詳細は 5.2.1 プロジェクトのプロパティに記載)
iodefine.h のバージョン	Version 2.0
エンディアン	リトルエンディアン、ビッグエンディアン
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 1.00
使用ボード	Renesas Starter Kit+ for RX65N-2MB (製品型名: RTK50565N2SXXXXXBE) (以下、RSK)

3.　使用 FIT モジュール

本アプリケーションノートで使用している FIT モジュールを以下に示します。併せて参照してください。

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 グラフィック LCD コントローラモジュール Firmware Integration Technology (R01AN3609)
RX ファミリ RSPI クロック同期式シングルマスタ制御モジュール Firmware Integration Technology（R01AN1914）
RX ファミリ Serial Flash memory アクセス クロック同期式制御モジュール Firmware Integration Technology（R01AN2662）
RX ファミリ　簡易 I^2C モジュール　Firmware Integration Technology (R01AN1691)
RX ファミリ CMT モジュール Firmware Integration Technology (R01AN1856)

最新版がある場合、最新版に差し替えて使用してください。最新版はルネサスエレクトロニクスホームページで確認および入手してください。

4. プロジェクトの実行方法

本章では、サンプルプログラムの実行方法について説明します。サンプルプログラムでは下記のとおり、プロジェクトを 2 つ用意しています。

- glcdc_main_rx65n
- serialflash_writer_rx65n

glcdc_main_rx65n は、画像を表示する本アプリケーションノートのメインとなるプロジェクトです。SerialFlash に格納された画像データを読み出し、LCD パネルへ表示するため、必要な画像データをあらかじめ SerialFlash に書き込んでおく必要があります。

serialflash_writer_rx65n は、SerialFlash へ画像データを書き込むためのプロジェクトです。こちらのプロジェクトを実行することで、SerialFlash へ画像データを書き込むことができます。なお、本プロジェクトでは SerialFlash のプロテクト解除は行っておりません。プロテクトが掛かっている状態で本プロジェクトを実行すると書き込みに失敗します。また、本プロジェクトは画像データ書き込みの際に、SerialFlash の全データを消去しますのでご了承ください。

glcdc_main_rx65n は、画像データの読み出しのみ行いますので、書かれたデータが消えることはありません。

以下の順に、サンプルプログラムの実行方法を説明します。

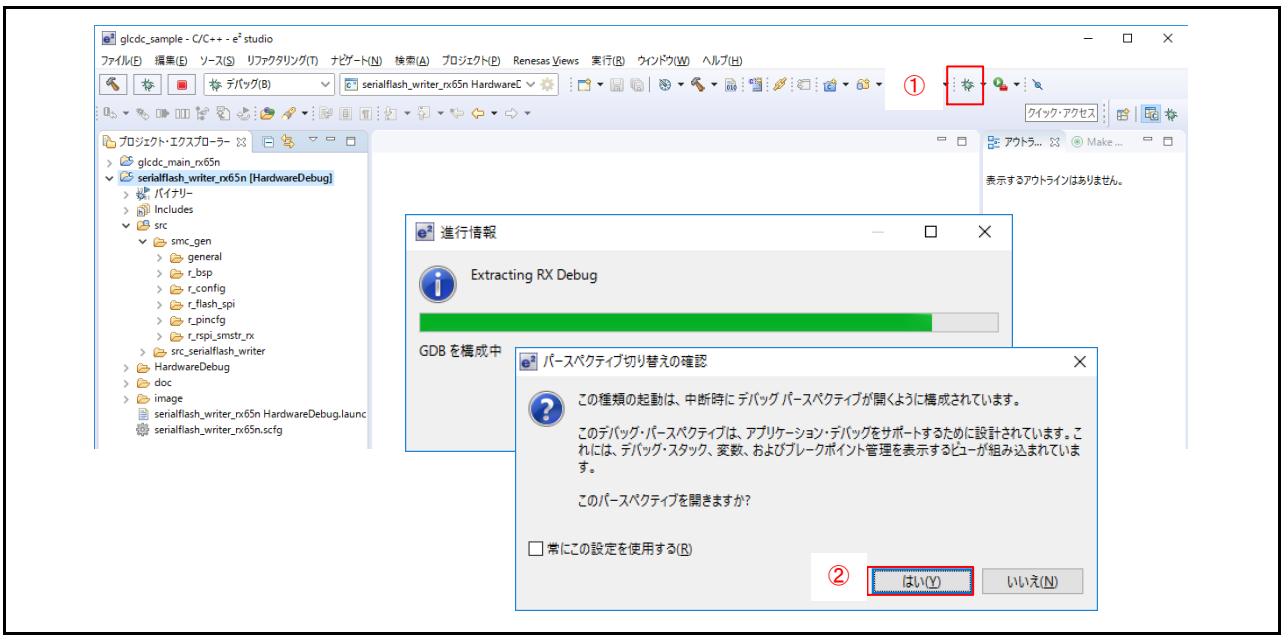
1. SerialFlash 書き込み用のプロジェクトの実行
2. 画像表示用のプロジェクトの実行

4.1　SerialFlash 書き込み用のプロジェクトの実行

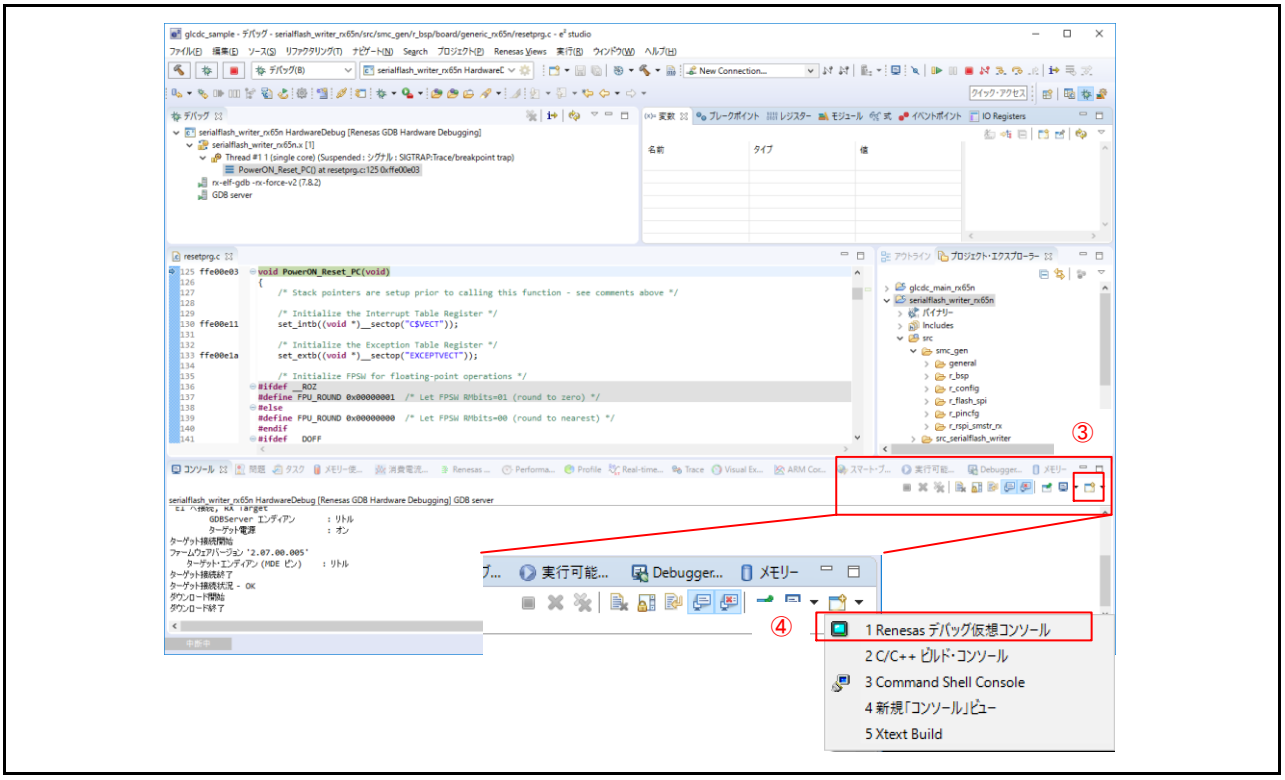
e² studio を起動後、serialflash_writer_rx65n プロジェクトをインポートし、デバッガ接続して実行します。画像データの書き込みステータスを Renesas デバッグ仮想コンソール(e² studio の場合)に出力しています。画像の書き込みが完了すると、LED0 を点灯します。もし書き込みでエラーが発生した場合は LED3 を点灯します。

以下に、e² studio を例にプロジェクトの実行手順を示します。(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。)

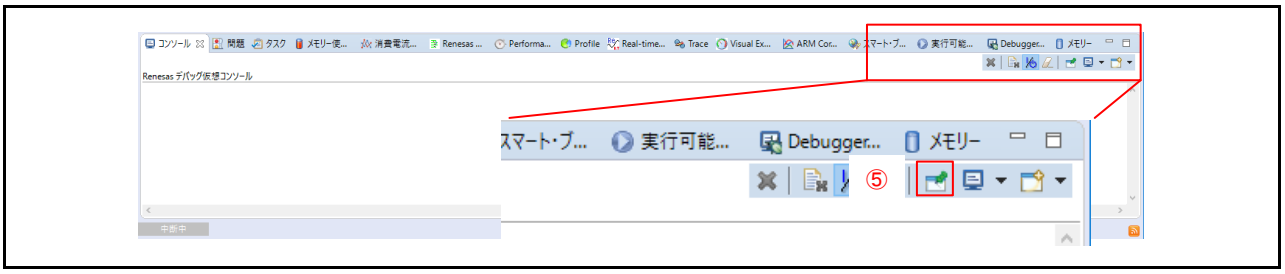
1. 画面の上部にあるデバッグボタン（①）をクリックします。デバッガが起動し、進行情報が表示された後、パースペクティブ切り替えの確認ウィンドウが表示されますので「はい（Y）」（②）をクリックします。



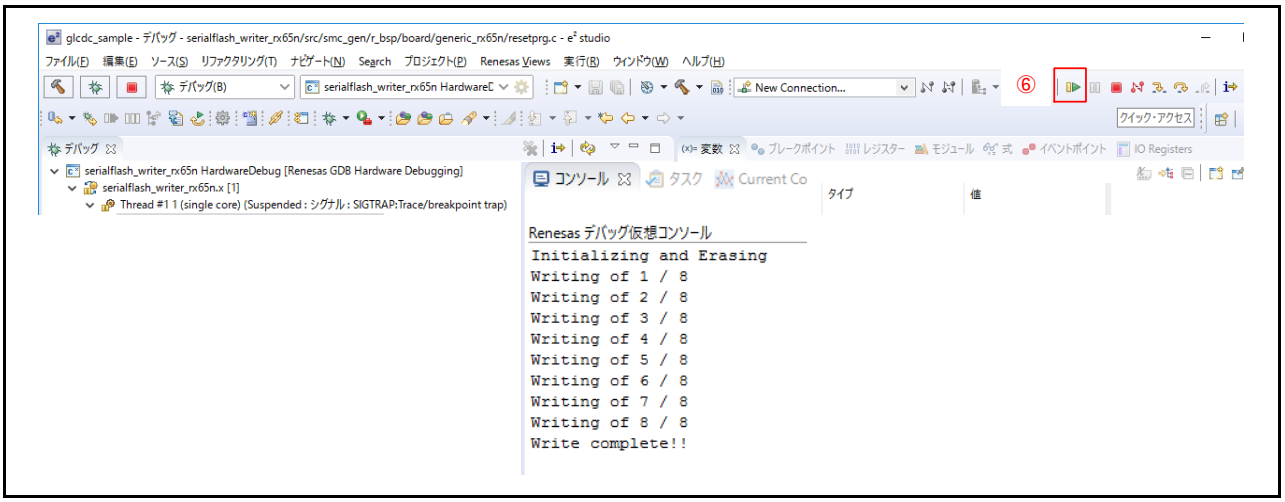
2. ウィンドウがデバッグのパースペクティブに切り替わります。切り替わり後、コンソールウィンドウの右端にある「コンソールを開く」ボタン（③）をクリックすると、サブメニューが表示されますので、「1. Renesas デバッグ仮想コンソール」（④）をクリックします。



3. コンソールが Renesas デバッグ仮想コンソールに切り替わりますので、「コンソールのピン留め」（⑤）をクリックします。



4. 「再開(M)(F8)」（⑥）をクリックしてプログラムを実行すると、Renesas デバッグ仮想コンソールにメッセージが表示されます。「Write complete!!」が表示されると書き込み完了となり、RSK の LED0 が点灯します。あとは、デバッガを切断して終了してください。

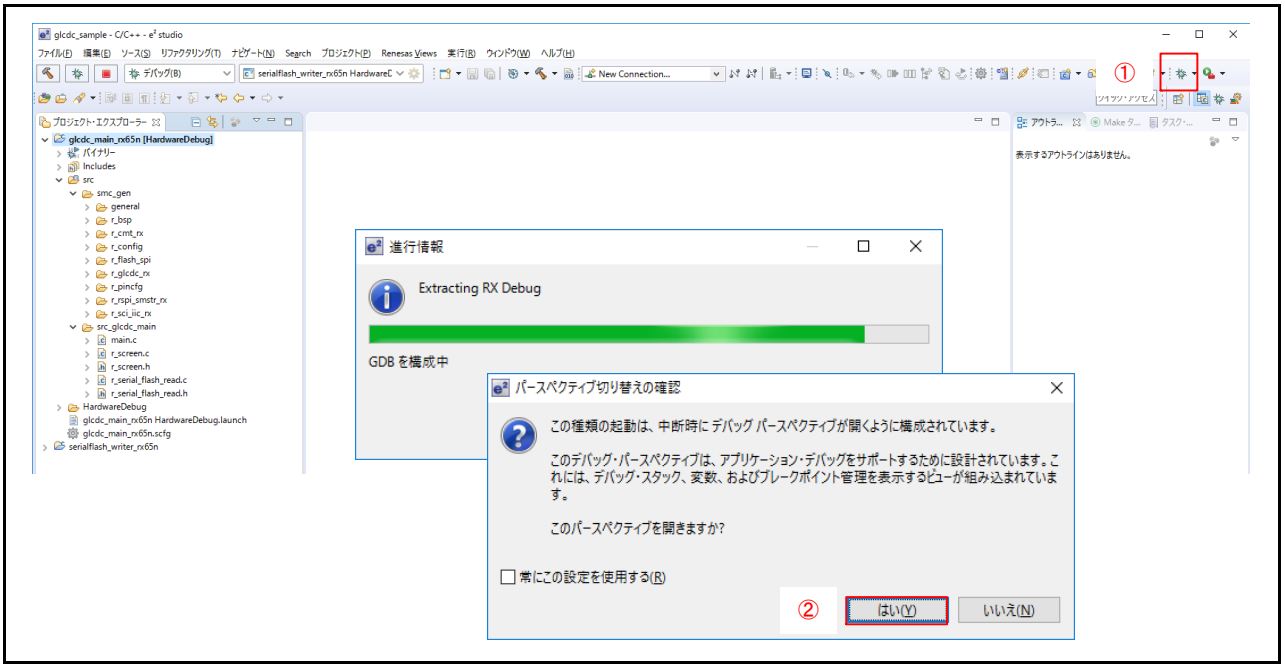


4.2　画像表示用のプロジェクトの実行

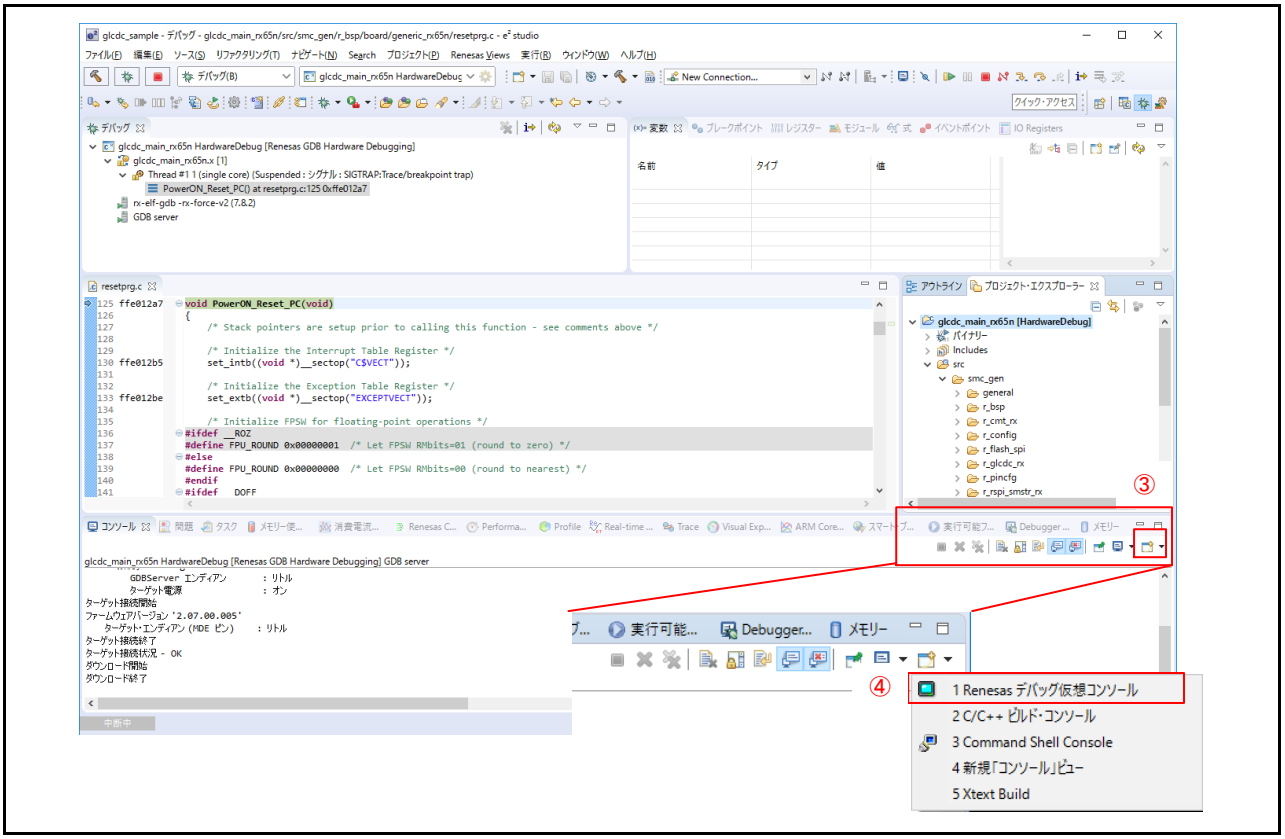
glcdc_main_rx65n プロジェクトをインポートし、デバッガ接続して実行します。もし、画像データが SerialFlash に書き込まれていない状態で実行した場合、Renesas デバッグ仮想コンソール(e² studio の場合)にエラーを出力、および LED3 を点灯します。

以下に、e² studio を例にプロジェクトの実行手順を示します。(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。)

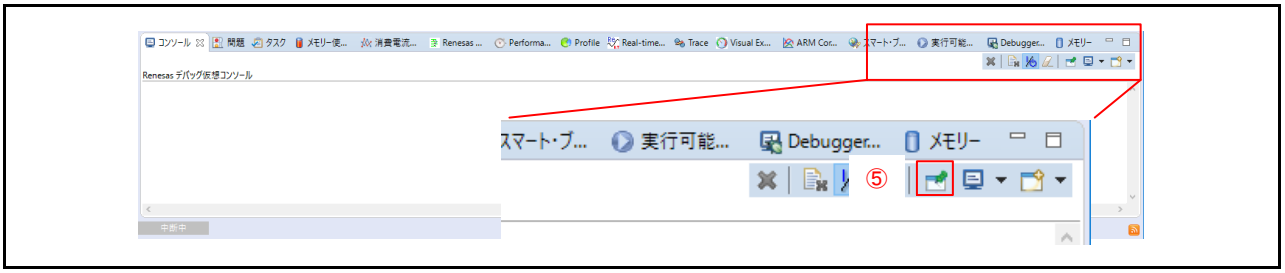
1. 画面の上部にあるデバッグボタン（①）をクリックします。デバッガが起動し、進行情報が表示された後、パースペクティブ切り替えの確認ウィンドウが表示されますので「はい（Y）」（②）をクリックします。



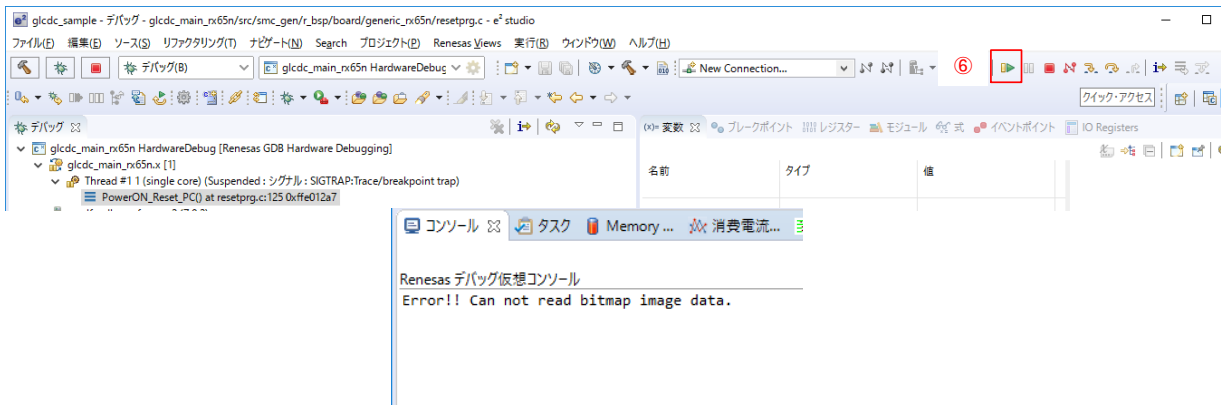
2. ウィンドウがデバッグのパースペクティブに切り替わります。切り替わり後、コンソールウィンドウの右端にある「コンソールを開く」ボタン（③）をクリックすると、サブメニューが表示されますので、「1. Renesas デバッグ仮想コンソール」（④）をクリックします。



3. コンソールが Renesas デバッグ仮想コンソールに切り替わりますので、「コンソールのピン留め」（⑤）をクリックします。



4. 「再開(M)(F8)」（⑥）をクリックしてプログラムを実行します。SerialFlash から画像データを読み出し LCD パネルに画像が表示されます。SerialFlash から画像データが読み出せない場合は、Renesas デバッグ仮想コンソールにメッセージが表示され、RSK の LED3 が点灯します。



※SerialFlashから画像データが読み出せない場合

5.　プロジェクトの変更情報

サンプルプログラムのプロジェクトは各 FIT モジュールのコンフィグレーションファイルとソースフォルダ、およびプロジェクト設定を変更しています。以下に詳細を示します。

なお、本変更情報は、新規にプロジェクトを構築する場合に参照してください。インポートしたプロジェクトを使用する場合は「6. ハードウェア説明」に進んでください。

5.1　コンフィグレーションファイルの変更

以下にコンフィグレーションファイルの変更箇所を示します。変更内容は画像表示用のプロジェクト「glcdc_main_rx65n」および SerialFlash 書き込み用のプロジェクト「serialflash_writer_rx65n」で共通です。コンフィグレーションファイルの項目と設定内容については、各 FIT モジュールのマニュアルを参照してください。なお、変更の記載が無いものはデフォルト設定のままです。

(1)　RSPI クロック同期式シングルマスタ制御モジュールの設定変更

チャネル 1 を有効にするために以下のように設定します。

【 r_config/ r_rspi_smstr_rx_config.h 】

```
/***********************************************************************************
SPECIFY CHANNELS TO INCLUDE SOFTWARE SUPPORT
***********************************************************************************/
/* If these are defined, then the code for the specified channel is valid.
     If the #define which channel is not supported on the MCU is uncommented,
     then the compile error occurs. */

/* #define for RSPI channel 0 to be valid. */
/* #define RSPI_SMSTR_CFG_CH0_INCLUDED */

/* #define for RSPI channel 1 to be valid. */
#define RSPI_SMSTR_CFG_CH1_INCLUDED

/* #define for RSPI channel 2 to be valid. */
/* #define RSPI_SMSTR_CFG_CH2_INCLUDED */
```

(2)　Serial Flash memory アクセス クロック同期式制御モジュールの設定変更

DEV0 との通信にチャネル 1 が使用できるよう以下のように設定します。

【 r_config/ r_flash_spi_config.h 】

```
/***********************************************************************************************************
CHANNEL NUMBER OF DRIVER INTERFACE
***********************************************************************************************************/
/* Channel number of the driver interface to use in the Flash memory. */
/* Set number of the driver interface. */
#define FLASH_SPI_CFG_DEV0_DRVIF_CH_NO   (1)          /* Device 0 Channel Number        */
#define FLASH_SPI_CFG_DEV1_DRVIF_CH_NO   (0)          /* Device 1 Channel Number        */
```

DEV0 との通信レートを以下のように設定します。

【 r_config/ r_flash_spi_config.h 】

```
/***********************************************************************************************
TRANSFER RATE
***********************************************************************************************/
/* Define the transfer rate for using MCU driver interface.
    Necessary to set command transmission, data transmission and data reception.
        e.g. (1) The transfer rate is RSPI Bit Rate Register (SPBR) for using RX RSPI driver.
        e.g. (2) The transfer rate is QSPI Bit Rate Register (SPBR) for using RX QSPI driver.
        e.g. (3) The transfer rate is Bit Rate Register (BBR) for using RX SCIFA driver.
/* Max transfer rate is 5.00MHz for Renesas Serial FLASH. */

#define FLASH_SPI_CFG_DEV0_BR                   (uint8_t)(0x01)
                                                /* Device 0 Transfer rate for command transmission.      */
#define FLASH_SPI_CFG_DEV0_BR_WRITE_DATA    (uint8_t)(0x01)
                                                /* Device 0 Transfer rate for data transmission.         */
#define FLASH_SPI_CFG_DEV0_BR_READ_DATA     (uint8_t)(0x01)
                                                /* Device 0 Transfer rate for data reception.            */
```

DEV0 用の#SS 端子を割り付けるポート番号を以下のように設定します。

【 r_config/ r_flash_spi_pin_config.h 】

```
/***********************************************************************************************
PIN ASSIGNMENT
***********************************************************************************************/
/* The #defines specify the ports used for SS#. */
#define FLASH_SPI_CS_DEV0_CFG_PORTNO      '3'       /* Device 0 Port Number : FLASH SS#      */
#define FLASH_SPI_CS_DEV0_CFG_BITNO       '1'       /* Device 0 Bit Number    : FLASH SS#      */
```

(3)　簡易 I^2C モジュールの設定変更

チャネル 7 が有効になるよう以下のように設定します。

【 r_config/ r_sci_iic_rx_config.h 】

```
/* SPECIFY CHANNELS TO INCLUDE SOFTWARE SUPPORT FOR 1=included, 0=not */
                                    :
                                    :
    #define SCI_IIC_CFG_CH7_INCLUDED      (1)
```

5.2　SerialFlash 書き込み用のプロジェクトの設定

5.2.1　プロジェクトのプロパティ

プロジェクトのプロパティで下記の設定を行います。

・アドレス「0xFFE80000」に「IMAGE」セクションを追加

・” binary” オプションをコンパイルオプションの設定に追加（下記参照)

-binary="${ProjDirPath}/image/pict_data1.bmp"(IMAGE:4/DATA,_g_pict_data1)
-binary="${ProjDirPath}/image/pict_data2.bmp"(IMAGE:4/DATA,_g_pict_data2)
-binary="${ProjDirPath}/image/pict_data3.bmp"(IMAGE:4/DATA,_g_pict_data3)
-binary="${ProjDirPath}/image/pict_data4.bmp"(IMAGE:4/DATA,_g_pict_data4)
-binary="${ProjDirPath}/image/pict_cover.bmp"(IMAGE:4/DATA,_g_pict_cover)
-binary="${ProjDirPath}/image/pict_setting.bmp"(IMAGE:4/DATA,_g_pict_setting)
-binary="${ProjDirPath}/image/pict_correction.bmp"(IMAGE:4/DATA,_g_pict_correction)
-binary="${ProjDirPath}/image/pict_gamma.bmp"(IMAGE:4/DATA,_g_pict_gamma)

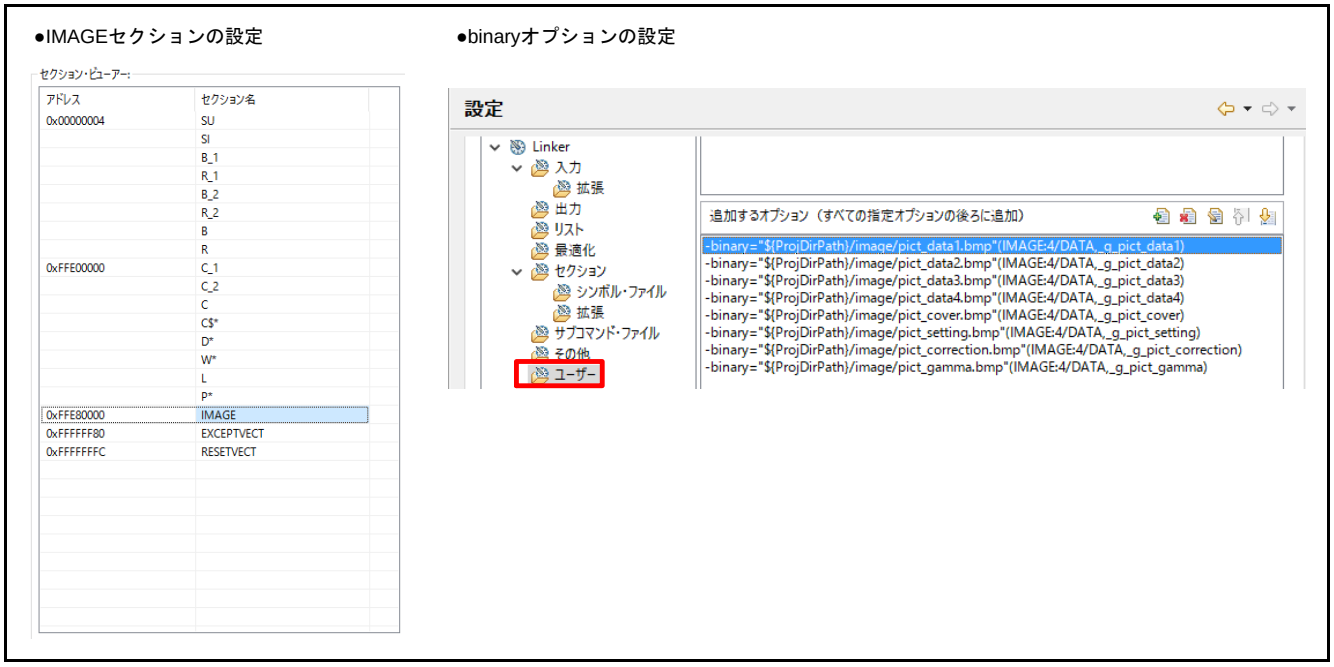


図 5.1　IMAGE セクションおよび binary オプションの設定

5.2.2　画像ファイルの配置

プロジェクトのディレクトリ直下に画像ファイルを配置します。フォルダ名は小文字で「image」です。

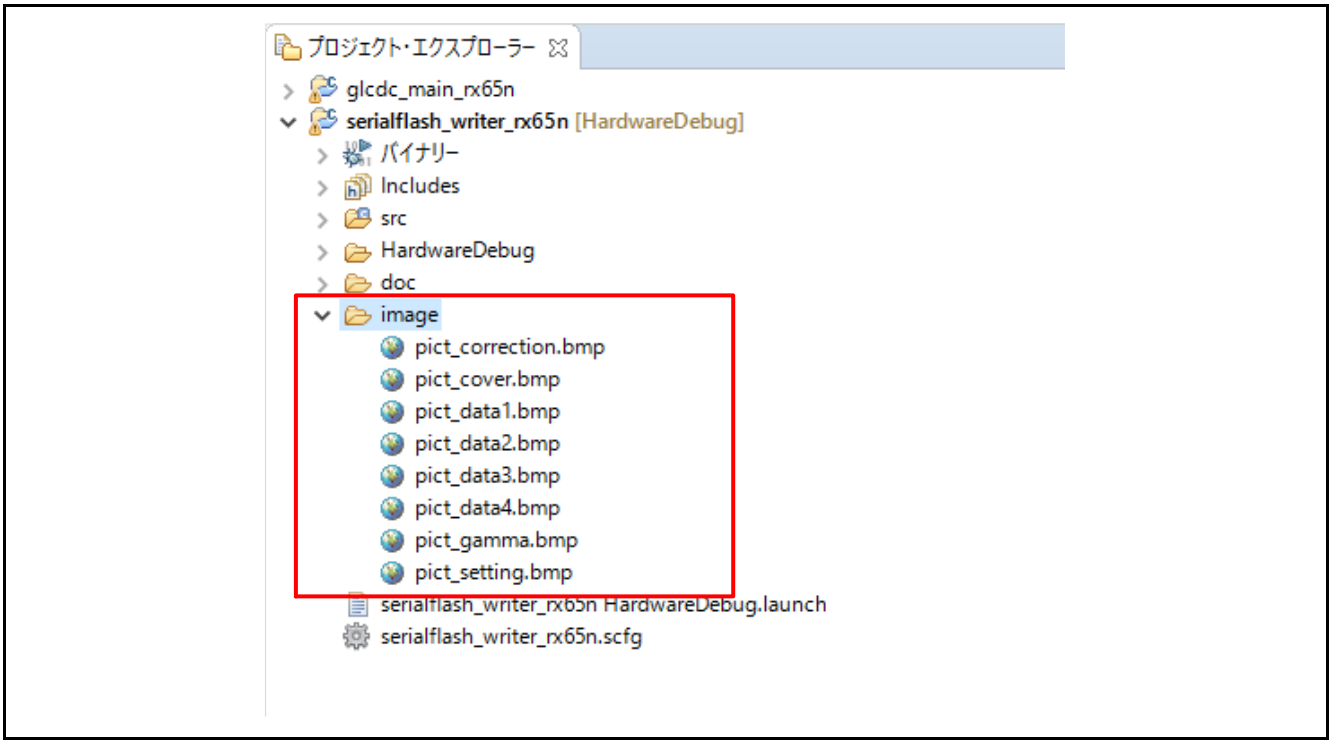


図 5.2　画像ファイルの配置

5.3　画像表示用のプロジェクトの設定

5.3.1　プロジェクトのプロパティ

プロジェクトのプロパティで下記の設定を行います。

・アドレス「0x00800000」に「RIMAGE」セクションを追加

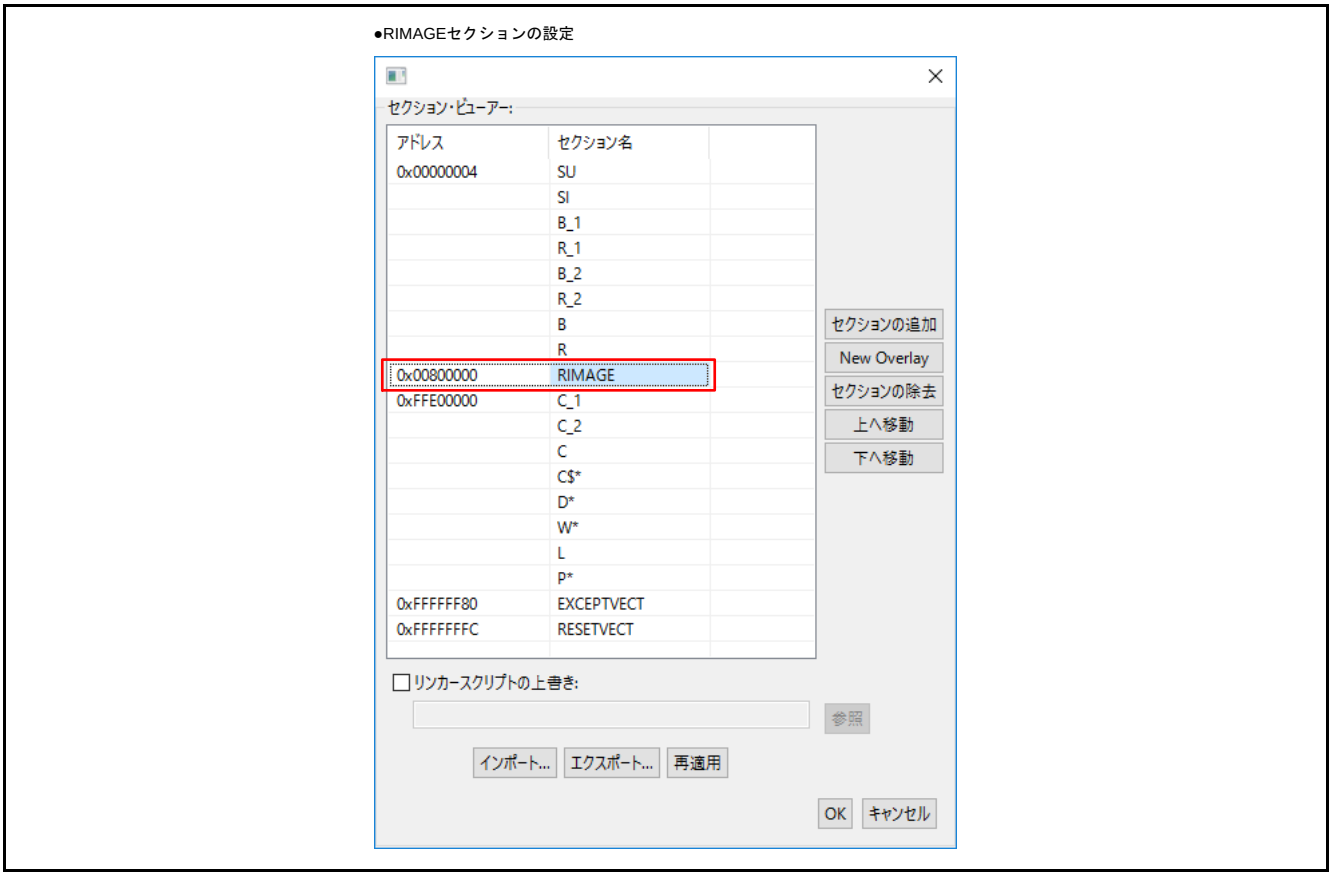


図 5.3　RIMAGE セクションの設定

5.4　CS+におけるビルドエラーの回避

5.4.1　不要フォルダの削除

Serial Flash memory アクセス クロック同期式制御モジュールは、同じ名前のファイルが別フォルダに存在します。CS+では、プロジェクト内に同じ名前のファイルが存在するとビルド時にエラーが発生します。

このため、本サンプルプログラムでは、図 5.4 に示す Serial Flash memory アクセス クロック同期式制御モジュールのフォルダを削除しています。

使用するフォルダ(削除しないフォルダ)：

・using_iodefine

・rx_fit_rspi

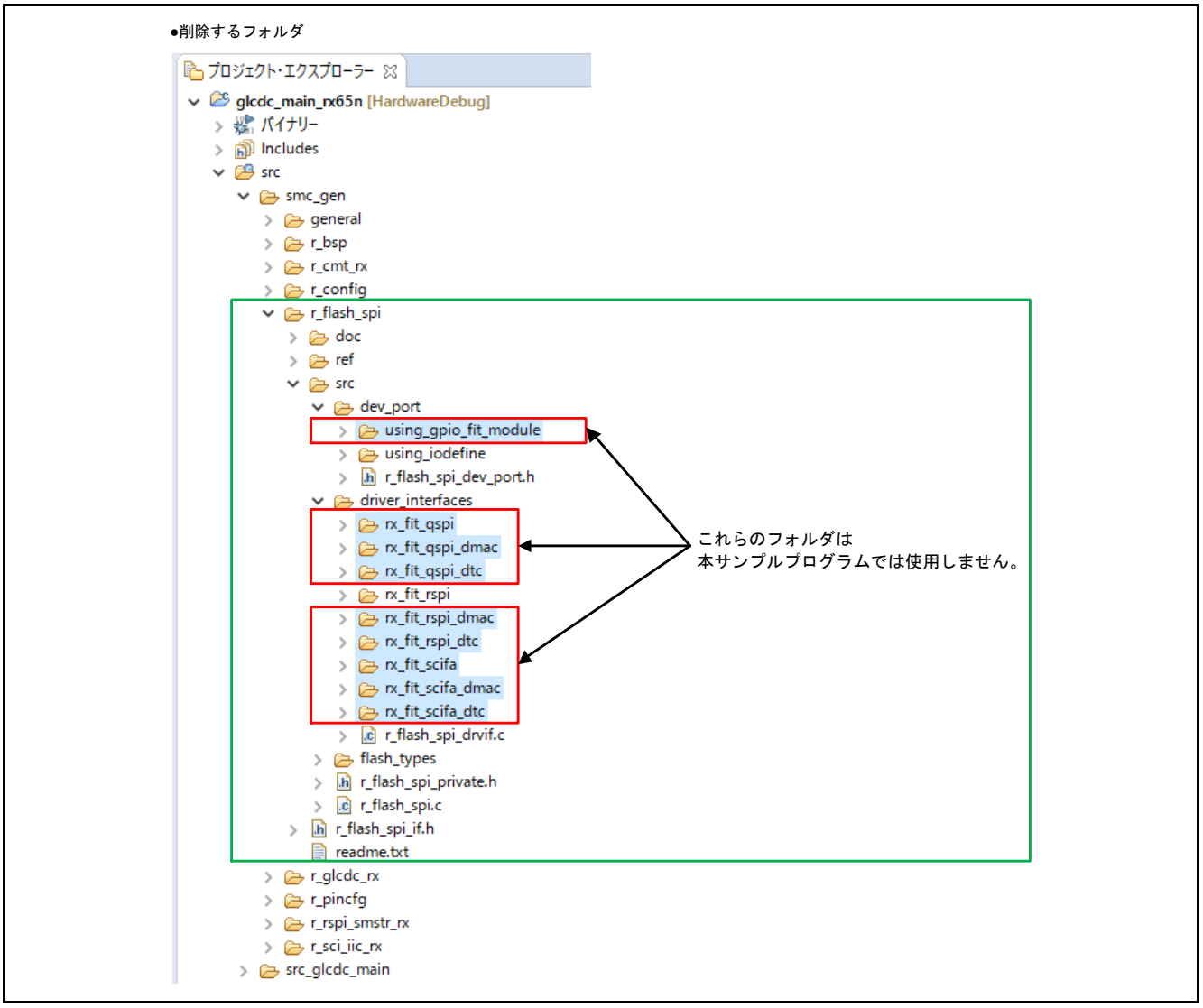


図 5.4　削除するフォルダ

6.　ハードウェア説明

6.1　　ハードウェア構成とジャンパ設定

本アプリケーションノートで使用するデバイスを表 6.1 使用デバイスとジャンパ設定に示します。

LCD パネルを使用するには、ジャンパの設定が必要です。サンプルプログラムを動作させる前に表 6.1 に示すジャンパを設定してください。

表 6.1　使用デバイスとジャンパ設定

デバイス	製品情報	ジャンパ設定
LCD パネル	メーカ: Newhaven Display 社製 型番:NHD-4.3-480272EF-ATXL#-CTP 画面サイズ:480×272 タッチコントローラ搭載	<SW4> Pin 3:OFF Pin 4:ON (OnBoard_TFT Available)
SerialFlash	メーカ:Micron Technology 社製 型番:MX25L3233FM2I-08G 容量:32Mbit (4Mbyte)	不要

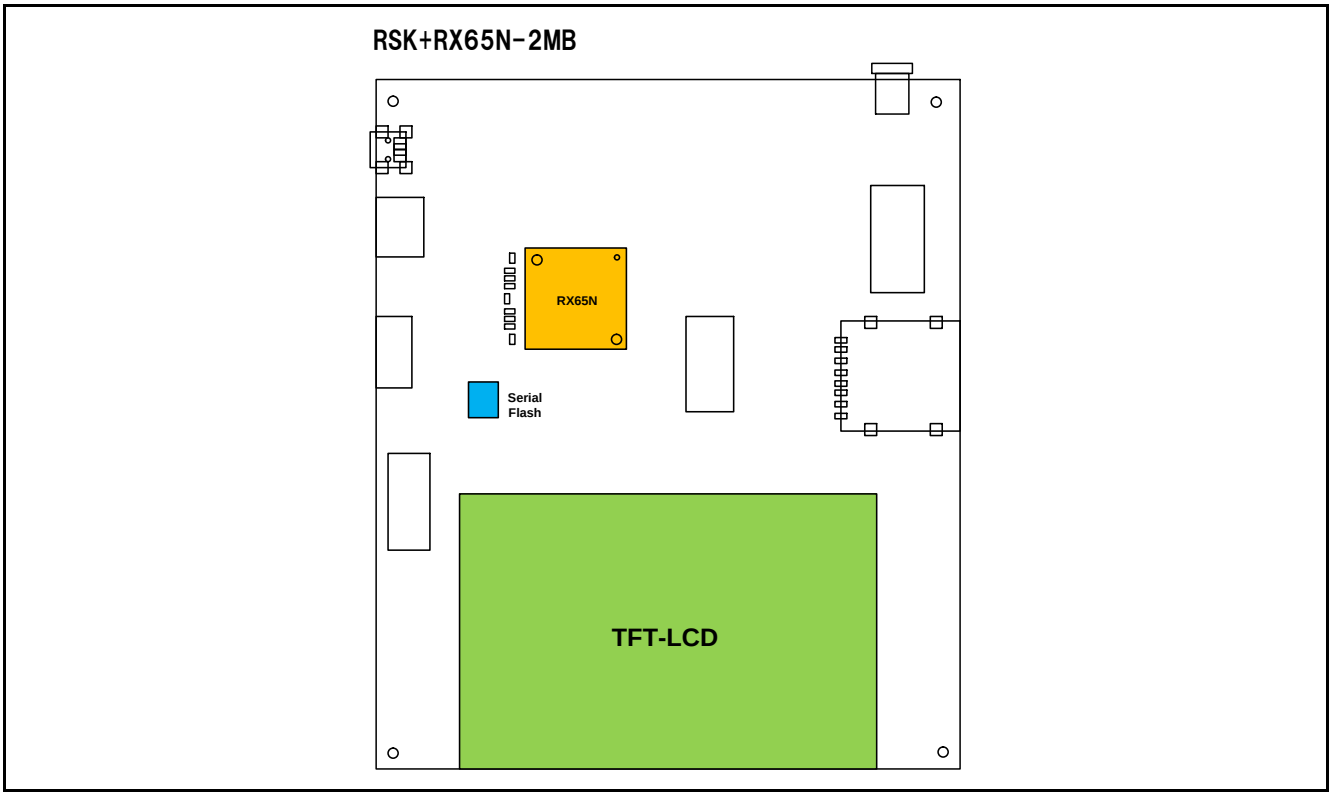


図 6.1　RSKRX65N-2MB

表 6.2 に使用端子と機能を示します。

使用端子は 176 ピンを想定しています。176 ピン版未満の製品を使用する場合は、使用する製品に合わせて端子を選択してください。

表 6.2　使用端子と機能

接続デバイス	端子名	入出力	内容
LCD パネル (GLCDC)	PB5/LCD_CLK-B	出力	パネルクロック出力
	PB4/LCD_TCON 0-B	出力	同期信号(VSYNC)出力
	PB2/LCD_TCON 2-B	出力	同期信号(HSYNC)出力
	PB1/LCD_TCON 3-B	出力	同期信号(DE)出力
	PB0/LCD_DATA 0-B	出力	LCD 信号出力　R[3]
	PA7/LCD_DATA 1-B	出力	LCD 信号出力　R[4]
	PA6/LCD_DATA 2-B	出力	LCD 信号出力　R[5]
	PA5/LCD_DATA 3-B	出力	LCD 信号出力　R[6]
	PA4/LCD_DATA 4-B	出力	LCD 信号出力　R[7]
	PA3/LCD_DATA 5-B	出力	LCD 信号出力　G[2]
	PA2/LCD_DATA 6-B	出力	LCD 信号出力　G[3]
	PA1/LCD_DATA 7-B	出力	LCD 信号出力　G[4]
	PA0/LCD_DATA 8-B	出力	LCD 信号出力　G[5]
	PE7/LCD_DATA 9-B	出力	LCD 信号出力　G[6]
	PE6/LCD_DATA 10-B	出力	LCD 信号出力　G[7]
	PE5/LCD_DATA 11-B	出力	LCD 信号出力　B[3]
	PE4/LCD_DATA 12-B	出力	LCD 信号出力　B[4]
	PE3/LCD_DATA 13-B	出力	LCD 信号出力　B[5]
	PE2/LCD_DATA 14-B	出力	LCD 信号出力　B[6]
	PE1/LCD_DATA 15-B	出力	LCD 信号出力　B[7]
	PB7	出力	バックライト
	P97	出力	パネル ON
LCD パネル タッチコントローラ(簡易 I^2C)	P92/SSCL7	入出力	クロック入出力
	P90/SSDA7	入出力	データ入出力
	P42	入力	トリガ入力
SerialFlash (RSPI)	P31	出力	スレーブセレクト
	P30/MISOB-A	入出力	スレーブ送出データ入出力
	P27/RSPCKB-A	入出力	クロック入出力
	P26/MOSIB-A	入出力	マスタ送出データ入出力
スイッチ	P03	入力	SW1
	P05	入力	SW2
	P07	入力	SW3

7.　ソフトウェア説明 (SerialFlash 書き込み用のプロジェクト)

7.1　　動作概要

7.1.1　　使用する周辺機能やデバイスの設定

SerialFlash への書き込みに使用する RSPI の設定を表 7.1 に示します。RSPI の詳細な設定については、RSPI クロック同期式シングルマスタ制御モジュールのマニュアルを参照してください。

表 7.1　RSPI の設定

項目	設定内容	備考
使用チャネル	チャネル 1	Serial Flash との通信に使用 書き込みのみ
通信方式	クロック同期式モード	
転送速度(ビットレート)	30Mbps	

7.1.2　　SerialFlash 書き込み処理

本サンプルプログラムでは、SerialFlash に画像データを書き込むために、RX65N の ROM に画像データを配置します。表 7.2 に RX65N と SerialFlash の画像データのアドレスマップを示します。合計サイズは 1,072,304 バイトとなります。ROM への配置方法は、「5.2 SerialFlash 書き込み用のプロジェクトの設定」を参照してください。

表 7.2　RX65N と SerialFlash の画像データのアドレスマップ(SerialFlash 書き込み用プロジェクト)

デバイス	内容	開始アドレス
RX65N (セクション：IMAGE)	pict_data1.bmp	0xFFE80000
	pict_data2.bmp	0xFFE9BEF8
	pict_data3.bmp	0xFFEB7DF0
	pict_data4.bmp	0xFFED3CE8
	pict_cover.bmp	0xFFEEFBE0
	pict_setting.bmp	0xFFF0BAD8
	pict_correction.bmp	0xFFF279D0
	pict_gamma.bmp	0xFFF438C8
	終端アドレス	0xFFF5F7BD
SerialFlash (書き込み先)	pict_data1.bmp	0x00000000
	pict_data2.bmp	0x00025800
	pict_data3.bmp	0x0004B000
	pict_data4.bmp	0x00070800
	pict_cover.bmp	0x00096000
	pict_setting.bmp	0x000BB800
	pict_correction.bmp	0x000E1000
	pict_gamma.bmp	0x00106800

7.2　ファイル構成

表 7.3 にサンプルプログラムで使用するファイルを示します。なお、FIT モジュールや統合開発環境で自動生成されるファイルは除きます。

表 7.3　サンプルプログラムで使用するファイル

ファイル名	概要	備考
main.c	メイン処理	
r_serial_flash_write.c	RSPI の初期化、SerialFlash へのデータの書き込み処理	
r_serial_flash_write.h	r_serial_flash_write.c のヘッダファイル	

7.3　オプション設定メモリ

表 7.4 にサンプルプログラムで使用するオプション設定メモリの状態を示します。必要に応じて、お客様のシステムに最適な値を設定してください。

表 7.4　サンプル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
OFS0	FE7F 5D04h～FE7F 5D07h	FFFF FFFFh	リセット後、WDT、IWDT は停止
OFS1	FE7F 5D08h～FE7F 5D0Bh	FFFF FFFFh	リセット後、電圧監視 0 リセット無効 リセット後、HOCO 発振が無効
MDE	FE7F 5D00h～FE7F 5D03h	FFFF FFFFh	リトルエンディアン

7.4　定数一覧

表 7.5 にサンプルプログラムで使用する定数を示します。

表 7.5　サンプルプログラムで使用する定数

定数名	設定値	内容
LED_ON	(0)	LED の点灯
LED_OFF	(1)	LED の消灯
LED0	(PORT7.PODR.BIT.B3)	LED0(P73)の PODR レジスタのビット
LED3	(PORTG.PODR.BIT.B5)	LED3(PG5)の PODR レジスタのビット
LED0_PDR	(PORT7.PDR.BIT.B3)	LED0(P73)の PDR レジスタのビット
LED3_PDR	(PORTG.PDR.BIT.B5)	LED3(PG5)の PDR レジスタのビット
IMAGE_NUM	(8)	Serial Flash に書き込む画像の数
BMP_SIZE	(114422)	ビットマップファイルのサイズ (バイト)

7.5　変数一覧

表 7.6 にグローバル変数を示します。

表 7.6　グローバル変数

型	変数名	内容
extern uint8_t	g_pict_data1[]	画像 1 のポインタ
extern uint8_t	g_pict_data2[]	画像 2 のポインタ
extern uint8_t	g_pict_data3[]	画像 3 のポインタ
extern uint8_t	g_pict_data4[]	画像 4 のポインタ
extern uint8_t	g_pict_cover[]	重ね合わせ画像 (切り替えボタン) のポインタ
extern uint8_t	g_pict_setting[]	重ね合わせ画像 (設定選択画面) のポインタ
extern uint8_t	g_pict_correction[]	重ね合わせ画像 (輝度・コントラスト設定画面) のポインタ
extern uint8_t	g_pict_gamma[]	重ね合わせ画像 (ガンマ設定画面) のポインタ
uint8_t *	gp_pict_table[IMAGE_NUM]	各画像へのポインタテーブル
uint32_t	g_pict_addr_table[IMAGE_NUM]	SerialFlash の画像配置アドレステーブル

7.6　関数一覧

表 7.7、表 7.8 に関数を示します。

表 7.7　関数(main.c)

関数名	概要
main	メイン処理

表 7.8　関数(r_serial_flash_write.c)

関数名	概要
serialflash_write_initialize	SerialFlash 通信の初期化
data_write	SerialFlash へのデータ書き込み処理

7.7　関数仕様

サンプルプログラムの関数仕様を示します。

main	
概　要	メイン処理
ヘッダ	なし
宣　言	void main(void)
説　明	SeialFlash に画像データを書き込みます。
引　数	なし
リターン値	なし

serialflash_write_initialize	
概　要	SerialFlash 通信の初期化
ヘッダ	r_serial_flash_write.h
宣　言	flash_spi_status_t serialflash_write_initialize(void)
説　明	SerialFlash との通信の初期化を行います。また、書き込みする準備として SerialFlash のデータをすべて削除します。
引　数	なし
リターン値	FLASH_SPI_SUCCESS　：正常終了 FLASH_SPI_ERR_HARD　：ハードウェアエラー FLASH_SPI_ERR_OTHER　：その他のエラー

data_write	
概　要	SerialFlash へのデータ書き込み処理
ヘッダ	r_serial_flash_write.h
宣　言	flash_spi_status_t data_write(uint8_t *p_src_addr,　uint32_t dest_addr, uint32_t data_size)
説　明	指定の画像データを SerialFlash の指定アドレスに書き込みます。
引　数	uint8_t *p_src_addr　書き込む画像データのアドレス uint32_t dest_addr　SerialFlash の配置先アドレス uint32_t data_size　書き込む画像データのサイズ
リターン値	FLASH_SPI_SUCCESS　：正常終了 FLASH_SPI_ERR_PARAM　：パラメータエラー FLASH_SPI_ERR_HARD　：ハードウェアエラー FLASH_SPI_ERR_OTHER　：その他のエラー

7.8　フローチャート

7.8.1　メイン処理

図 7.1 にメイン処理のフローチャートを示します。

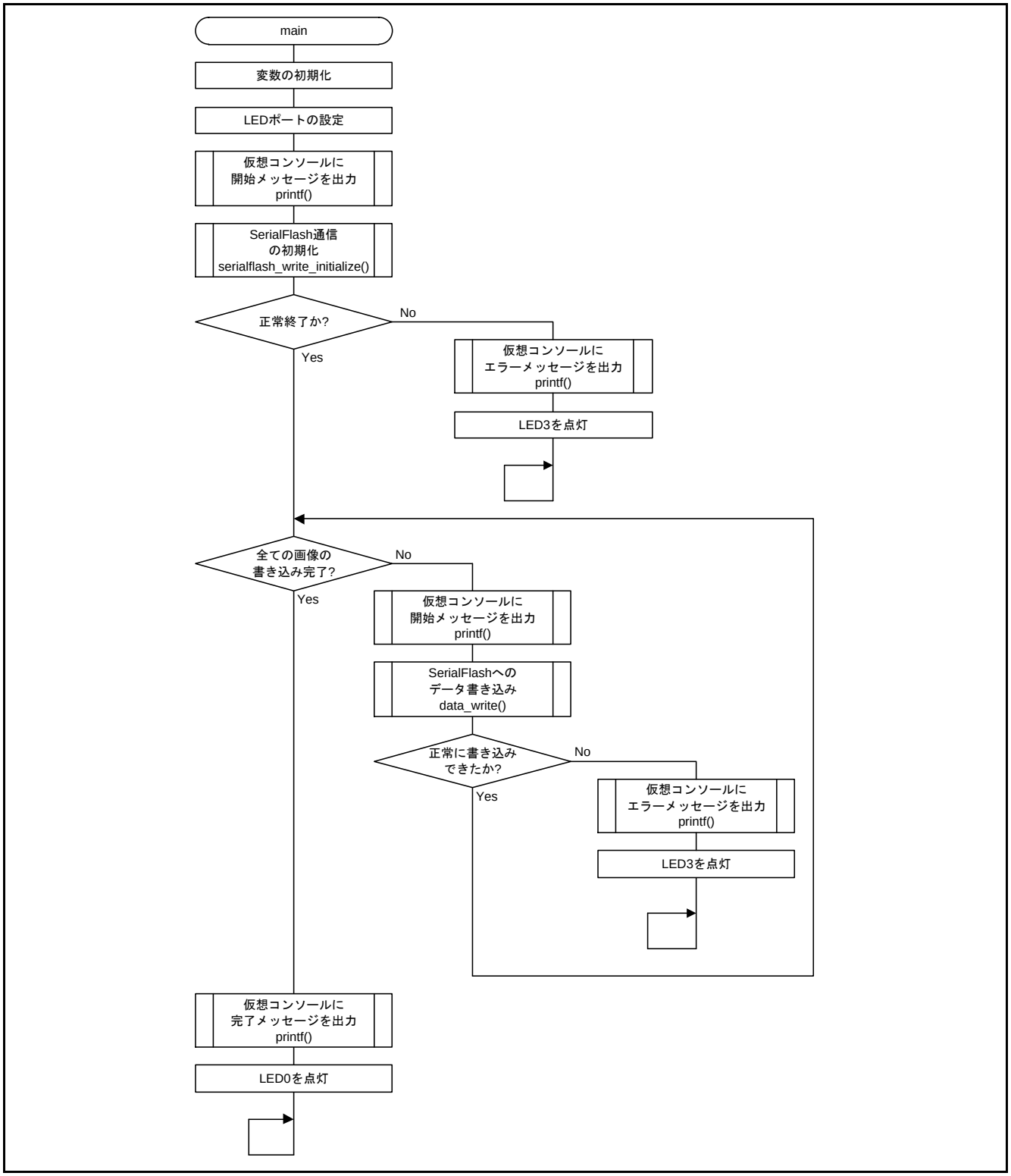


図 7.1　メイン処理

7.8.2　SerialFlash 通信の初期化

図 7.2 に SerialFlash 通信の初期化のフローチャートを示します。

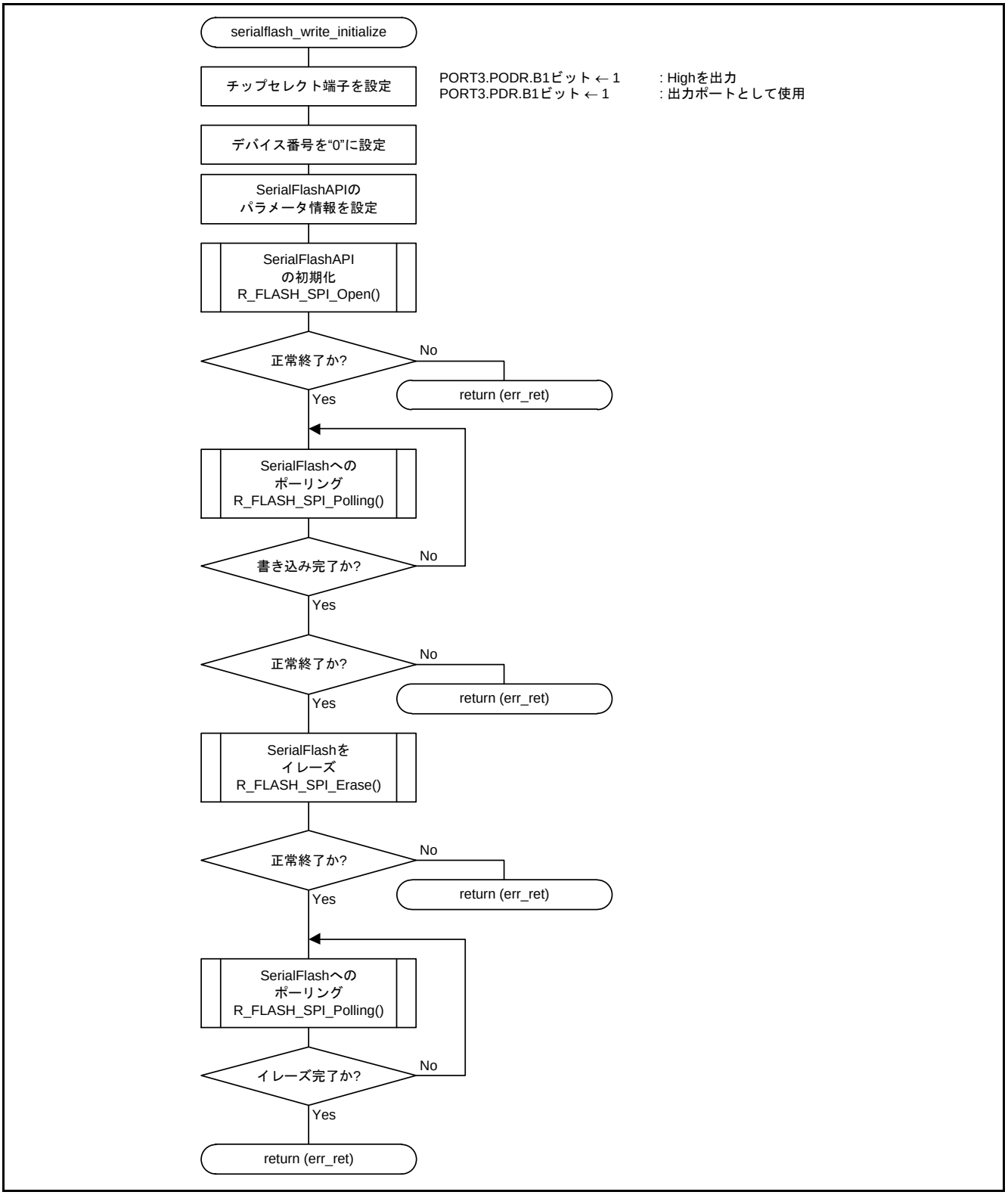


図 7.2　SerialFlash 通信の初期化

7.8.3　SerialFlash へのデータ書き込み処理

図 7.3 に SerialFlash へのデータ書き込み処理のフローチャートを示します。

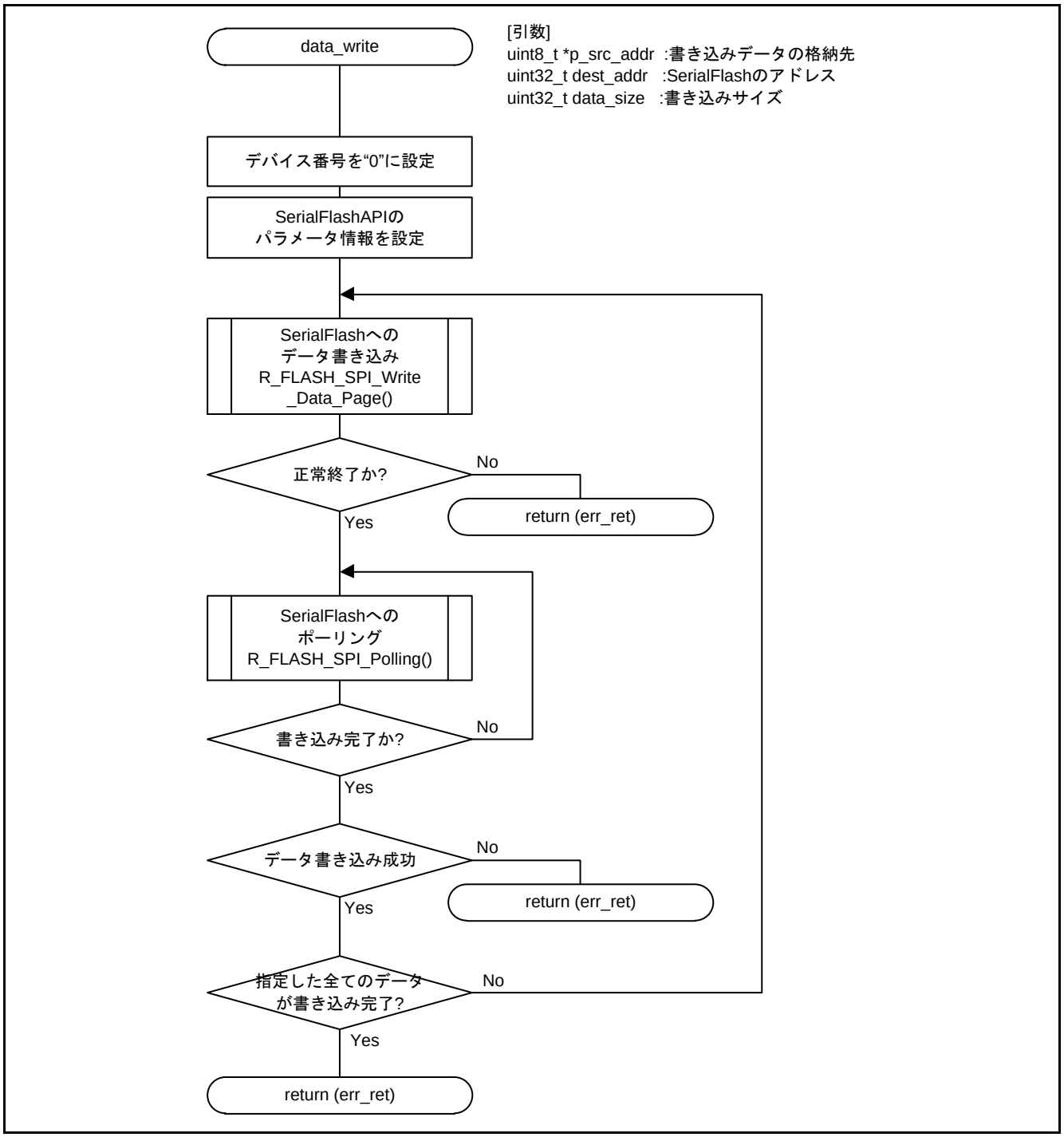


図 7.3　SerialFlash へのデータ書き込み処理

8. ソフトウェア説明 (画像表示用プロジェクト)

8.1 動作概要

本章では、画像表示用のサンプルプログラムの処理内容を説明します。サンプルプログラムは主に、画像表示処理、タッチ検出・判定処理、設定変更処理、SerialFlash 読み出し処理に分かれます。

8.1.1 使用する周辺機能やデバイスの設定

GLCDC の設定を表 8.1 に、CMT の設定を表 8.2 に、SCI の設定を表 8.3 に、RSPI の設定を表 8.4 に示します。各周辺機能の詳細については、各 FIT モジュールのマニュアルを参照してください。

表 8.1　GLCDC の設定

項目		設定内容	備考
グラフィック	グラフィック画面	グラフィック 1:重ね合わせ画像 グラフィック 2 表示画像 バックグラウンド画面:灰色表示	動作に応じて設定を変更
	カラーフォーマット	CLUT(8)プログレッシブフォーマット (CLUT インデックス:8 ビット(256 エントリ))	画像データに合わせて設定
	CLUT メモリ (以後、カラーパレット)	画像のカラーパレットを設定	画像データのカラーパレットを抽出
	アルファブレンド	[ブレンド表示モード時] 矩形領域アルファブレンド (矩形範囲は画像データに合わせて 448px × 253px を指定し、アルファ値は設定モードで設定した値を使用) [画像表示モード時] ピクセル単位アルファブレンド (画像が持つアルファ値を使用)	本サンプルプログラムの動作モードに合わせて使い分け
画像フォーマット		出力サイズ　：480px × 272px 水平フロントポーチ　：3 ピクセル 水平バックポーチ　：2 ピクセル 水平同期信号パルス幅　：41 ピクセル 垂直フロントポーチ　：2 ライン 垂直バックポーチ　：2 ライン 垂直同期信号パルス幅　：10 ライン	LCD パネルの仕様に合わせて設定
データフォーマット変換	出力データフォーマット	RGB(565)（パラレル 16 ビット）	LCD パネルの仕様に合わせて設定
	ディザ処理	2×2 パターンディザ	画像の粗さを低減
	同期信号出力	TCON0:VSYNC (極性反転あり) TCON1:未使用 TCON2:HSYNC (極性反転あり) TCON3:DE　(極性反転なし)	LCD パネルの仕様に合わせて設定
パネル用補正処理	輝度/コントラスト	輝度：-512 ～ +512 から 5 段階で設定 コントラスト：0 倍～2 倍 から 5 段階で設定	動作に応じて設定を変更
	ガンマ補正	ガンマ値に応じたテーブル(5 個)を 5 段階で設定	動作に応じて設定を変更

項目	設定内容	備考
割り込み	指定ライン検出割り込み グラフィック 1 アンダフロー割り込み グラフィック 2 アンダフロー割り込み	アンダフロー検出割り込みは画像データの読み出しが間に合わない(バス負荷が高い)場合に発生

表 8.2　CMT の設定

項目	設定内容	備考
使用チャネル	チャネル 0	メイン処理の実行周期タイマとして使用
カウントクロック	PCLKB の 128 分周	PCLKB = 60MHz
コンペアマッチ周期	100ms (23,912 カウント)	

表 8.3　SCI の設定

項目	設定内容	備考
使用チャネル	チャネル 7	LCD パネルのタッチコントローラとの通信に使用
通信方式	簡易 I^2C モード	
転送速度(ビットレート)	384kbps	

表 8.4　RSPI の設定

項目	設定内容	備考
使用チャネル	チャネル 1	Serial Flash との通信に使用 読み出しのみ
通信方式	クロック同期式モード	
転送速度(ビットレート)	30Mbps	

8.1.2　　画像表示処理

画像の表示は GLCDC で行います。必要なレジスタへの設定後、動作を許可することで、メモリ領域の指定アドレスから画像データを読み出し、必要な処理を施した後、LCD パネルに出力します。GLCDC は、動作している間この指定アドレスから繰り返し画像データを読み出しており、LCD パネルへ波形を出力し続けています。

本アプリケーションノートでは、GLCDC が画像データを読み出すアドレスは固定にし、画像データをそのアドレスへ直接上書きすることで、LCD パネルに表示される画像を変更しています。

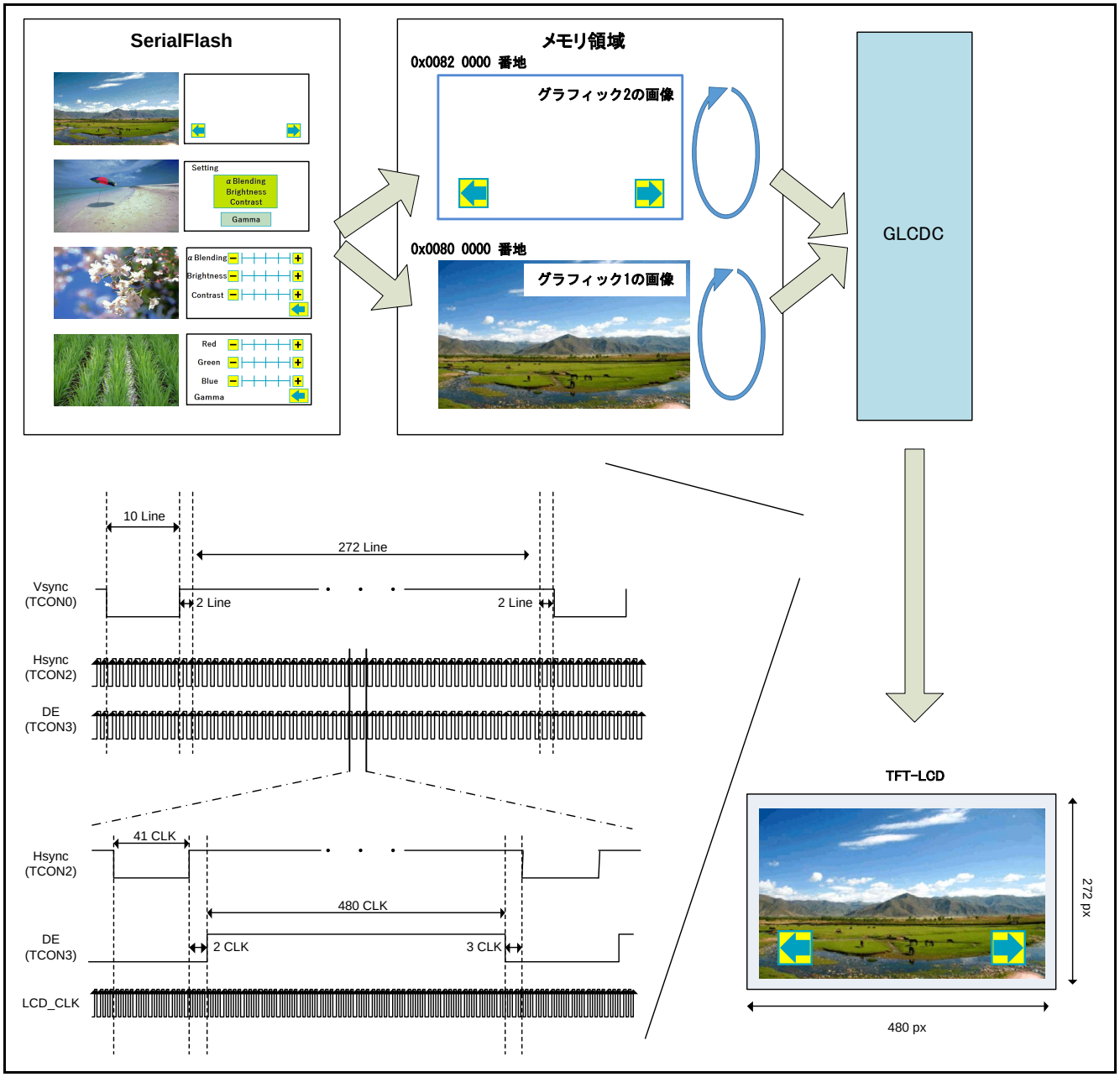


図 8.1　LCD パネルへの出力動作

8.1.3　　タッチ検出・判定処理

LCD パネルに搭載されているタッチコントローラとの通信は I^2C プロトコルで行います。また、タッチコントローラとの通信周期は、CMT0 を使用して 100ms ごとに行います。100ms ごとにタッチコントローラから情報を読み出し、LCD パネル上のボタンがタッチされたかどうかを判定します。

以下に、タッチ検出および判定処理の内容を示します。

1.　タッチ検出処理 (タッチコントローラからの情報取得)

タッチコントローラには、タッチ検出の状態を格納するステータスレジスタがあり、マスタからアドレスを送信することで、対応するステータスレジスタの値を取得します。本サンプルプログラムでは以下のレジスタを読み出し、タッチされたかどうかを判定します。

LCD パネルのタッチコントローラの詳細な仕様については、LCD パネルのデータシートを参照してください。

表 8.5　参照するタッチコントローラのレジスタ

レジスタ名	アドレス [ビット]	内容
Touch Points レジスタ	02h　[3:0 bits]	000b :未タッチ 001b～101b :タッチ検出 (5 個までのポイント検出可)
Touch 1 Event Flag レジスタ	03h　[7:6 bits]	00b:押下 01b:放す 10b:接触状態
TOUCH1_XH レジスタ	03h　[3:0 bits]	X 座標の上位 4 ビット
TOUCH1_XL レジスタ	04h　[7:0 bits]	X 座標の下位 8 ビット
TOUCH1_YH レジスタ	05h　[3:0 bits]	Y 座標の上位 4 ビット
TOUCH1_YL レジスタ	06h　[7:0 bits]	Y 座標の下位 8 ビット

2.　判定処理

タッチコントローラから取得した情報をもとに、LCD パネル上のボタンが押されたかどうかを判定します。

表 8.5 より、下記の条件でタッチされたと判定します。

・Touch Points レジスタの値 > 000b

・Touch 1 Event Flag レジスタの値 == 10b

その後、取得した座標が図 8.2 で示す範囲内であるときに、各ボタンに応じたフラグ情報を設定します。

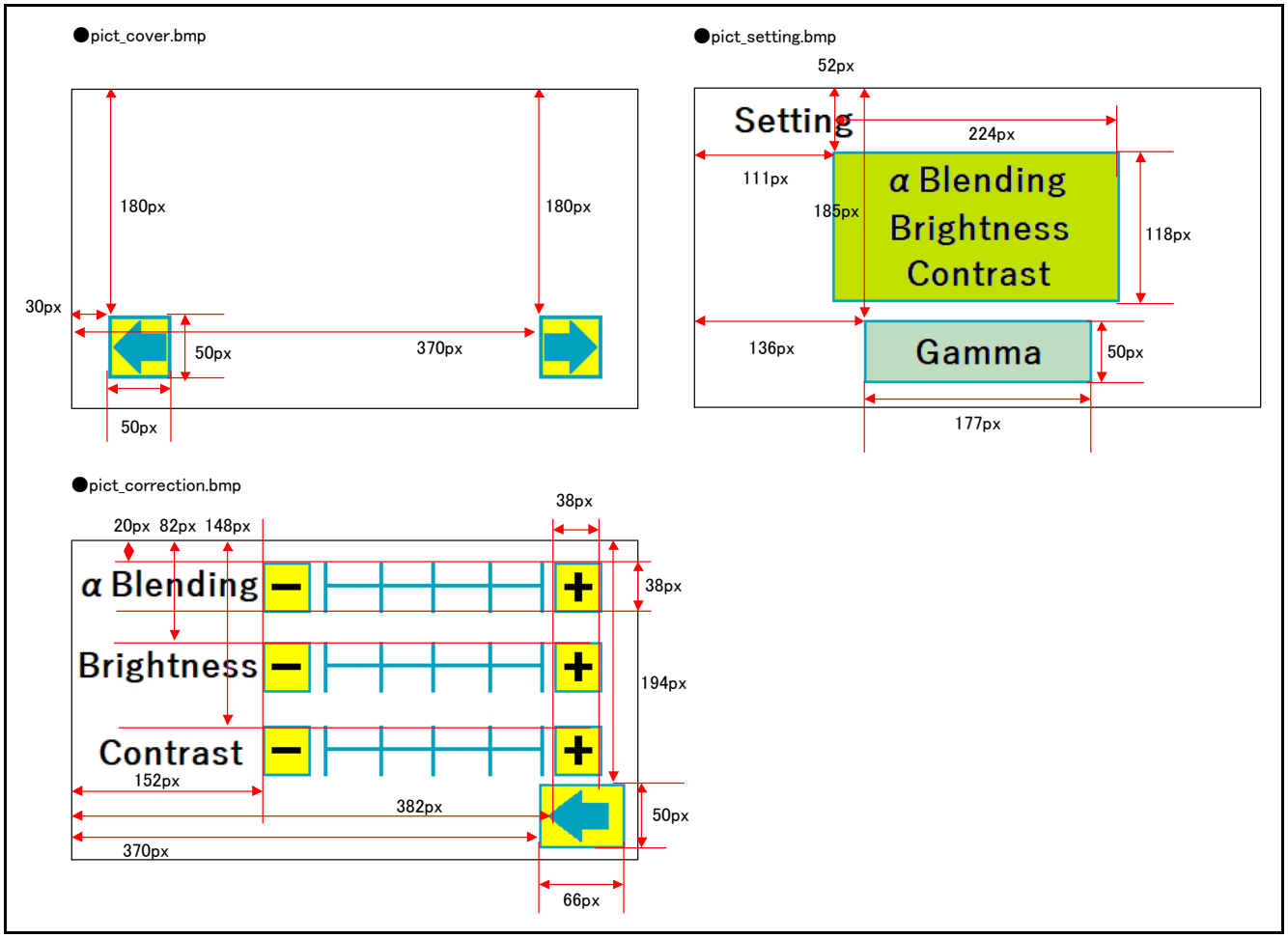


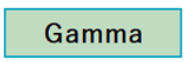
図 8.2　ボタンの座標

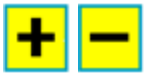
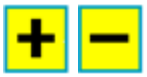
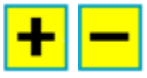
8.1.4　　設定変更処理

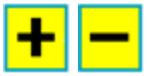
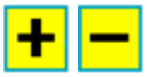
タッチ検出・判定処理の結果およびスイッチ(SW1～SW3)をもとに、表示モードや表示画像の変更、輝度/コントラスト/ガンマ値などの情報を更新します。

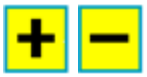
各ボタンにより実行する処理内容の一覧を表 8.6 に示します。

表 8.6　ボタン操作における処理内容一覧

画面モード	ボタン or スイッチ	変更対象	処理内容
画像表示モード	←矢印 	グラフィック 1 の画像	前の画像を SerialFlash から取得し、表示中の画像データに上書きします。 <表示例 (数字は画像番号)> 1 → 4 → 3 → 2 → 1 → 4 ・・・
	→矢印 		次の画像を SerialFlash から取得し、表示中の画像データに上書きします。 <表示例 (数字は画像番号)> 1 → 2 → 3 → 4 → 1 ・・・
ブレンド表示モード	画面全体	グラフィック 1 とグラフィック 2 の画像	次の画像を SerialFlash から取得し、表示中の画像データに上書きします。 <表示例 (数字は画像番号)> 1 , 2 → 2 , 3 → 3 , 4 → 4 , 1 → 1, 2 ・・・
設定モード	αBlending Brightness Contrast 	グラフィック 2 の画像	アルファ値、輝度、コントラストの設定画像(pict_correction.bmp)を SerialFlash から取得して、表示中の画像データに上書きします。 また、設定画面の横バーに設定値を反映する処理を実行します。(縦線との交点に■を表示)
	Gamma 	グラフィック 2 の画像	ガンマの設定画像(pict_gamma.bmp)を SerialFlash から取得して、表示中の画像データに上書きします。 また、設定画面の横バーに設定値を反映する処理を実行します。(縦線との交点に■を表示)

画面モード	ボタン or スイッチ	変更対象	処理内容
設定モード (アルファ値、輝度、コントラスト)	アルファブレンド値の+,- 	グラフィック 2 のアルファ値 <対象レジスタ> GR2AB7 レジスタ ・ARCDEF ビット	グラフィック 2 のアルファ値(GR2AB7 レジスタの ARCDEF ビットに設定する値)を 5 段階で変更します。(設定範囲: 0 ～ 255) 5: 255 (上層の画像が表示) 4: 191 3: 128 (サンプルプログラム実行時の初期値) 2: 64 1: 0　　(下層の画像が表示) また、設定画面の横バーに設定値を反映する処理を実行します。(縦線との交点に■を表示)
	輝度の+,- 	輝度値 <対象レジスタ> BRIGHT1 レジスタ ・BRTG ビット BRIGHT2 レジスタ ・BRTR ビット ・BRTB ビット	輝度値(BRIGHT1 レジスタの BRTG ビット、および BRIGHT2 レジスタの BRTR ビットと BRTB ビットに設定する値)を 5 段階で変更します。(設定範囲: 0 ～ 1023) 5: 1023 (明るい) 4: 767 3: 512　(サンプルプログラム実行時の初期値) 2: 256 1: 0　　(暗い) ※RGB の輝度値は同値とする。 また、設定画面の横バーに設定値を反映する処理を実行します。(縦線との交点に■を表示)
	コントラストの+,- 	コントラスト値 <対象レジスタ> CONTRAST レジスタ ・CONTR ビット ・CONTB ビット ・CONTG ビット	コントラスト値(CONTRAST レジスタの CONTR ビット、CONTB ビット、CONTG ビットに設定する値)を 5 段階で変更します。(設定範囲: 0 ～ 255) 5: 255 (薄い) 4: 191 3: 128 (サンプルプログラム実行時の初期値) 2: 64 1: 0　　(濃い) ※RGB のコントラスト値は同値です。 また、設定画面の横バーに設定値を反映する処理を実行します。(縦線との交点に■を表示)

画面モード	ボタン or スイッチ	変更対象	処理内容
	バックボタン 	グラフィック 2 の画像	設定モードの画像(pict_setting.bmp)を SerialFlash から読み出し、表示中の画像と差し替えます。
設定モード (ガンマ値)	R 値の+,- 	R のガンマ値 <対象レジスタ> GAMRLUT1 ～ GAMRLUT8 レジスタ GAMRAREA1 ～ GAMRAREA4 レジスタ	R のガンマ値(GAMRLUT1～GAMRLUT8 レジスタ、GAMRAREA1～GAMRAREA4 レジスタ)を変更します。各設定値については γ 値に応じた 5 つのテーブルデータを切り替えます。 5: テーブル 5 → γ = 1.30 4: テーブル 4 → γ = 1.10 3: テーブル 3 → γ = 0.90 (サンプルプログラム実行時の初期値) 2: テーブル 2 → γ = 0.70 1: テーブル 1 → γ = 0.50 また、設定画面の横バーに設定値を反映する処理を実行します。 (縦線との交点に■を表示)
	G 値の+,- 	G のガンマ値 <対象レジスタ> GAMGLUT1 ～ GAMGLUT8 レジスタ GAMGAREA1 ～ GAMGAREA4 レジスタ	G のガンマ値(GAMGLUT1～GAMGLUT8 レジスタ、GAMGAREA1～GAMGAREA4 レジスタ)を変更します。各設定値については γ 値に応じた 5 つのテーブルデータを切り替えます。 5: テーブル 5 → γ = 1.30 4: テーブル 4 → γ = 1.10 3: テーブル 3 → γ = 0.90 (サンプルプログラム実行時の初期値) 2: テーブル 2 → γ = 0.70 1: テーブル 1 → γ = 0.50 また、設定画面の横バーに設定値を反映する処理を実行します。 (縦線との交点に■を表示)

画面モード	ボタン or スイッチ	変更対象	処理内容
	B 値の+,- 	B のガンマ値 <対象レジスタ> GAMBLUT1 ～ GAMBLUT8 レジスタ GAMBAREA1 ～ GAMBAREA4 レジスタ	B のガンマ値(GAMBLUT1～GAMBLUT8 レジスタ、GAMBAREA1～GAMBAREA4 レジスタ)を変更します。各設定値については γ 値に応じた 5 つのテーブルデータを切り替えます。 5: テーブル 5 → γ = 1.30 4: テーブル 4 → γ = 1.10 3: テーブル 3 → γ = 0.90 (サンプルプログラム実行時の初期値) 2: テーブル 2 → γ = 0.70 1: テーブル 1 → γ = 0.50 また、設定画面の横バーに設定値を反映する処理を実行します。 (縦線との交点に■を表示)
設定モード (ガンマ値)	バックボタン 	グラフィック 2 の画像	設定モードの画像(pict_setting.bmp)を SerialFlash から読み出し、表示中の画像と差し替えます。
全ての画面モード	SW1	グラフィック 1 およびグラフィック 2 の画像 グラフィック 2 のアルファブレンド機能とアルファブレンド値 <対象レジスタ> GR2AB1 レジスタ ・ARCON ビット	現在のモードの画像情報を保存してから、画像表示モードへの切り替えを行います。 保存情報をもとにグラフィック 1、グラフィック 2 の画像を SerialFlash から読み出し、データを差し替えます。 さらに、アルファブレンドの設定を「ピクセル単位アルファブレンド」に変更します。
	SW2	グラフィック 1 およびグラフィック 2 の画像 グラフィック 2 のアルファブレンド機能とアルファブレンド値 <対象レジスタ> GR2AB1 レジスタ ・ARCON ビット GR2AB7 レジスタ ・ARCDEF ビット	現在のモードの画像情報を保存してから、ブレンド表示モードへの切り替えを行います。 保存情報をもとにグラフィック 1、グラフィック 2 の画像を SerialFlash から読み出し、データを差し替えます。 さらに、アルファブレンドの設定を「矩形領域アルファブレンド」に変更し、アルファブレンド値をブレンド表示モード用の値に変更します。

画面モード	ボタン or スイッチ	変更対象	処理内容
	SW3	グラフィック 1 の画像 グラフィック 2 のアルファブレンド機能とアルファブレンド値 <対象レジスタ> GR2AB1 レジスタ ・ARCON ビット	現在のモードの画像情報を保存してから、設定モードへの切り替えを行います。 設定モードの画像(pict_setting.bmp)を SerialFlash から読み出し、表示中の画像と差し替えます。 さらに、アルファブレンドの設定を「ピクセル単位アルファブレンド」に変更します。

8.1.5　SerialFlash 読み出し処理

画像データは SerialFlash に格納しており、必要に応じて都度読み出します。SerialFlash に格納されているデータは BMP (Windows Bitmap Image)そのままのため、カラーパレットと画像データを分けて読み出します。

表 8.7 に RX65N と SerialFlash の画像データのアドレスマップを示します。使用する画像については、1.2 使用する画像データを参照してください。

表 8.7　RX65N と SerialFlash の画像データのアドレスマップ(画像表示用プロジェクト)

デバイス	内容		開始アドレス
RX65N (セクション：RIMAGE)	グラフィック 1 用のデータ配置アドレス		0x00800000
	グラフィック 2 用のデータ配置アドレス		0x00820000
	バッファ用のデータ配置アドレス (ビッグエンディアン時のみ使用)		0x00840000
SerialFlash (読み出し元)	pict_data1.bmp	ヘッダ(54byte)	0x00000000
		カラーパレット(1024byte)	0x00000036
		画像データ(113,344byte)	0x00000436
	pict_data2.bmp	ヘッダ(54byte)	0x00025800
		カラーパレット(1024byte)	0x00025836
		画像データ(113,344byte)	0x00025C36
	pict_data3.bmp	ヘッダ(54byte)	0x0004B000
		カラーパレット(1024byte)	0x0004B036
		画像データ(113,344byte)	0x0004B436
	pict_data4.bmp	ヘッダ(54byte)	0x00070800
		カラーパレット(1024byte)	0x00070836
		画像データ(113,344byte)	0x00070C36
	pict_cover.bmp	ヘッダ(54byte)	0x00096000
		カラーパレット(1024byte)	0x00096036
		画像データ(113,344byte)	0x00096436
	pict_setting.bmp	ヘッダ(54byte)	0x000BB800
		カラーパレット(1024byte)	0x000BB836
		画像データ(113,344byte)	0x000BBC36
	pict_correction.bmp	ヘッダ(54byte)	0x000E1000
		カラーパレット(1024byte)	0x000E1036
		画像データ(113,344byte)	0x000E1436
	pict_gamma.bmp	ヘッダ(54byte)	0x00106800
		カラーパレット(1024byte)	0x00106836
		画像データ(113,344byte)	0x00106C36

1. カラーパレットの読み出し

カラーパレットは 1024byte あり、B:1byte、G:1byte、R:1byte、A:1byte で構成されています。それらの情報を画像ごとに GLCDC のカラールックアップテーブル(GRnCLUTm[k])(n=1,2、m=0,1、k=0～255)レジスタへ格納します。ただし、BMP の 256 色(8bit)において A:1byte は通常“00h”になっており、そのままレジスタへ格納するとアルファ値が“00h”(ピクセル単位のアルファブレンド機能では下層レイヤが表示)になってしまうため、レジスタへ格納する前に全て“FFh”で上書きします。

なお、BMP の 256 色(8bit)では、カラーパレットの値は画像によらず同じになります。そのため、本アプリケーションノートでは、GLCDC の初期化に合わせて一度だけカラールックアップテーブルレジスタに設定を行っています。初期化以降、カラーパレットの値は読み捨てになります。BMP 以外のフォーマットや、画像ごとに独自のカラーパレットを使用する場合には、画像に合わせてカラールックアップテーブルレジスタの変更も必要になります。

2. 画像データの読み出し

読み出した画像データは、表 8.7 で示すグラフィック 1 用のデータ配置アドレス(0x00800000)またはグラフィック 2 用のデータ配置アドレス(0x00820000)へ直接格納します。GLCDC は、指定したベースアドレスから画像データを読み出すため、画像データを上書きすることで表示画像をそのまま変更します。

ただし、SerialFlash から読み出すデータはプロジェクトの設定、デバイスの設定によらずリトルエンディアンになります。そのため、ビッグエンディアンで動作させる場合は、カラーパレットおよび画像データをバッファ用のデータ配置アドレス(0x00840000)に一度格納し、エンディアン変換を行ってから、それぞれのデータ配置アドレスに格納します。

8.2　ファイル構成

表 8.8 にサンプルプログラムで使用するファイル、表 8.9 に標準インクルードファイルを示します。なお、FIT モジュールのファイルや統合開発環境で自動生成されるファイルは除きます。

表 8.8　サンプルプログラムで使用するファイル

ファイル名	概要	備考
main.c	メイン処理	
r_screen.c	GLCDC の初期化処理、表示画像の制御処理、タッチ制御処理など	
r_screen.h	r_screen.c のヘッダファイル	
r_serial_flash_read.c	RSPI の初期化、SerialFlash からのデータの読み込み処理	
r_serial_flash_read.h	r_serial_flash_read.c のヘッダファイル	

表 8.9　標準インクルードファイル

ファイル名	概要
stdbool.h	論理型、および論理値に関するマクロを定義します。
stdint.h	指定した幅の整数型を宣言してマクロを定義します。
machine.h	RX ファミリ用組み込み関数の形式を定義します。
string.h	文字列の比較、複写等を行うライブラリです。
stddef.h	各標準インクルードファイルで共通に使用するマクロ名を定義します。

8.3　オプション設定メモリ

表 8.10 にサンプルプログラムで使用するオプション設定メモリの状態を示します。必要に応じて、お客様のシステムに最適な値を設定してください。

表 8.10　サンプル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
OFS0	FE7F 5D04h～FE7F 5D07h	FFFF FFFFh	リセット後、IWDT は停止 リセット後、WDT は停止
OFS1	FE7F 5D08h～FE7F 5D0Bh	FFFF FFFFh	リセット後、電圧監視 0 リセット無効 リセット後、HOCO 発振が無効
MDE	FE7F 5D00h～FE7F 5D03h	FFFF FFFFh	リトルエンディアン

8.4　定数一覧

表 8.11、表 8.12 にサンプルコードで使用する定数を示します。

表 8.11　サンプルプログラムで使用する定数(main.c)

定数名	設定値	内容
SW1_PIDR	(PORT0.PIDR.BIT.B3)	SW1(P03)の PIDR レジスタのビット
SW2_PIDR	(PORT0.PIDR.BIT.B5)	SW2(P05)の PIDR レジスタのビット
SW3_PIDR	(PORT0.PIDR.BIT.B7)	SW3(P07)の PIDR レジスタのビット
SW1_PDR	(PORT0.PDR.BIT.B3)	SW1(P03)の PDR レジスタのビット
SW2_PDR	(PORT0.PDR.BIT.B5)	SW2(P05)の PDR レジスタのビット
SW3_PDR	(PORT0.PDR.BIT.B7)	SW3(P07)の PDR レジスタのビット
PUSH_SW1	(1)	SW1 が押下されたときの定義値
PUSH_SW2	(2)	SW2 が押下されたときの定義値
PUSH_SW3	(3)	SW3 が押下されたときの定義値
PUSH_NONE	(-1)	SW が押下されていないときの定義値
LED_ON	(0)	LED の点灯
LED_OFF	(1)	LED の消灯
LED3	(PORTG.PODR.BIT.B5)	LED3(PG5)の PODR レジスタのビット
LED3_PDR	(PORTG.PDR.BIT.B5)	LED3(PG5)の PDR レジスタのビット

表 8.12　サンプルプログラムで使用する定数(r_screen.h)

定数名	設定値	内容
BMP_HEADER_CODE	(0xBEF64D42)	ビットマップファイルの先頭文字列コード（“BM”）
BMP_SIZE	(114422)	ビットマップファイルのサイズ
BMP_HEADER_SIZE	(54)	ビットマップファイルのヘッダサイズ
BMP_COLOR_PALLETE_SIZE	(1024)	ビットマップファイルのカラーパレットサイズ
BMP_IMAGE_DATA_SIZE	(113344)	ビットマップファイルの画像データサイズ
BMP_IMAGE_DATA_OFFSET	(BMP_HEADER_SIZE + BMP_COLOR_PALLETE_SIZE)	ビットマップファイルの先頭から画像データまでのオフセット
BMP_IMAGE_WIDTH	(448)	ビットマップ画像の横幅
BMP_IMAGE_HEIGHT	(253)	ビットマップ画像の縦幅
IMAGE_RELOCATION_ADDR	(0x00800000)	グラフィック 1 用のデータ配置アドレス
IMAGE_RELOCATION_ADDR2	(0x00820000)	グラフィック 2 用のデータ配置アドレス
IMAGE_BUFFER_ADDR	(0x00840000)	バッファ用のデータ配置アドレス(ビッグエンディアン時のみ使用)
IMAGE_BASE_ADDR	(IMAGE_RELOCATION_ADDR + (BMP_IMAGE_DATA_SIZE - BMP_IMAGE_WIDTH))	GLCDC に設定するグラフィック 1 のベースアドレス

定数名	設定値	内容
IMAGE_BASE_ADDR2	(IMAGE_RELOCATION_ADDR2 + (BMP_IMAGE_DATA_SIZE - BMP_IMAGE_WIDTH))	GLCDC に設定するグラフィック 2 のベースアドレス
IMAGE_COUNT_NUM	(4)	表示画像の枚数
IMAGE_COVER_NUM	(4)	重ね合わせ用の画像の枚数
CORRECTION_LEVEL_NUM	(5)	補正処理の設定レベル数
INITIAL_CORRECTION_LEVEL	(2)	アプリケーション起動時の設定値
TOUCH_READ_INFO_SIZE	(5)	タッチパネルコントローラから取得するデータサイズ
TOUCH_BUTTON_NUM	(12)	タッチパネルで使用するボタンの種類数
IMAGE_X_OFFSET	(16)	画像の表示開始位置(X 軸)
IMAGE_Y_OFFSET	(9)	画像の表示開始位置(Y 軸)
RECTANGLE_BLOCK_SIZE	(28)	矩形図の縦横のピクセルサイズ
RECTANGLE_BLOCK_COLOR	(0x80)	矩形図の色(カラーパレットの番号)
RECTANGLE_BUF_SIZE	(32)	矩形図の描画で使用するバッファ

8.5　構造体/列挙型一覧

画面モードの変更やタッチ検出・判定など、アプリケーションの制御で使用する構造体および列挙型を以下に示します。

```
/* ボタンのサイズ構造体 */
typedef struct
{
    uint16_t width;                 /* ボタンの幅 */
    uint16_t height;                /* ボタンの高さ */
} touch_button_size_t;

/* ボタンの情報構造体 */
typedef struct
{
    touch_event_t event;            /* ボタンイベント */
    glcdc_coordinate_t pos;         /* ボタンの左上座標*/
    touch_button_size_t size;       /* ボタンのサイズ */
    uint8_t mode;                   /* ボタンが有効となるモード */
} touch_button_t;

/* タッチコントローラのレジスタ定義情報構造体 */
#pragma bit_order left
#pragma unpack
typedef struct st_touch_info {
    struct {
        uint8_t :5;
        uint8_t touch_points:3;     /* タッチ検出数 */
    } td_status;
    struct {
        uint8_t event_flag:2;       /* タッチ状態(押下、放す、接触状態) */
        uint8_t :2;
        uint8_t x_pos_msb:4;        /* X座標の上位4ビット */
    } touch1_xh;
    uint8_t touch1_xl;              /* X座標の下位8ビット */
    struct {
        uint8_t :4;
        uint8_t y_pos_msb:4;        /* Y座標の上位4ビット */
    } touch1_yh;
    uint8_t touch1_yl;              /* Y座標の下位8ビット */
}touch_info_t;
#pragma bit_order
#pragma packoption
```

図 8.3　構造体

```
/* 画面モード */
typedef enum
{
    IMAGE_VIEW_MODE = 0x01,                 /* 画像表示モード */
    BLEND_VIEW_MODE = 0x02,                 /* ブレンド表示モード */
    SETTING_MODE = 0x04,                    /* 設定モード(設定選択) */
    SETTING_CORRECTION_MODE = 0x08,    /* 設定モード(アルファ値・輝度・コントラスト設定) */
    SETTING_GAMMA_MODE = 0x10,          /* 設定モード(ガンマ設定) */
    ALL_MODE = 0x1F                         /* モード共通 */
} view_mode_t;

/* タッチイベント */
typedef enum
{
    NO_TOUCH = 0U,
   /* 画像表示モード */
    TOUCH_LEFT,                 /* 左ボタン */
    TOUCH_RIGHT,                /* 右ボタン */
   /* 設定モード(設定選択) */
    TOUCH_CORRECTION,      /* アルファ値、輝度、コントラスト設定ボタン */
    TOUCH_GAMMA,              /* ガンマ設定ボタン */
   /* 設定モード(アルファ値、輝度、コントラスト設定またはガンマ設定) */
    TOUCH_BAR1_DEC,         /* バー1のデクリメントボタン */
    TOUCH_BAR1_INC,         /* バー1のインクリメントボタン */
    TOUCH_BAR2_DEC,         /* バー2のデクリメントボタン */
    TOUCH_BAR2_INC,         /* バー2のインクリメントボタン */
    TOUCH_BAR3_DEC,         /* バー3のデクリメントボタン */
    TOUCH_BAR3_INC,          /* バー3のインクリメントボタン */
    TOUCH_BACK,               /* バックボタン */
   /* モード共通 */
    TOUCH_OTHER,              /* ボタン以外 */
} touch_event_t;
```

図 8.4　列挙型

8.6　変数一覧

表 8.13～表 8.14 に static 型変数を、表 8.15 に const 型変数を示します。FIT モジュールが持つ変数は除きます。

表 8.13　static 型変数(main.c)

型	変数名	内容	使用関数
static volatile bool	scan_period_flag	スイッチおよびタッチパネル検出周期チェック用フラグ	main set_scan_period_flag

表 8.14　static 型変数(r_screen.c)

型	変数名	内容	使用関数
sci_iic_info_t	siic_info	簡易 I^2C モジュールの設定用	touch_initialize scan_touch
uint8_t	touch_data[TOUCH_READ_INFO_SIZE]	タッチコントローラからの受信データ	scan_touch
glcdc_cfg_t	glcdc_init_cfg	GLCDC の初期設定用	screen_mode_change screen_update glcdc_initialize
uint32_t	gr_clut_table[256]	カラーパレットデータ	screen_mode_change screen_update glcdc_initialize
bool	first_interrupt_flag	GLCDC の初回割り込みチェック用フラグ	glcdc_initialize glcdc_callback
view_mode_t	current_mode	カレントモード	get_current_mode screen_mode_change
int8_t	image_count	表示中の画像の番号	screen_mode_change screen_update
int8_t	blend_image1_count	ブレンド表示中の画像の番号 1	screen_mode_change screen_update
int8_t	blend_image2_count	ブレンド表示中の画像の番号 2	screen_mode_change screen_update
uint8_t	current_blend_level	ブレンド表示におけるカレント値	screen_mode_change screen_update
uint8_t	current_bright_level	輝度のカレント値	screen_mode_change screen_update
uint8_t	current_contrast_level	コントラストのカレント値	screen_mode_change screen_update
uint8_t	current_gamma_r_level	ガンマのカレント値(R 値)	screen_mode_change screen_update
uint8_t	current_gamma_g_level	ガンマのカレント値(G 値)	screen_mode_change screen_update
uint8_t	current_gamma_b_level	ガンマのカレント値(B 値)	screen_mode_change screen_update

表 8.15　const 型変数(r_screen.c)

型	変数名	内容	使用関数
uint32_t	gp_pict_table[IMAGE_COUNT_NUM]	SerialFlash の画像配置アドレステーブル(画像 1～画像 4)	main glcdc_initialize screen_mode_change screen_update bmp_image_check
uint32_t	gp_pict_cover_table[IMAGE_COVER_NUM]	SerialFlash の画像配置アドレステーブル(重ね合わせ画像)	glcdc_initialize screen_mode_change bmp_image_check
uint32_t	g_common_level_table[CORRECTION_LEVEL_NUM]	コントラスト/ブレンド比の設定テーブル	screen_mode_change screen_update
uint32_t	g_bright_level_table[CORRECTION_LEVEL_NUM]	輝度の設定テーブル	screen_update
gamma_correction_t	g_gamma_table	ガンマ補正用データ	glcdc_initialize
glcdc_coordinate_t	g_cross_point_bar_table[3][5]	バーの交点座標テーブル	screen_mode_change
touch_button_t	g_touch_button_table[TOUCH_BUTTON_NUM]	LCD パネルのボタン配置座標テーブル	scan_touch

8.7　関数一覧

表 8.16～表 8.18 に関数を示します。

表 8.16　関数(main.c)

関数名	概要
main	メイン処理
check_sw	スイッチのチェック
set_scan_period_flag	スイッチ、タッチパネルの検出周期フラグのセット

表 8.17　関数(r_screen.c)

関数名	概要
get_current_mode	現在の画面モードの取得
screen_mode_change	画面モードの変更
screen_update	表示画像および GLCDC の設定更新
glcdc_initialize	GLCDC の初期化と動作開始
glcdc_port_setting	LCD パネルで使用する端子の設定
glcdc_callback	GLCDC の API 関数のコールバック処理
touch_initialize	タッチパネル通信の初期化
scan_touch	タッチパネルからのタッチ情報取得
touch_callback	簡易 I^2C FIT モジュールのコールバック処理
fill_clut_alpha_data	カラーパレットのアルファ値上書き
draw_rectangle	矩形描画
bmp_image_load	SerialFlash からの画像データ読み出し
bmp_image_check	SerialFlash に格納されている画像のチェック
convert_endian	エンディアン変換 (CPU のエンディアンがビッグエンディアンのときのみ使用)

表 8.18　関数(r_serial_flash_read.c)

関数名	概要
serialflash_read_initialize	SerialFlash 通信の初期化
data_read	SerialFlash からのデータ読み込み処理

8.8　関数仕様

サンプルプログラムの関数仕様を示します。

8.8.1　関数(main.c)

main	
概　要	メイン処理
ヘッダ	なし
宣　言	void main(void)
説　明	GLCDC の初期設定後、スイッチの押下や LCD パネルのタッチによって表示する画像や画面モードを切り替えます。
引　数	なし
リターン値	なし

check_sw	
概　要	スイッチのチェック
ヘッダ	なし
宣　言	static int8_t check_sw (void)
説　明	スイッチが押されると、押されたスイッチの値を返します。
引　数	なし
リターン値	押下したスイッチの番号 1　: SW1 2　: SW2 3　: SW3 -1　: それ以外

set_scan_period_flag	
概　要	スイッチ、タッチパネルの検出周期フラグのセット
ヘッダ	なし
宣　言	static void set_scan_period_flag (void)
説　明	検出周期のフラグを“true”にセットします。
引　数	なし
リターン値	なし

8.8.2　関数(r_screen.c)

get_current_mode		
概　要	現在の画面モードの取得	
ヘッダ	screen.h	
宣　言	view_mode_t get_current_mode (void)	
説　明	現在の画面モードのステータスを返します。	
引　数	なし	
リターン値	IMAGE_VIEW_MODE	: 画像表示モード
	BLEND_VIEW_MODE	: ブレンド表示モード
	SETTING_MODE	: 設定モード(設定選択)
	SETTING_CORRECTION_MODE	: 設定モード(アルファ値・輝度・コントラスト設定)
	SETTING_GAMMA_MODE	: 設定モード(ガンマ設定)

screen_mode_change		
概　要	画面モードの変更	
ヘッダ	screen.h	
宣　言	void screen_mode_change(view_mode_t mode)	
説　明	引数で指定した表示モードに変更します。	
引　数	view_mode_t mode	変更先のモード
リターン値	なし	

screen_update		
概　要	表示画像および GLCDC の設定更新	
ヘッダ	screen.h	
宣　言	void screen_update(view_mode_t mode, touch_event_t event)	
説　明	第 1 引数と第 2 引数の値に応じて、表示画像の更新や、GLCDC の設定(アルファ値、輝度・コントラスト、ガンマ)の変更を行います。	
引　数	view_mode_t mode	現在の画面モード
	touch_event_t	イベント
リターン値	なし	

glcdc_initialize	
概　要	GLCDC の初期設定と動作開始
ヘッダ	screen.h
宣　言	static void glcdc_initialize (void)
説　明	画像データの準備、GLCDC の端子設定、GLCDC の初期化と GLCDC の動作を開始します。
引　数	なし
リターン値	なし

glcdc_port_setting	
概　要	LCD パネルで使用する端子の設定
ヘッダ	なし
宣　言	static void glcdc_port_setting (void)
説　明	GLCDC の端子設定を行います。
引　数	なし
リターン値	なし

glcdc_callback		
概　要	GLCDC の API 関数のコールバック	
ヘッダ	なし	
宣　言	static void glcdc_callback(void * pdata)	
説　明	GLCDC の割り込み処理内から呼び出される関数。 API 関数でコールバック関数の登録をしています。	
引　数	void * pdata	発生した割り込みの情報 GLCDC_EVENT_GR1_UNDERFLOW : グラフィック 1 アンダフロー GLCDC_EVENT_GR2_UNDERFLOW : グラフィック 2 アンダフロー GLCDC_EVENT_LINE_DETECTION :グラフィック 2 ライン検出
リターン値	なし	
備考	GLCDC ソフトウェアリセット解除後の初回のみ、意図しないグラフィック 2 指定ライン通知、グラフィック 1,2 アンダフローが検出されます。そのため、本サンプルプログラムでは R_GLCDC_Open 関数実行後の初回のグラフィック 2 指定ライン検出による割り込み処理では何もせず、次回の割り込み処理からユーザ処理を実行するようにしています。詳細はグラフィック LCD コントローラモジュールのマニュアルを参照してください。ユーザシステムに組み込む場合は、必要に応じて各割り込みに対する処理を組み込んでください。	

touch_initialize	
概　要	タッチパネル通信の初期化
ヘッダ	screen.h
宣　言	void touch_initialize(void)
説　明	簡易 I²C FIT モジュールの初期化と通信設定
引　数	なし
リターン値	なし

scan_touch	
概　要	タッチパネルからのタッチ情報取得
ヘッダ	screen.h
宣　言	touch_event_t scan_touch(view_mode_t mode)
説　明	タッチパネルコントローラと通信し、タッチされたボタンの情報を取得します。
引　数	view_mode_t mode　　モード情報
リターン値	[全モード]
	NO_TOUCH　　：未タッチ
	TOUCH_OTHER　　：ボタン以外
	[画像表示モードのとき]
	TOUCH_LEFT　　：左ボタン (画像表示モードのとき)
	TOUCH_RIGHT　　：右ボタン (画像表示モードのとき)
	[設定モード(設定選択)のとき]
	TOUCH_CORRECTION　：　アルファ値、輝度、コントラスト設定ボタン
	TOUCH_GAMMA　　：ガンマ設定ボタン
	[設定モード(アルファ値、輝度、コントラスト設定またはガンマ設定)のとき]
	TOUCH_BAR1_DEC　　：バー1 のデクリメントボタン
	TOUCH_BAR1_INC　　：バー1 のインクリメントボタン
	TOUCH_BAR2_DEC　　：バー2 のデクリメントボタン
	TOUCH_BAR2_INC　　：バー2 のインクリメントボタン
	TOUCH_BAR3_DEC　　：バー3 のデクリメントボタン
	TOUCH_BAR3_INC　　：バー3 のインクリメントボタン
	TOUCH_BACK　　：バックボタン

touch_callback	
概　要	簡易 I²C FIT モジュールのコールバック処理
ヘッダ	なし
宣　言	void touch_callback(void)
説　明	簡易 I²C FIT モジュールのコールバック処理
引　数	なし
リターン値	なし

fill_clut_alpha_data	
概　要	カラーパレットのアルファ値上書き
ヘッダ	なし
宣　言	static void fill_clut_alpha_data(uint8_t * p_clut_data)
説　明	引数に指定したカラーパレットデータのアルファ値を 0xFF で上書きします。
引　数	uint8_t * p_clut_data　　カラーパレットのポインタ
リターン値	なし

draw_rectangle	
概　要	矩形描画
ヘッダ	なし
宣　言	static void draw_rectangle (glcdc_coordinate_t draw_point)
説　明	引数で指定したグラフィック 2 の座標上に 28px×28px の矩形を描画します。指定の座標から-14(定数 RECTANGLE_BLOCK_SIZE の半分の値)の位置が描画の開始位置になります。輝度値・コントラスト値・ガンマ値の現在レベルを表示する際に使用します。
引　数	glcdc_coordinate_t draw_point　　矩形描画する座標
リターン値	なし
備考	本関数は、ビットマップ画像(256 色)の画像フォーマットに矩形を描画するための関数です。

bmp_image_load	
概　要	SerialFlash からの画像データ読み出し
ヘッダ	なし
宣　言	static void bmp_image_load (uint32_t image_data, uint8_t * p_dest_addr, uint8_t * p_clut_data)
説　明	指定した SerialFlash のアドレスのビットマップ画像(256 色)からカラーパレットデータと画像データを抽出して、指定のアドレスにコピーします。
引　数	uint32_t image_data　　SerialFlash のビットマップ画像の先頭アドレス uint8_t * p_dest_addr　　画像データのコピー先 uint8_t * p_clut_data　　カラーパレットのコピー先
リターン値	なし
備考	ビッグエンディアンで動作させる場合、データのエンディアン変換をしてからコピーします。

bmp_image_check	
概　要	SerialFlash に格納されている画像のチェック
ヘッダ	なし
宣　言	bool bmp_image_check (void)
説　明	SerialFlash に画像データが格納されているかを確認します。各画像データが格納されているはずの先頭 4 バイトを読み出し、ビットマップファイルの先頭文字列コード（“BM”)と一致するかをチェックしています。
引　数	なし
リターン値	true　：画像データあり false　：画像データなし

convert_endian		
概　要	エンディアン変換	
ヘッダ	なし	
宣　言	static void convert_endian (uint32_t *p_src_addr, uint32_t size)	
説　明	指定データのエンディアンを変換します。	
引　数	uint32_t * p_src_addr	変換したいデータのポインタ
	uint32_t size	変換したいデータのサイズ(4 の倍数バイト)
リターン値	なし	
備考	本関数は、コンパイラのエンディアンオプションでビッグエンディアンを選択したときに有効になります。	

8.8.3　関数(r_serial_flash_read.c)

serialflash_read_initialize	
概　要	SerialFlash 通信の初期化
ヘッダ	r_serial_flash_read.h
宣　言	void serialflash_read_initialize (void)
説　明	SerialFlash との通信の初期化を行います。
引　数	なし
リターン値	なし

data_read		
概　要	SerialFlash からのデータ読み込み処理	
ヘッダ	r_serial_flash_read.h	
宣　言	flash_spi_status_t data_read(uint8_t *p_dest_addr,　uint32_t src_addr, uint32_t data_size)	
説　明	指定の画像データを SerialFlash から読み込み、指定アドレスに配置します。	
引　数	uint8_t * p_dest_addr	読み込む画像データの配置先アドレス
	uint32_t src_addr	SerialFlash に格納されている画像データのアドレス
	uint32_t size	読み込む画像データのサイズ
リターン値	FLASH_SPI_SUCCESS	：正常終了
	FLASH_SPI_ERR_PARAM	：パラメータエラー
	FLASH_SPI_ERR_HARD	：ハードウェアエラー
	FLASH_SPI_ERR_OTHER	：その他のエラー

8.9　概要フローチャート

本章では、アプリケーションにおける概要フローチャートを示します。

8.9.1　メイン処理

メイン処理では、各周辺機能の初期化を行った後、RSK のスイッチやタッチ入力を定期的に確認し、画面モードの変更や、タッチ入力に応じた処理を行います。

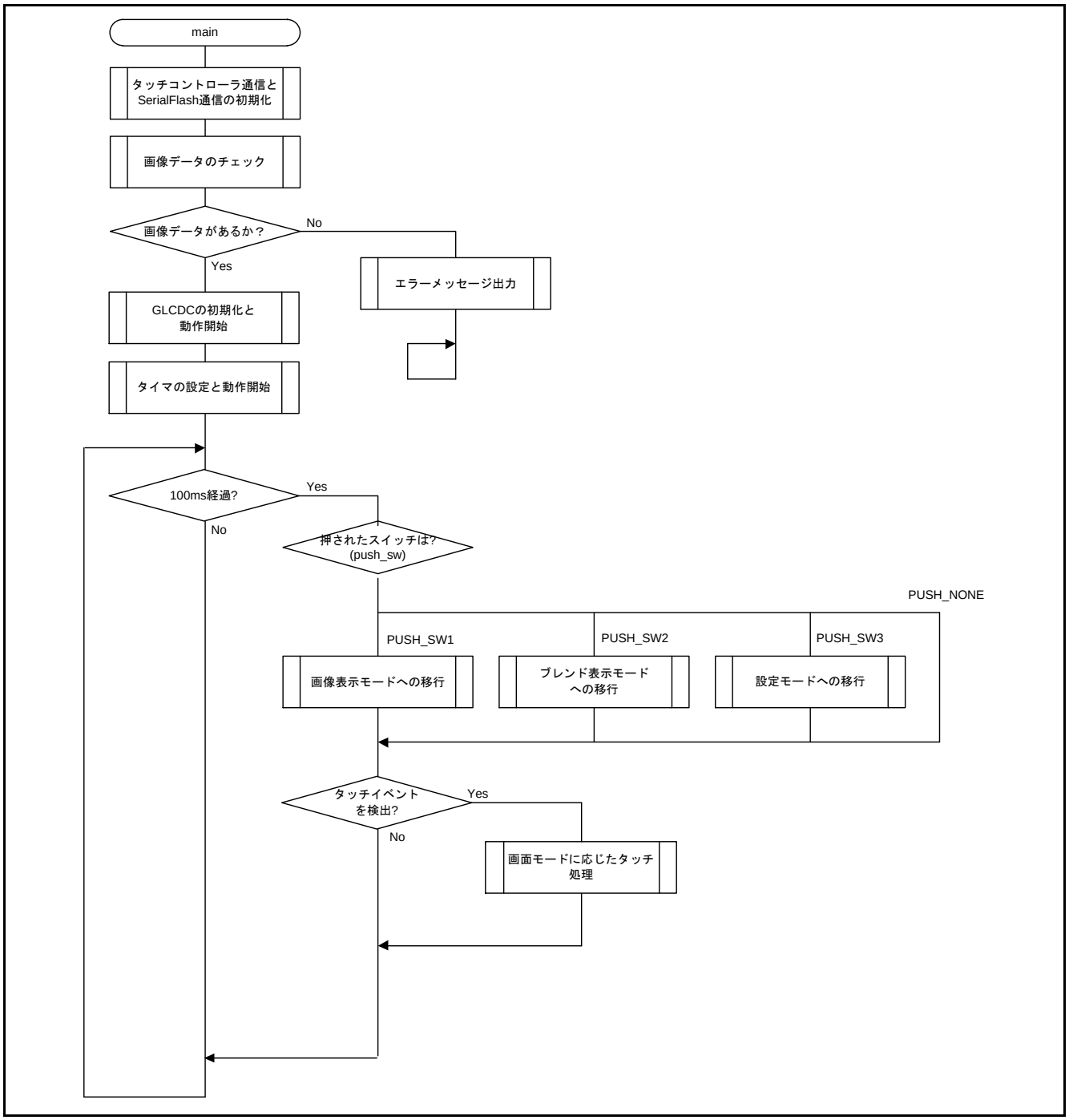


図 8.5　メイン処理の概要フロー

8.9.2　GLCDC の初期化と動作開始

GLCDC を動作させる前に SerialFlash から画像データを読み出し、GLCDC に読み込ませるメモリ領域へ格納しておきます。その後、R_GLCDC_Open 関数で GLCDC の初期化を行い、R_GLCDC_Control 関数で GLCDC の動作を開始します。これにより、メモリ領域から画像データが読み込まれて LCD パネルへ画像が表示されます。

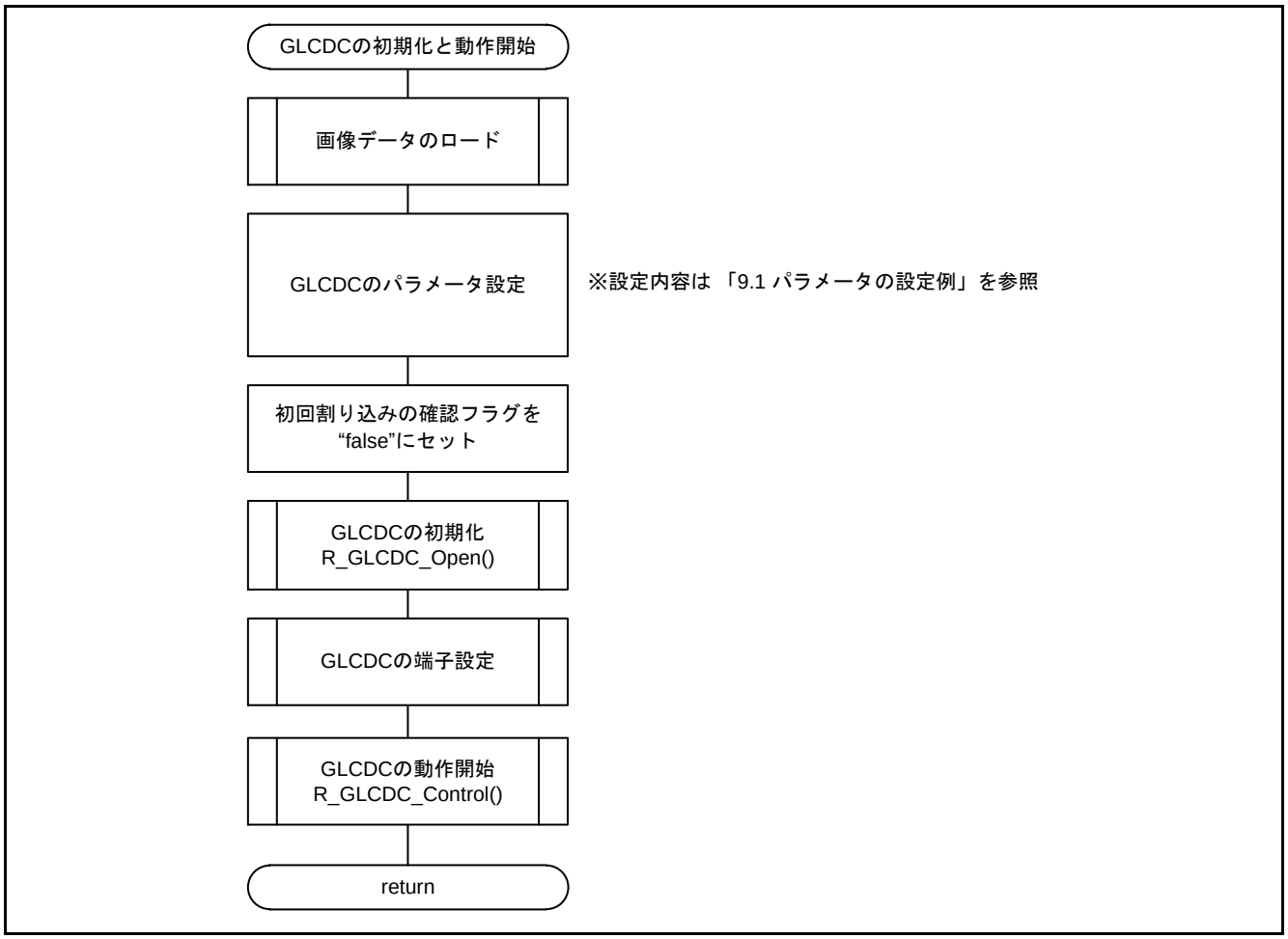


図 8.6　GLCDC 初期化の概要フロー

8.9.3　　モード移行

モード移行では、移行するモードに応じた処理を行います。各モードに対応する画像データを SerialFlash から読み出し、表示中の画像データが格納されているメモリ領域に上書きします。GLCDC は動作している間、指定領域から繰り返し画像データを読み出しており、画像データをその領域へ直接上書きすることで、LCD パネルに表示される画像が変わります。そして、必要に応じて GLCDC の設定を更新し、各モードに必要な設定に切り替えます。アルファブレンドやクロマキー処理、グラフィック 1 またはグラフィック 2 の画面構成の変更は R_GLCDC_LayerChange 関数で行います。

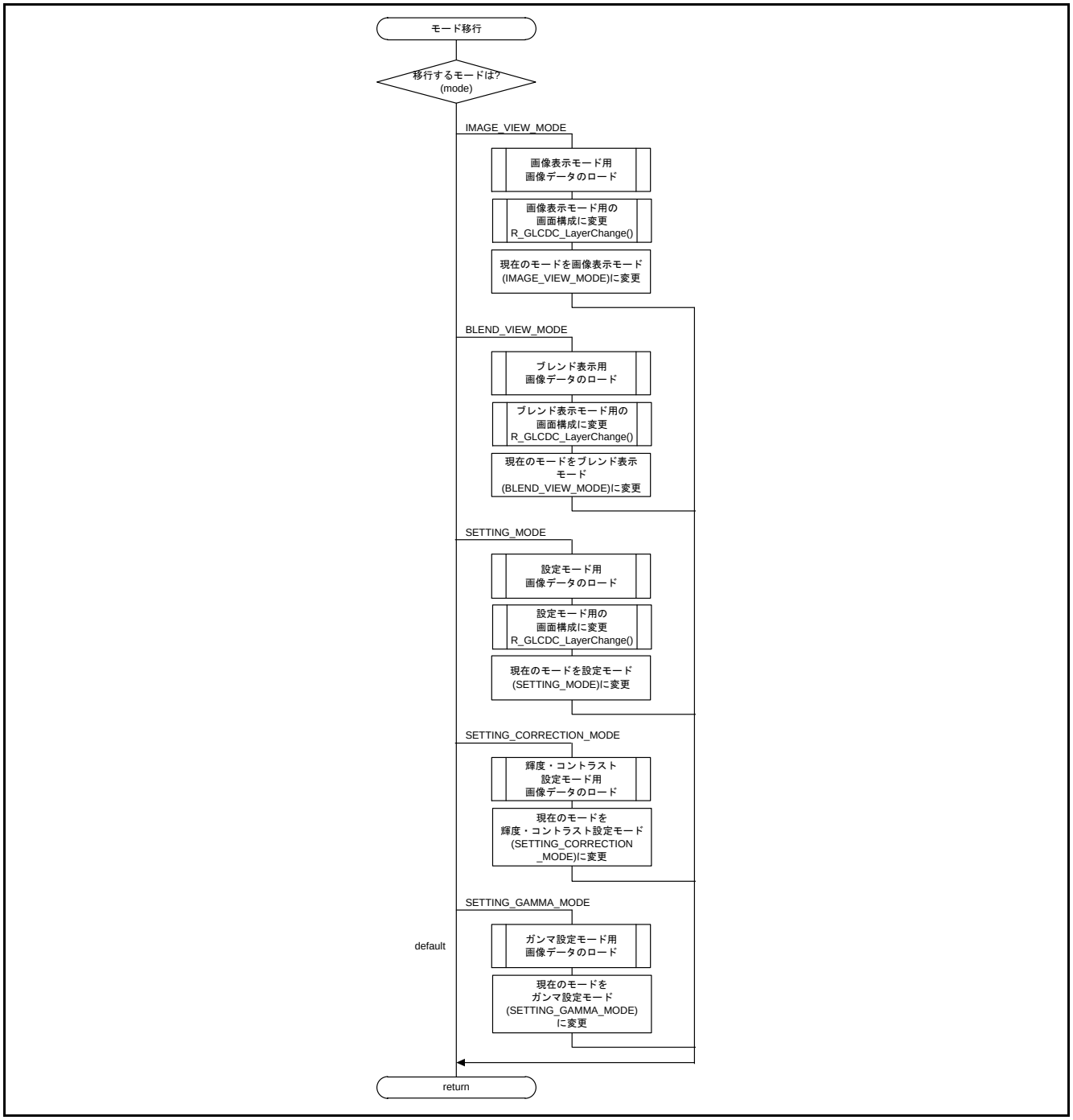


図 8.7　モード移行処理の概要フロー

8.9.4　画面モードに応じたタッチ処理

(1)　画像表示モードのときのタッチ処理

画像表示モード中にタッチ入力を行うと、表示画像を切り替えます。タッチイベントを検出すると画像データを SerialFlash から読み出し、表示中の画像データが格納されているメモリ領域に上書きします。上書きすると、表示される画像が変わります。

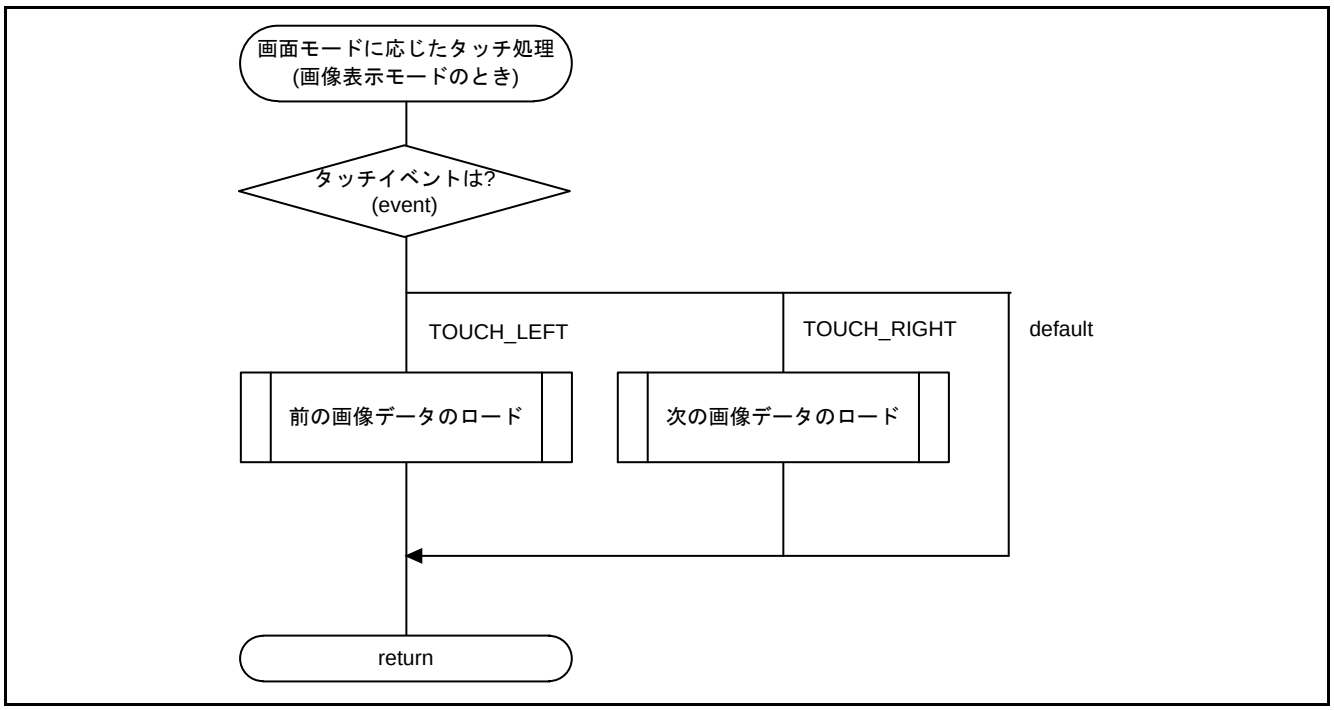


図 8.8　画像表示モードにおけるタッチ処理の概要フロー

(2)　ブレンド表示モードのときのタッチ処理

ブレンド表示モード中にタッチ入力を行うと、ブレンド画像を切り替えます。タッチイベントを検出すると、画像データを SerialFlash から読み出し、表示中の画像データが格納されているメモリ領域に上書きします。上書きすると、表示される画像が変わります。

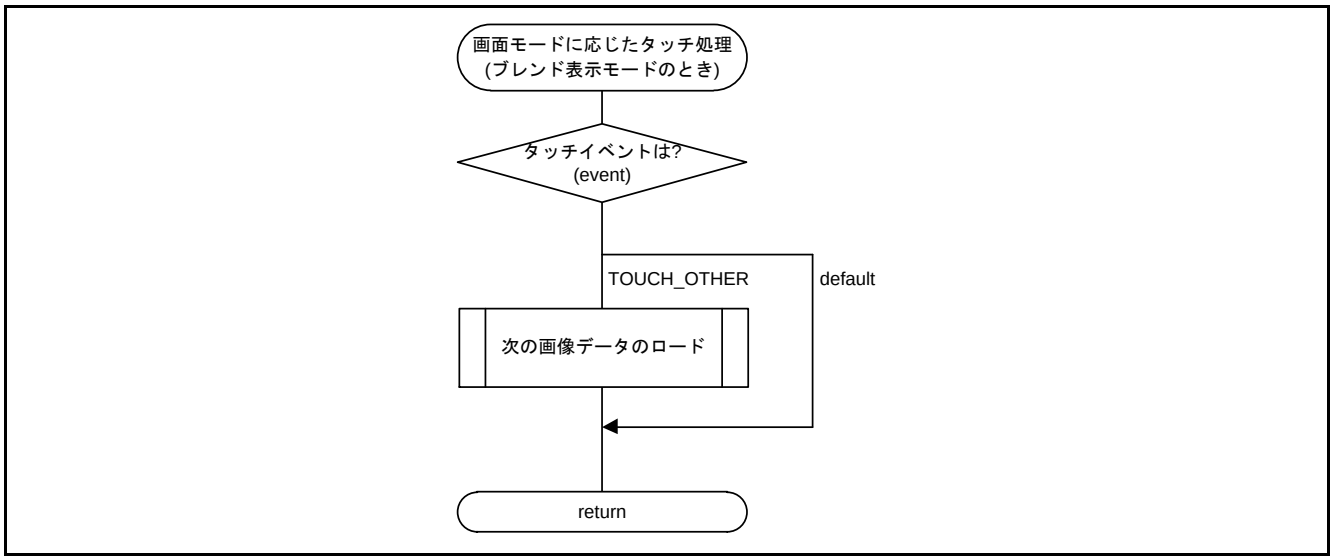


図 8.9　ブレンド表示モードにおけるタッチ処理の概要フロー

(3)　設定モードのときのタッチ処理

設定モード中にタッチ入力を行うと、モード移行および表示画像を切り替えます。タッチイベントを検出するとイベントに応じたモード移行を行い、それぞれのモード用の画像を SerialFlash から読み出して、表示中の画像データが格納されているメモリ領域に上書きします。上書きすると、表示される画像が変わります。

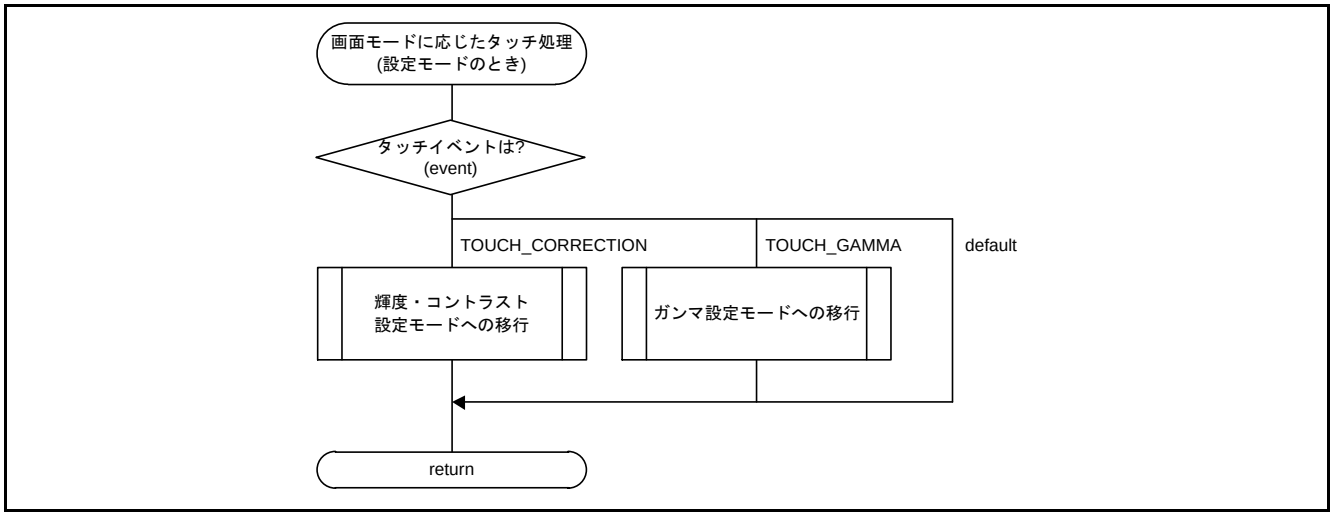


図 8.10　設定モードにおけるタッチ処理の概要フロー

(4)　輝度・コントラスト設定モードのときのタッチ処理

輝度・コントラスト設定モード中にタッチ入力を行うと、アルファブレンドや輝度・コントラストの変更を行います。アルファブレンド値の変更は、ブレンド表示モードへの移行時に反映されます。輝度・コントラスト値は、R_GLCDC_ColorCorrection 関数で変更します。画面には関数実行後に反映されます。

変更後、画面の横バーに設定内容を反映させるため、輝度・コントラスト設定モードへの移行処理(表示画面の更新)を行います。

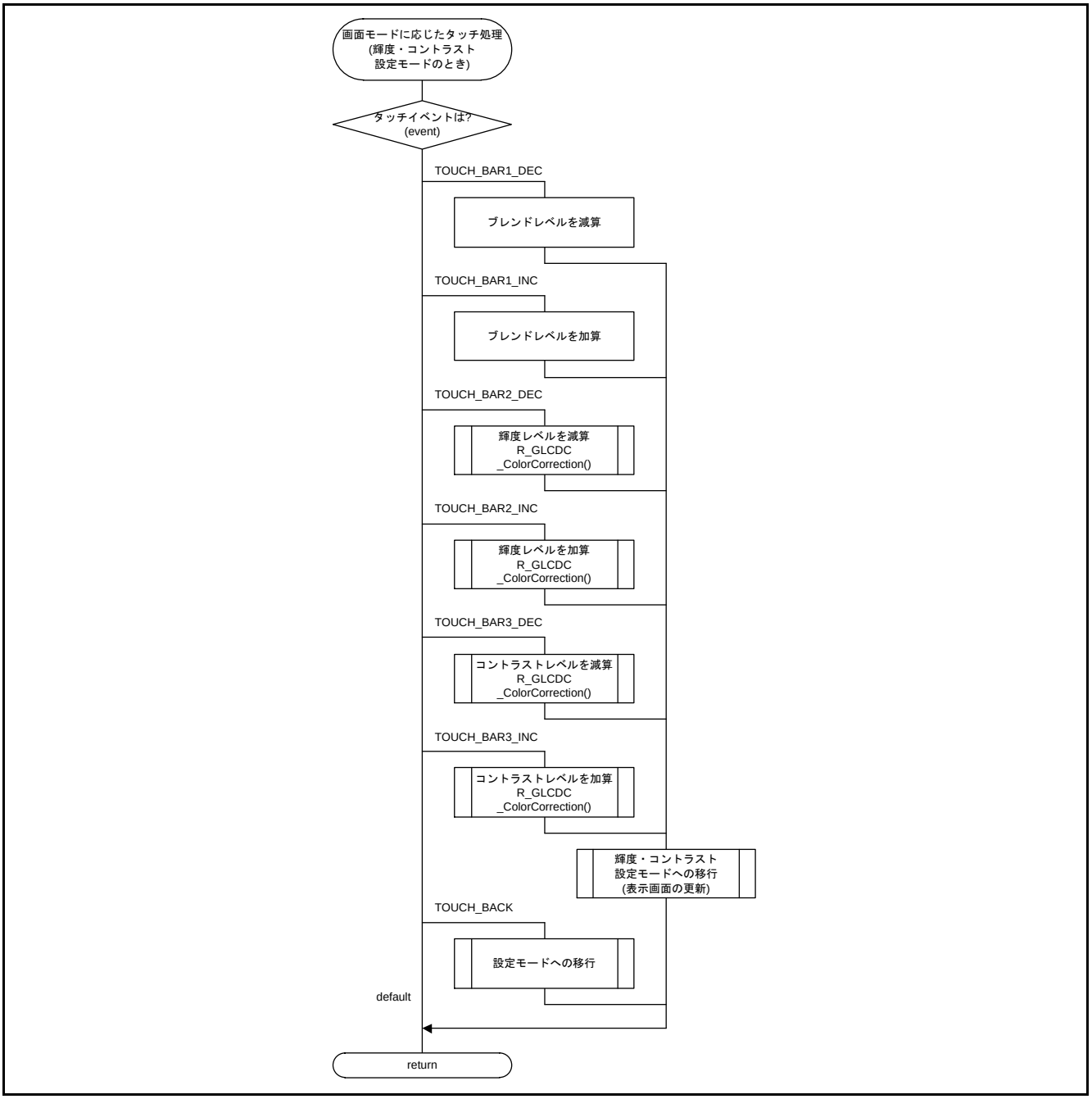


図 8.11　輝度・コントラスト設定モードにおけるタッチ処理の概要フロー

(5)　ガンマ設定モードのときのタッチ処理

ガンマ設定モード中にタッチ入力を行うと、RGB ガンマ値の変更を行います。ガンマ値は、R_GLCDC_ColorCorrection 関数で変更します。画面には関数実行後に反映されます。

変更後、画面の横バーに設定内容を反映させるため、ガンマ設定モードへの移行処理(表示画面の更新)を行います。

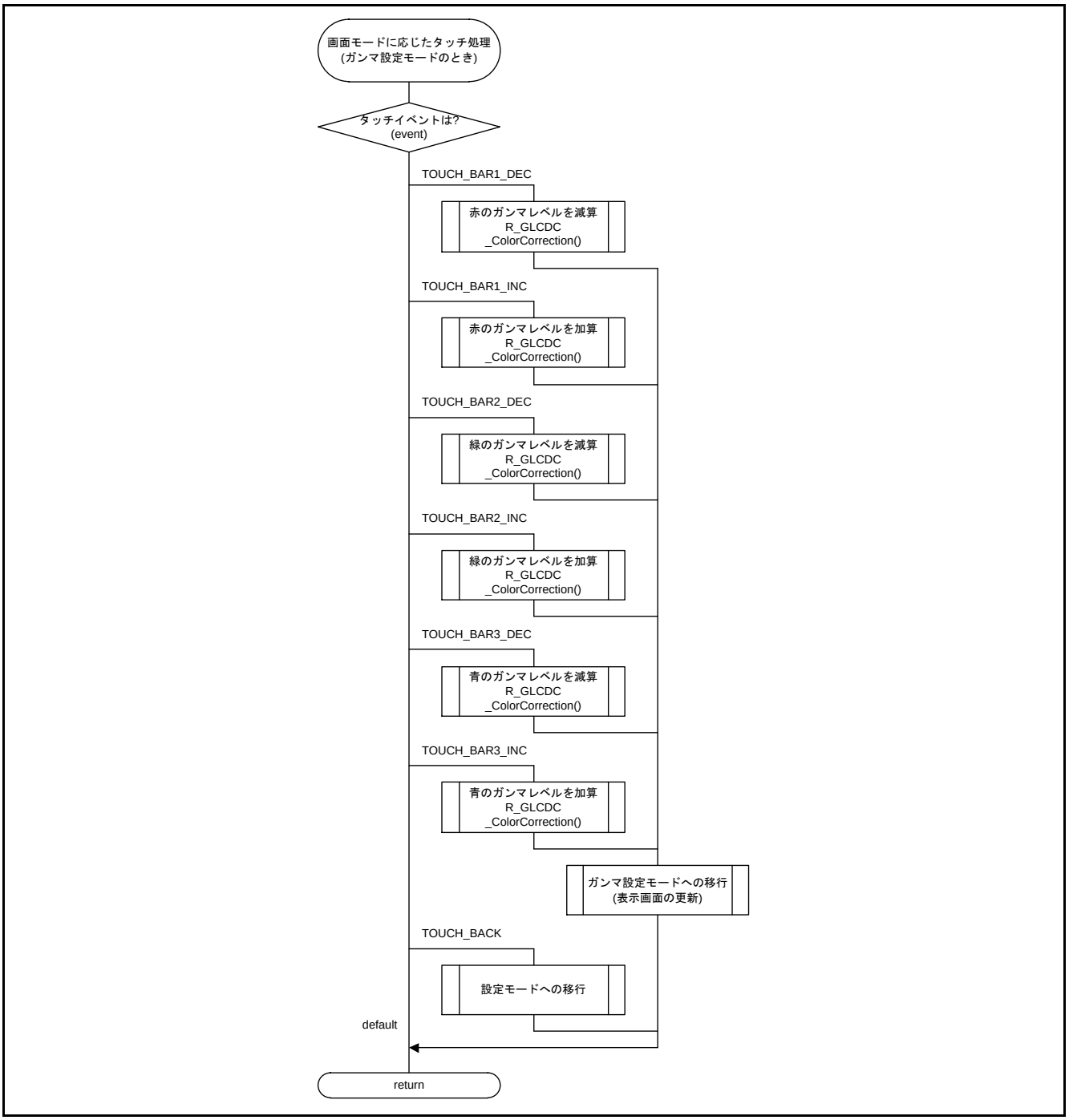


図 8.12　ガンマ設定モードにおけるタッチ処理の概要フロー

9. 付録

9.1 パラメータの設定例

R_GLCDC_Open 関数の引数 glcdc_cfg_t p_cfg に設定するパラメータの設定例を以下に示します。設定内容は、r_screen.c の glcdc_initialize 関数で設定しているものと同じです。

```
/* Include */
#include "r_glcdc_rx_if.h"

/* Variables declaration */
glcdc_cfg_t        glcdc_init_cfg;        /* Declaration of glcdc_cfg_t structure variable */
uint32_t           gr_clut_table[256];    /* Declaration of clut table variables */

/* gamma table (ex. r = 0.9) */
const gamma_correction_t g_gamma_table =
{
  /* Gain (gamma value = 0.90) <0 to 2047> */
  { 753, 873, 925, 961, 988, 1010, 1029, 1046, 1061, 1074, 1086, 1097, 1107, 1117, 1126, 1116 },
  /* Threshold (increase by 64) <0 to 1023> */
  {  64, 128, 192, 256, 320,  384,  448,  512,  576,  640,  704,  768,  832,  896,  960 }
};


    /*-- Open parameter configure --*/
    /* Graphic 2 setting */
    glcdc_init_cfg.input[GLCDC_FRAME_LAYER_2].p_base = (uint32_t *) IMAGE_BASE_ADDR2;
    glcdc_init_cfg.input[GLCDC_FRAME_LAYER_2].hsize = BMP_IMAGE_WIDTH;
    glcdc_init_cfg.input[GLCDC_FRAME_LAYER_2].vsize = BMP_IMAGE_HEIGHT;
    glcdc_init_cfg.input[GLCDC_FRAME_LAYER_2].offset = (BMP_IMAGE_WIDTH * -1);
    glcdc_init_cfg.input[GLCDC_FRAME_LAYER_2].format = GLCDC_IN_FORMAT_CLUT8;
    glcdc_init_cfg.input[GLCDC_FRAME_LAYER_2].frame_edge = false;
    glcdc_init_cfg.input[GLCDC_FRAME_LAYER_2].coordinate.x = IMAGE_X_OFFSET;
    glcdc_init_cfg.input[GLCDC_FRAME_LAYER_2].coordinate.y = IMAGE_Y_OFFSET;
    glcdc_init_cfg.input[GLCDC_FRAME_LAYER_2].bg_color.argb = 0x00CCCCCC;

    glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_2].visible = true;
    glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_2].blend_control = GLCDC_BLEND_CONTROL_PIXEL;
    glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_2].fixed_blend_value = 0x00;
    glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_2].fade_speed = 0x00;
    glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_2].frame_edge = false;
    glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_2].start_coordinate.x = IMAGE_X_OFFSET;
    glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_2].start_coordinate.y = IMAGE_Y_OFFSET;
    glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_2].end_coordinate.x = (BMP_IMAGE_WIDTH + IMAGE_X_OFFSET);
    glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_2].end_coordinate.y = (BMP_IMAGE_HEIGHT + IMAGE_Y_OFFSET);

    glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_2].enable = true;
    glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_2].before.byte.r = 0xFF;
    glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_2].before.byte.g = 0xFF;
    glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_2].before.byte.b = 0xFF;
    glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_2].after.byte.a = 0x00;
    glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_2].after.byte.r = 0xFF;
    glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_2].after.byte.g = 0xFF;
    glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_2].after.byte.b = 0xFF;

    glcdc_init_cfg.clut[GLCDC_FRAME_LAYER_2].enable = true;
    glcdc_init_cfg.clut[GLCDC_FRAME_LAYER_2].p_base = (uint32_t *) gr_clut_table;
    glcdc_init_cfg.clut[GLCDC_FRAME_LAYER_2].size = 256;
    glcdc_init_cfg.clut[GLCDC_FRAME_LAYER_2].start = 0;
```

図 9.1　パラメータの設定例(1/3)

```
/* Graphic 1 setting */
glcdc_init_cfg.input[GLCDC_FRAME_LAYER_1].p_base = (uint32_t *) IMAGE_BASE_ADDR;
glcdc_init_cfg.input[GLCDC_FRAME_LAYER_1].hsize = BMP_IMAGE_WIDTH;
glcdc_init_cfg.input[GLCDC_FRAME_LAYER_1].vsize = BMP_IMAGE_HEIGHT;
glcdc_init_cfg.input[GLCDC_FRAME_LAYER_1].offset = (BMP_IMAGE_WIDTH * -1);
glcdc_init_cfg.input[GLCDC_FRAME_LAYER_1].format = GLCDC_IN_FORMAT_CLUT8;
glcdc_init_cfg.input[GLCDC_FRAME_LAYER_1].frame_edge = false;
glcdc_init_cfg.input[GLCDC_FRAME_LAYER_1].coordinate.x = IMAGE_X_OFFSET;
glcdc_init_cfg.input[GLCDC_FRAME_LAYER_1].coordinate.y = IMAGE_Y_OFFSET;
glcdc_init_cfg.input[GLCDC_FRAME_LAYER_1].bg_color.argb = 0x00CCCCCC;

glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_1].visible = true;
glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_1].blend_control = GLCDC_BLEND_CONTROL_NONE;
glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_1].fixed_blend_value = 0x00;
glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_1].fade_speed = 0x00;
glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_1].frame_edge = false;
glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_1].start_coordinate.x = 0;
glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_1].start_coordinate.y = 0;
glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_1].end_coordinate.x = 0;
glcdc_init_cfg.blend[GLCDC_FRAME_LAYER_1].end_coordinate.y = 0;

glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_1].enable = false;
glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_1].before.byte.r = 0x00;
glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_1].before.byte.g = 0x00;
glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_1].before.byte.b = 0x00;
glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_1].after.byte.a = 0x00;
glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_1].after.byte.r = 0x00;
glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_1].after.byte.g = 0x00;
glcdc_init_cfg.chromakey[GLCDC_FRAME_LAYER_1].after.byte.b = 0x00;

glcdc_init_cfg.clut[GLCDC_FRAME_LAYER_1].enable = true;
glcdc_init_cfg.clut[GLCDC_FRAME_LAYER_1].p_base = (uint32_t *) gr_clut_table;
glcdc_init_cfg.clut[GLCDC_FRAME_LAYER_1].size = 256;
glcdc_init_cfg.clut[GLCDC_FRAME_LAYER_1].start = 0;



/* Output timing */
glcdc_init_cfg.output.htiming.front_porch = 3;
glcdc_init_cfg.output.htiming.back_porch = 2;
glcdc_init_cfg.output.htiming.display_cyc = 480;
glcdc_init_cfg.output.htiming.sync_width = 41;

glcdc_init_cfg.output.vtiming.front_porch = 2;
glcdc_init_cfg.output.vtiming.back_porch = 2;
glcdc_init_cfg.output.vtiming.display_cyc = 272;
glcdc_init_cfg.output.vtiming.sync_width = 10;

/* Output format */
glcdc_init_cfg.output.format = GLCDC_OUT_FORMAT_16BITS_RGB565;
glcdc_init_cfg.output.endian = GLCDC_ENDIAN_LITTLE;
glcdc_init_cfg.output.color_order = GLCDC_COLOR_ORDER_BGR;
glcdc_init_cfg.output.data_enable_polarity = GLCDC_SIGNAL_POLARITY_HIACTIVE;
glcdc_init_cfg.output.hsync_polarity = GLCDC_SIGNAL_POLARITY_LOACTIVE;
glcdc_init_cfg.output.vsync_polarity = GLCDC_SIGNAL_POLARITY_LOACTIVE;
glcdc_init_cfg.output.sync_edge = GLCDC_SIGNAL_SYNC_EDGE_RISING;
glcdc_init_cfg.output.bg_color.argb = 0x00CCCCCC;

/* Output pin */
glcdc_init_cfg.output.tcon_hsync = GLCDC_TCON_PIN_2;
glcdc_init_cfg.output.tcon_vsync = GLCDC_TCON_PIN_0;
glcdc_init_cfg.output.tcon_de = GLCDC_TCON_PIN_3;

/* Output clock */
glcdc_init_cfg.output.clksrc = GLCDC_CLK_SRC_INTERNAL;
glcdc_init_cfg.output.clock_div_ratio = GLCDC_PANEL_CLK_DIVISOR_24;
```

図 9.2　パラメータの設定例(2/3)

```
/* Correction circuit sequence */
glcdc_init_cfg.output.correction_proc_order = GLCDC_BRIGHTNESS_CONTRAST_TO_GAMMA;

/* Brightness */
glcdc_init_cfg.output.brightness.enable = true;

glcdc_init_cfg.output.brightness.r = 0x200;
glcdc_init_cfg.output.brightness.g = 0x200;
glcdc_init_cfg.output.brightness.b = 0x200;

/* Contrast */
glcdc_init_cfg.output.contrast.enable = true;

glcdc_init_cfg.output.contrast.r = 0x80;
glcdc_init_cfg.output.contrast.g = 0x80;
glcdc_init_cfg.output.contrast.b = 0x80;

/* Gamma */
glcdc_init_cfg.output.gamma.enable = true;

glcdc_init_cfg.output.gamma.p_r = (gamma_correction_t *) &g_gamma_table;
glcdc_init_cfg.output.gamma.p_g = (gamma_correction_t *) &g_gamma_table;
glcdc_init_cfg.output.gamma.p_b = (gamma_correction_t *) &g_gamma_table;

/* Dethering */
glcdc_init_cfg.output.dithering.dithering_on = true;
glcdc_init_cfg.output.dithering.dithering_mode = GLCDC_DITHERING_MODE_2X2PATTERN;
glcdc_init_cfg.output.dithering.dithering_pattern_a = GLCDC_DITHERING_PATTERN_11;
glcdc_init_cfg.output.dithering.dithering_pattern_b = GLCDC_DITHERING_PATTERN_00;
glcdc_init_cfg.output.dithering.dithering_pattern_c = GLCDC_DITHERING_PATTERN_10;
glcdc_init_cfg.output.dithering.dithering_pattern_d = GLCDC_DITHERING_PATTERN_01;

/* Detection */
glcdc_init_cfg.detection.vpos_detect = true;
glcdc_init_cfg.detection.gr1uf_detect = true;
glcdc_init_cfg.detection.gr2uf_detect = true;

/* Interrupt */
glcdc_init_cfg.interrupt.vpos_enable = true;
glcdc_init_cfg.interrupt.gr1uf_enable = true;
glcdc_init_cfg.interrupt.gr2uf_enable = true;

glcdc_init_cfg.p_callback = (void (*) (void *)) glcdc_callback;
```

図 9.3　パラメータの設定例(3/3)

10. プロジェクトをインポートする方法

サンプルプログラムは 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 および CS+ 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。

10.1 e² studio での手順

e² studio でご使用になる際は、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。)

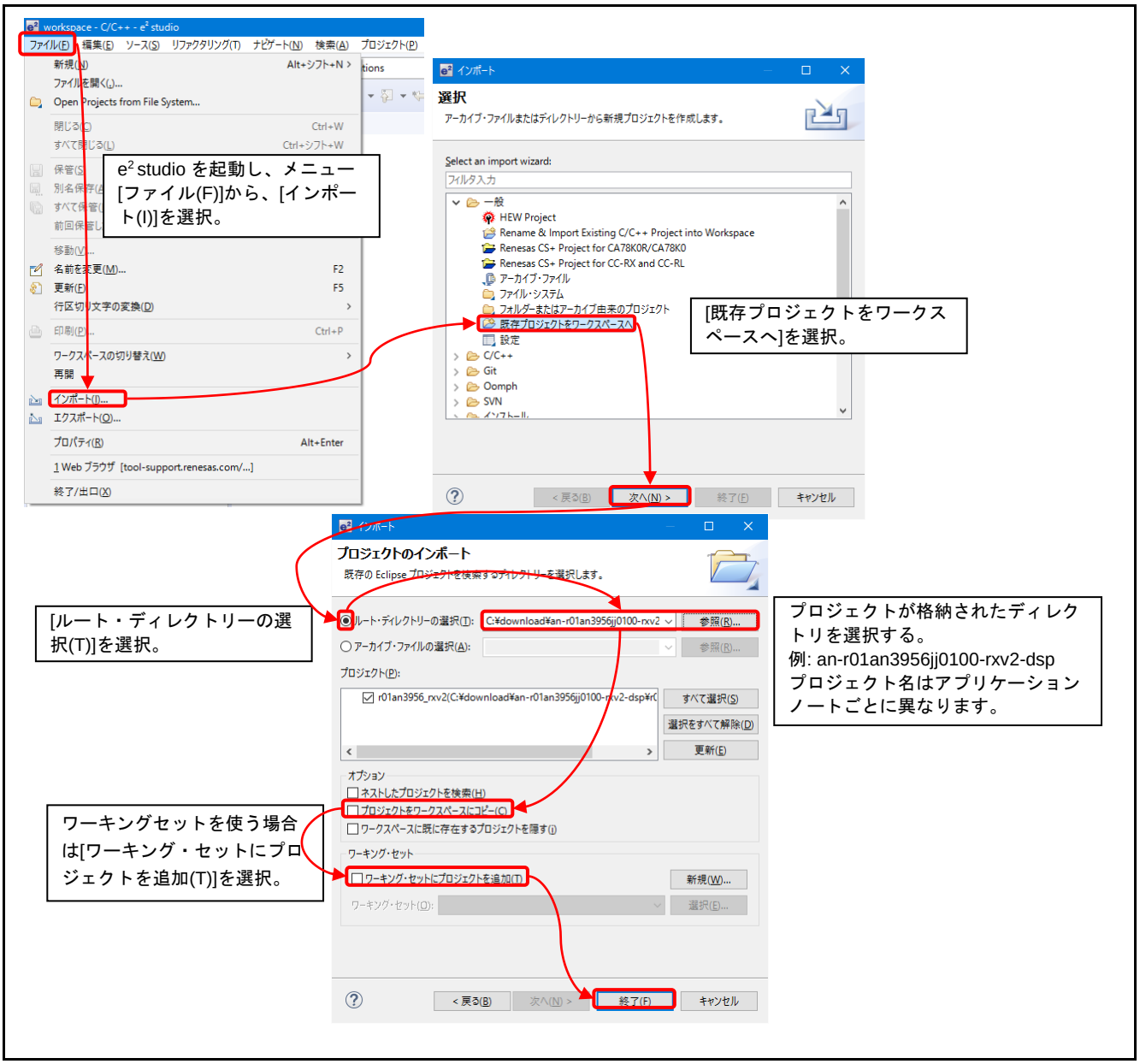


図 10.1　プロジェクトを e² studio にインポートする方法

10.2　CS+での手順

CS+でご使用になる際は、下記の手順で CS+にインポートしてください。

(使用する CS+のバージョンによっては画面が異なる場合があります。)

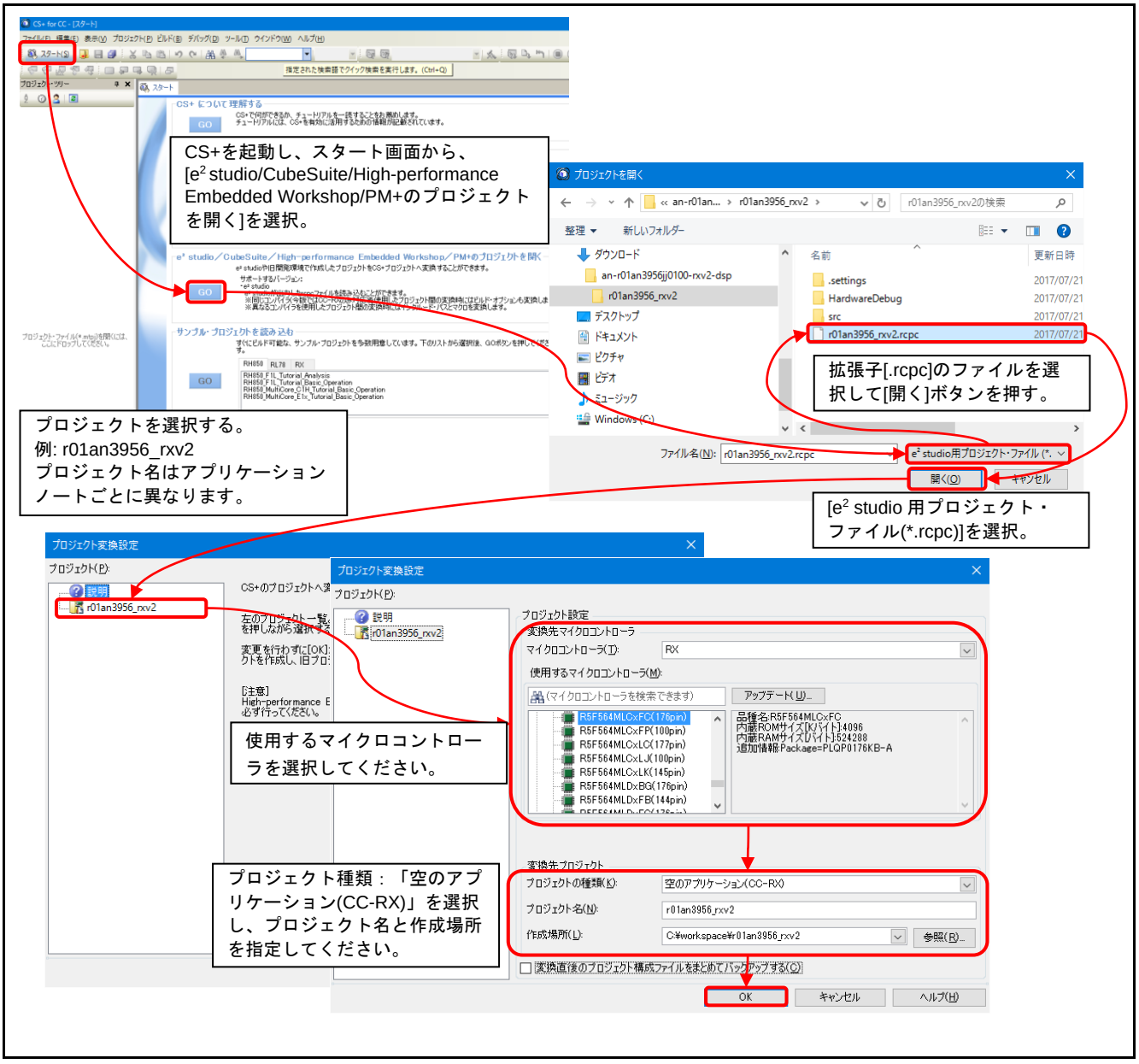


図 10.2　プロジェクトを CS+にインポートする方法

11. 参考ドキュメント

ユーザーズマニュアル：ハードウェア
　RX ファミリ ユーザーズマニュアル ハードウェア編（R01UH0590）
　（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート／テクニカルニュース
　（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境
　RX ファミリコンパイラ CC-RX ユーザーズマニュアル（R20UT3248）
　（最新版をルネサス エレクトロニクスホームページから入手してください。）

ホームページとサポート窓口

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	2017.10.02	-	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. 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/