RX Family

R01AN3509EJ0100
Rev. 1.00
Oct. 20, 2017

Sample Program for Displaying Images on the TFT-LCD Panel Using the Graphic LCD Controller Module Firmware Integration Technology

Introduction

This application note describes the method to display images on the TFT-LCD panel (LCD panel) using the Graphic LCD controller (GLCDC) and the GLCDC FIT module.

Target Devices

- **RX651, RX65N Groups, ROM capacity: 1.5 Mbytes to 2 Mbytes**

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Related Documents

- Firmware Integration Technology User's Manual (R01AN1833)
- Board Support Package Module Using Firmware Integration Technology (R01AN1685)
- Adding Firmware Integration Technology Modules to Projects (R01AN1723)
- Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)
- Renesas e^2 studio Smart Configurator User Guide (R20AN0451)

Contents

1. Specifications

The sample program in this application note performs the following operations using the GLCDC.

- Switching between displayed images: Loaded images can be switched.
- Changing the alpha blending setting: Transparency of the displayed image can be changed.
- Adjusting brightness, contrast, and gamma: Each setting value can be changed.

The LCD panel used in this sample program supports touch input and controls the sample program with touch operation and switches on the RSK.

Figure 1.1 shows the Operation Overview.

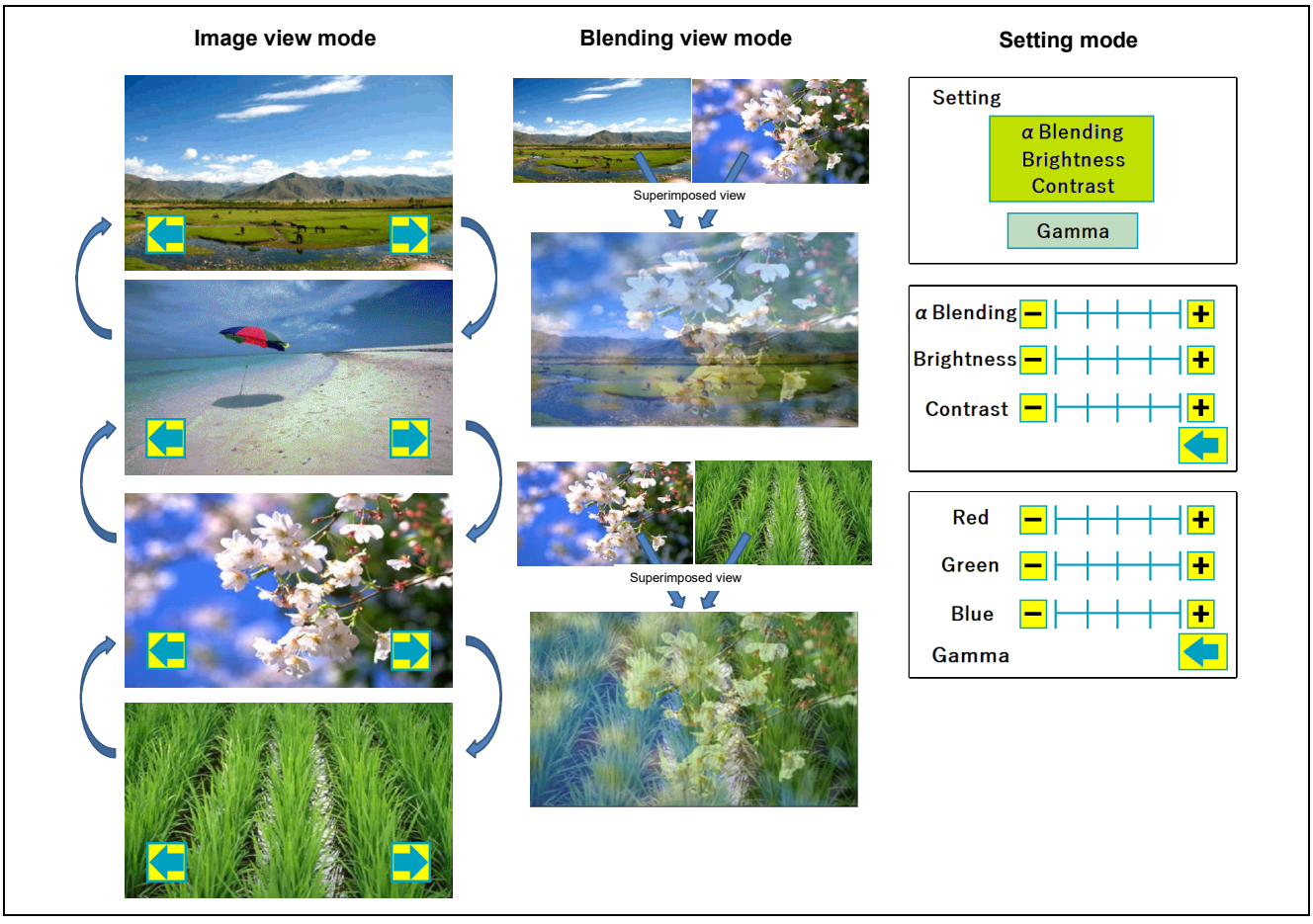


Figure 1.1 Operation Overview

The sample program operates in the following three screen modes.

- **Image view mode**
  Images are displayed one by one. An image is switched to the next image by touching the arrow on the panel.

- **Blending view mode**
  An image displayed is superimposed on another image (an image on upper layer is transparent). An image is superimposed on the next image by touching the panel. The alpha value can be specified in “setting mode”.

- **Setting mode**
  Brightness, contrast, and gamma in image view mode and blending view mode can be adjusted. Each setting can be changed in five levels. The result of the alpha value adjusted can be seen only in blending view mode. The image displayed in image view mode is used for the background in setting mode.

The peripheral modules listed in Table 1.1 are used to achieve the features above.

Table 1.1 Peripheral Modules Used and Their Applications

Peripheral Module	Application
GLCDC	Display and control of screens
Channel 1 of RSPI	Communication with the SerialFlash
Channel 7 of SCI (simple I^2C mode)	Communication with the touch controller of the LCD panel
Channel 0 of CMT	Timer for touch detection period

1.1 Structure of Displayed Images

The GLCDC has the information of three-layer structure; Graphic 1, Graphic 2, and Background screens. The GLCDC process images according to the information of these screens and outputs one image on the LCD panel. Images for Graphic 1 and Graphic 2 screens can be specified and the displayed layer can be dynamically switched. Also, with the alpha blending feature, specific colors can be transparent and a superimposed image can be generated. The sample program uses this feature to output the superimposed image shown in Figure 1.2.

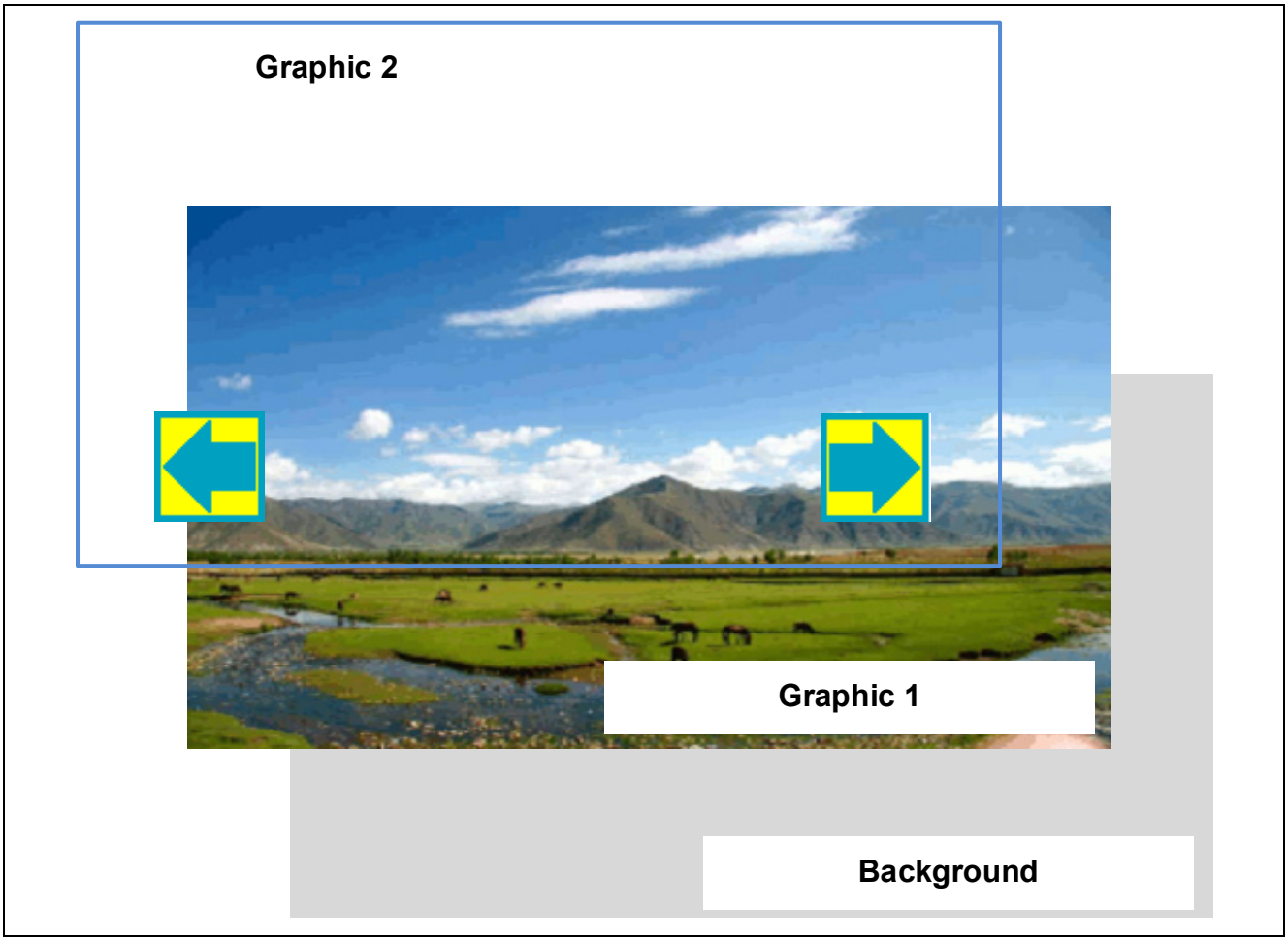


Figure 1.2 Structure of Displayed Images

1.2 Image Data Used in the Sample Program

The sample program in this application note uses images listed in Table 1.2. The image data are basically stored in the SerialFlash and are loaded at start-up or each time when necessary.

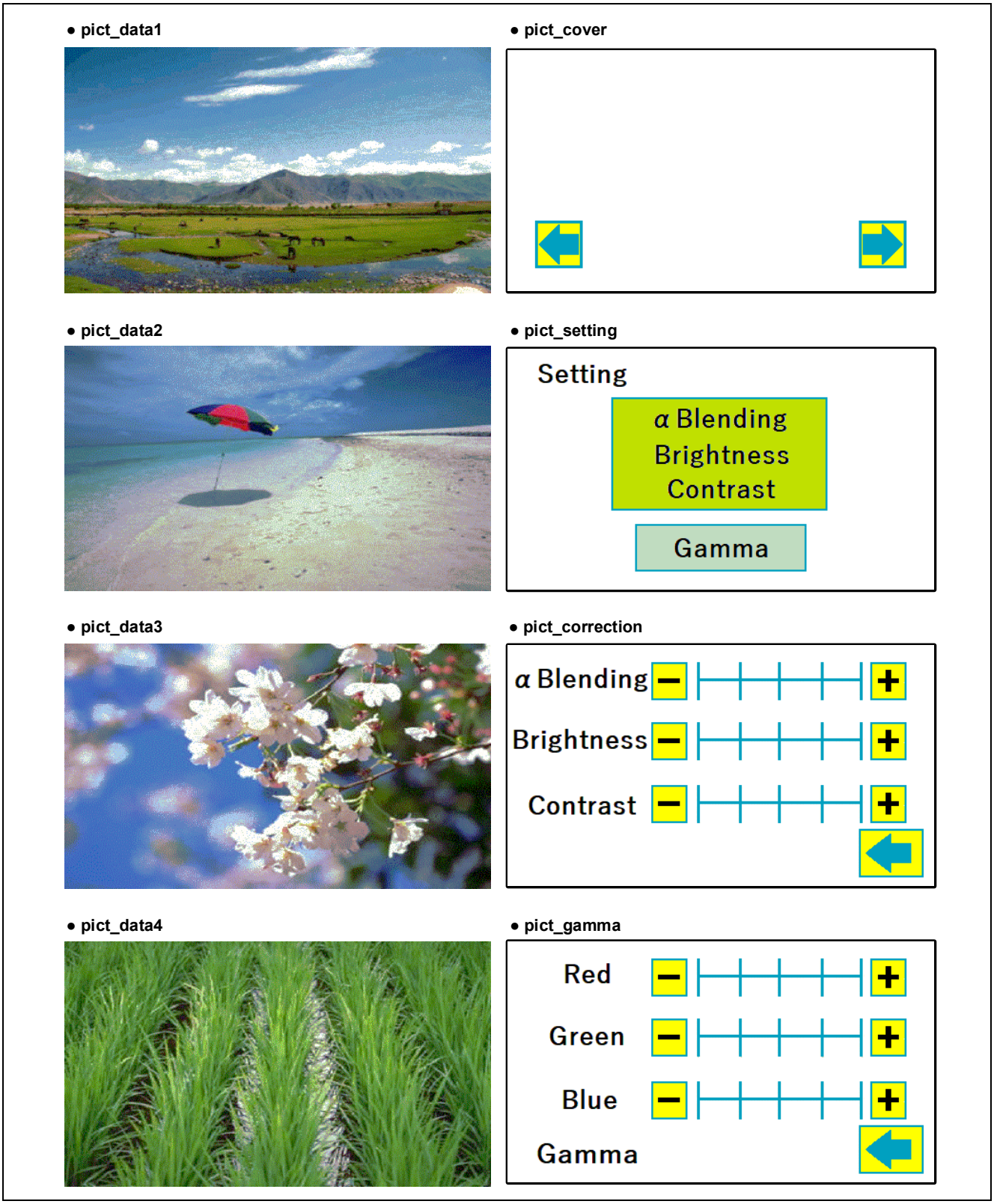


Figure 1.3 Image Data Used in the Sample Program

Table 1.2 Information of Images Used in the Sample Program

File Name	Image Size (byte)	Image Format	Description
pict_data1.bmp	114,422	BMP (256 colors)	Image 1 for displaying
pict_data2.bmp	114,422		Image 2 for displaying
pict_data3.bmp	114,422		Image 3 for displaying
pict_data4.bmp	114,422		Image 4 for displaying
pict_cover.bmp	114,422		Superimposed image (switch button)
pict_setting.bmp	114,422		Superimposed image (screen for setting selection)
pict_correction.bmp	114,422		Superimposed image (screen for brightness/contrast setting)
pict_gamma.bmp	114,422		Superimposed image (screen for gamma setting)
Total size	915,376	—	—

The image format is BMP (Windows Bitmap Image).

The image size is 448 × 253 with 256 colors (8 bits).

The horizontal width of images displayed on the GLCDC needs to satisfy the following condition: “byte size per pixel” × “horizontal width (pixel) of image” is divisible by 64 bytes. The size per pixel is 8 bits for 256-color BMP image format. Thus the horizontal width of the image is 448 pixels (448/64 bytes = 7) in the sample program.

1.3 Touch Control

The LCD panel used in the sample program has the touch controller. The touch input information can be obtained by communicating with this controller. The communication is made between RX65N as the master and the touch controller as the slave in I^2C protocol.

The sample program obtains the touch input information by communicating with the touch controller regularly and determines whether a button on the screen is pressed.

According to the button pressed, the sample program switches between displayed images, or adjusts alpha value, brightness, contrast, or gamma.

2. Operation Confirmation Environment

The operation of the sample program in this application note has been confirmed under the following conditions.

Table 2.1 Operation Confirmation Conditions

Item	Contents
MCU used	R5F565NEDDFC (RX65N Group)
Operating frequency	● Main clock: 24 MHz ● PLL: 240 MHz (main clock × 1/1 ×10) ● System clock (ICLK): 120 MHz (PLL × 1/2) ● Peripheral module clock A (PCLKA): 120 MHz (PLL × 1/2) ● Peripheral module clock B (PCLKB): 60 MHz (PLL × 1/4) ● LCD panel clock (LCD_CLK): 10 MHz (PLL × 1/24)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics e² studio Version 6.0.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.07.00
	Compiler option -lang = c99 Project for displaying images “RIMAGE (0x00800000)” is added to sections. Project for writing the SerialFlash “IMAGE (0xFFE80000)” is added to sections. The binary option is added to the linker (see 5.2.1 Project Properties for details).
iodefine.h version	Version 2.0
Endian	Little endian, big endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample program version	Version 1.00
Board used	Renesas Starter Kit+ for RX65N-2MB (RSK) (product No.: RTK50565N2SXXXXXBE)

3. FIT Modules Used in the Sample Program

The following FIT modules are used in this application note. Please refer to the following documents as reference.

- RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)
- RX Family Graphic LCD Controller Module Using Firmware Integration Technology (R01AN3609)
- RX Family RSPI Clock Synchronous Single Master Control Module Firmware Integration Technology (R01AN1914)
- RX Family Clock Synchronous Control Module for Serial Flash Memory Access Firmware Integration Technology (R01AN2662)
- RX Family Simple I^2C Module Using Firmware Integration Technology (R01AN1691)
- RX Family CMT Module Using Firmware Integration Technology (R01AN1856)

Please use the latest version when it is available. Visit the Renesas Electronics website to check and obtain the latest version.

4. Executing the Projects

This section describes the methods to execute the sample program. The sample program includes the following two projects.

- glcdc_main_rx65n
- serialflash_writer_rx65n

The main project in this application note is “glcdc_main_rx65n” for displaying images. Image data need to be loaded into the SerialFlash beforehand for displaying them on the LCD panel.

Another project is “serialflash_writer_rx65n” for writing image data into the SerialFlash. Image data can be written into the SerialFlash by executing this project. Note that the project does not disable the write-protection for the SerialFlash. If you try to write data into the SerialFlash with the protection enabled, the write operation would fail. Also, when writing image data, the project erases all data in the SerialFlash.

On the other hand, the glcdc_main_rx65n project only reads the image data, thus written data would not be erased by this project.

The methods to execute the sample program are described in the following order.

1. Executing the project for writing the SerialFlash (serialflash_writer_rx65n)

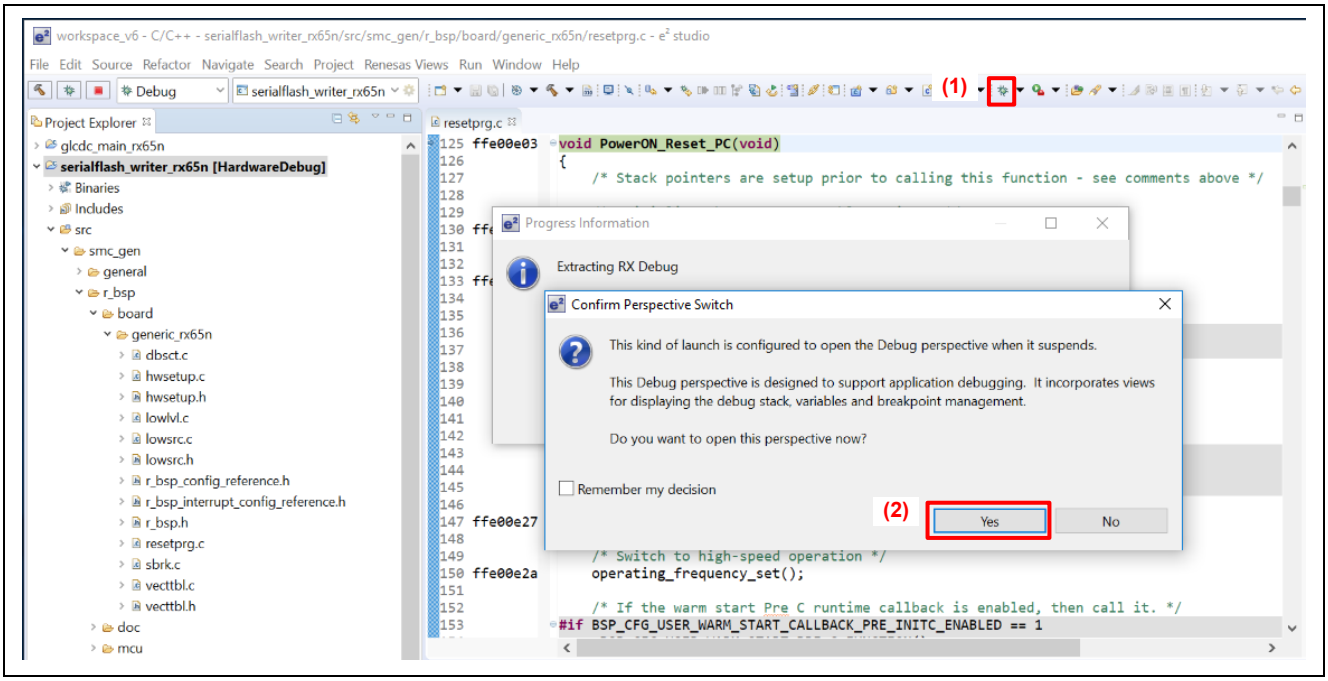
2. Executing the project for displaying images (glcdc_main_rx65n)

4.1 Executing the Project for Writing the SerialFlash

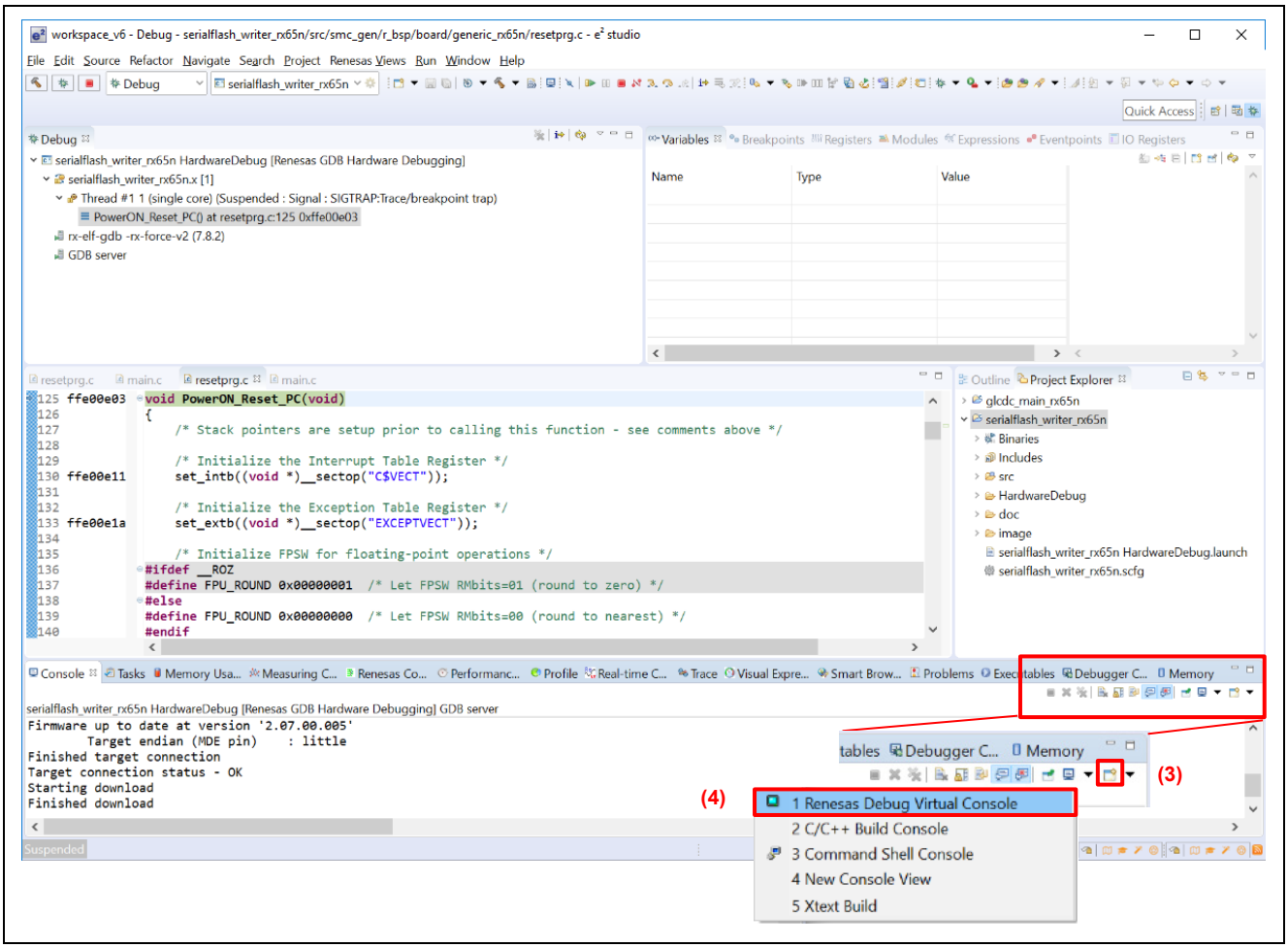
After the e^2 studio is started, import and execute the serialflash_writer_rx65n project with the debugger connected. The status of operation for writing the image data is output to the Renesas Debug Virtual Console (for the case of e^2 studio). When the images have been successfully written, LED0 turns on. If an error occurs during write operation, LED3 turns on.

The following describes the procedure to execute the project in the e^2 studio. (Windows or dialogs of e^2 studio may differ depending on the e^2 studio version used.)

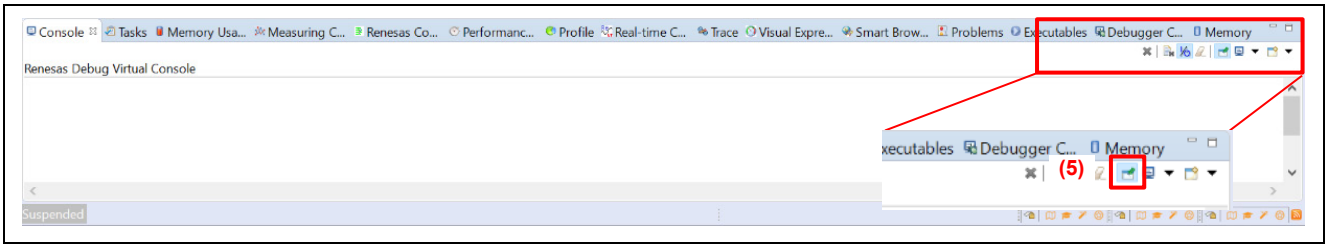
1. Click the Debug icon (**1**) at upper side of the window. The debugger is launched and the Progress Information window is displayed. When the Confirm Perspective Switch dialog appears, click Yes (**2**).



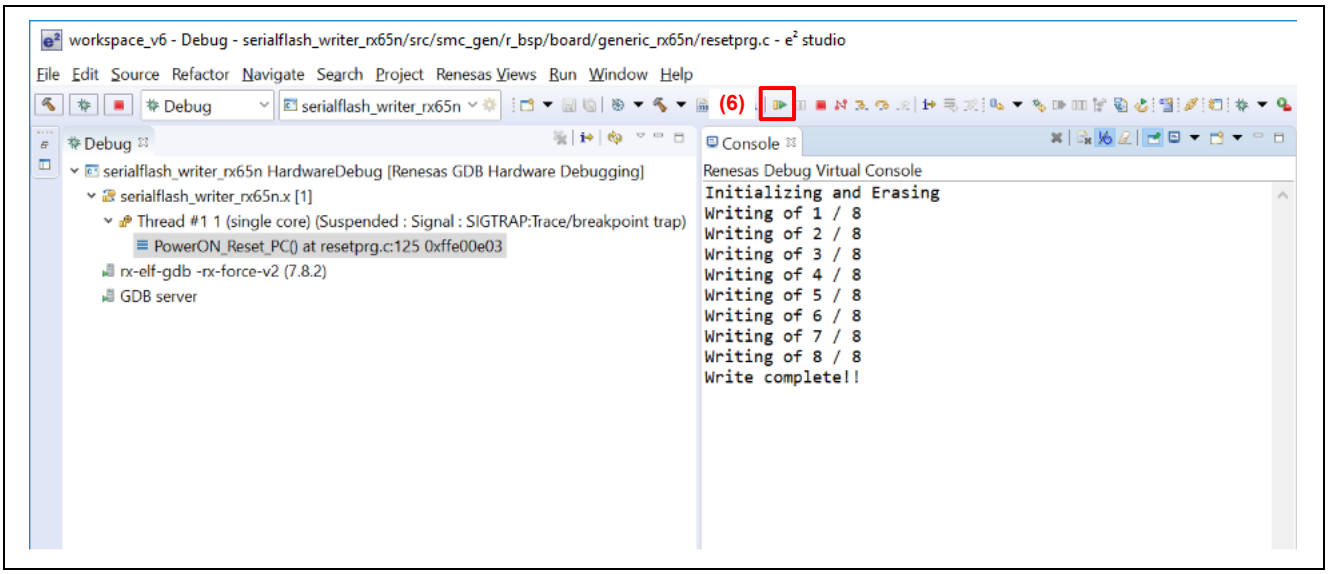
2. The window is switched to the Debug perspective. Click the Open Console icon (**3**) on the right side of the Console window. Select *1. Renesas Debug Virtual Console* (**4**) on the sub menu.



3. When the console is switched to the Renesas Debug Virtual Console, click the Pin Console icon (**5**).



4. Click the Resume icon (**6**) to execute the program. Then messages are displayed in the Renesas Debug Virtual Console. Write operation is completed when “Write complete!!” is displayed and LED0 on the RSK turns on. Disconnect the debugger to finish.

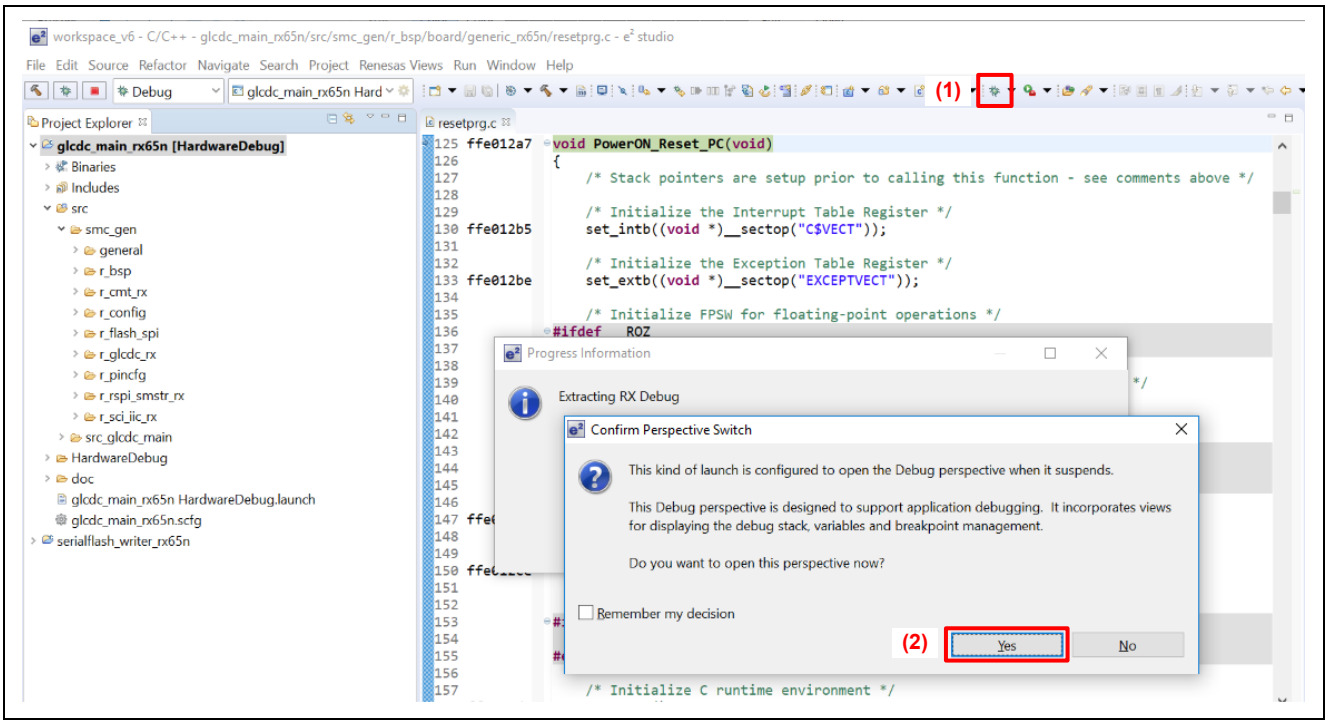


4.2 Executing the Project for Displaying Images

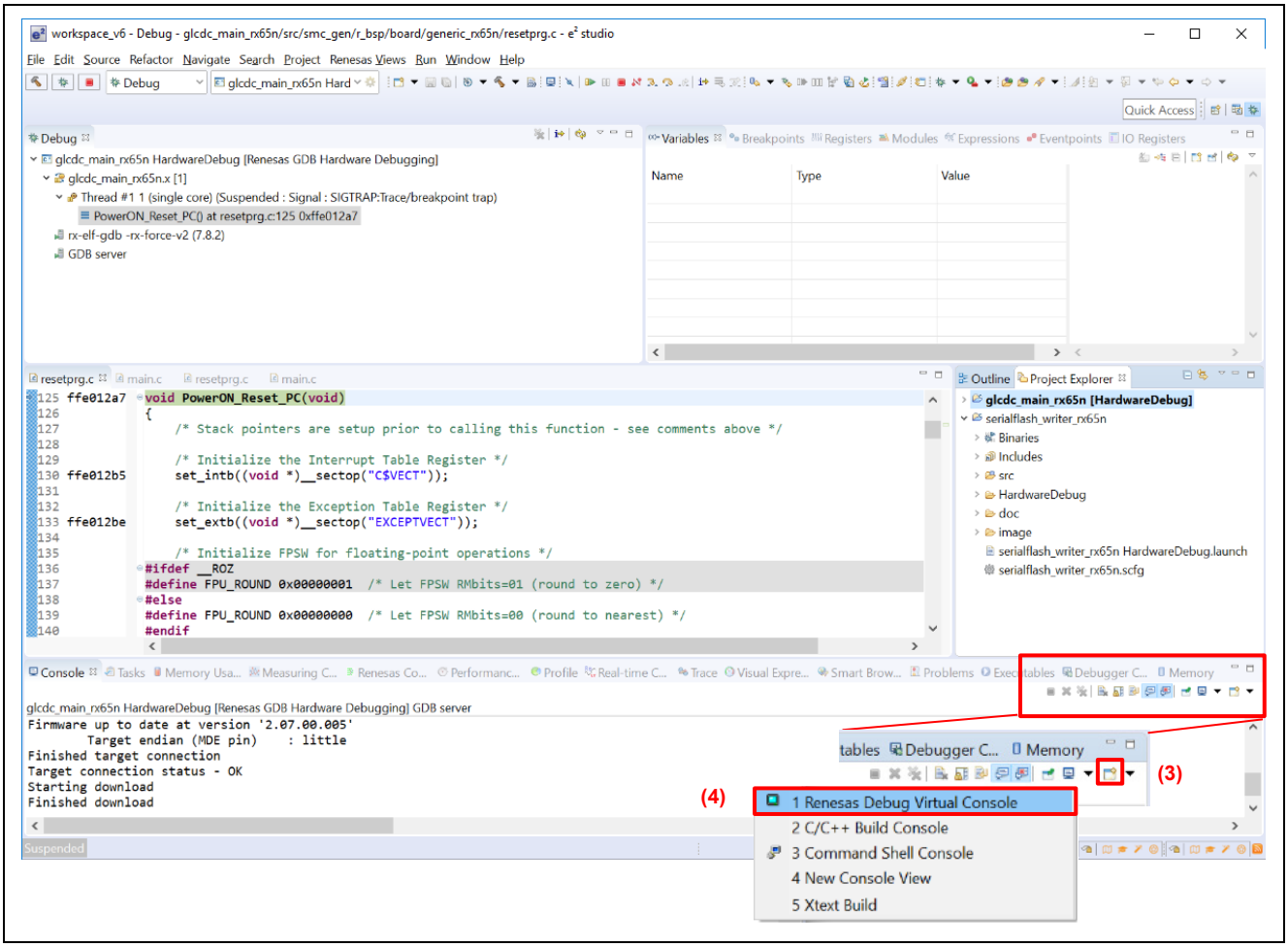
Import and execute the glcdc_main_rx65n project with the debugger connected. If the project is executed while the image data has not been written to the SerialFlash, an error is output to the Renesas Debug Virtual Console (for the case of e^2 studio) and LED3 turns on.

The following describes the procedure to execute the project in the e^2 studio. (Windows or dialogs of e^2 studio may differ depending on the e^2 studio version used.)

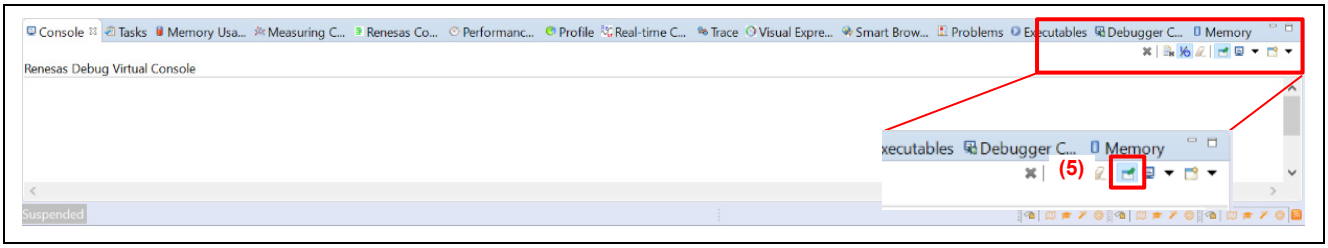
1. Click the Debug icon (**1**) at upper side of the window. The debugger is launched and the Progress Information is displayed. When the Confirm Perspective Switch dialog appears, click Yes (**2**).



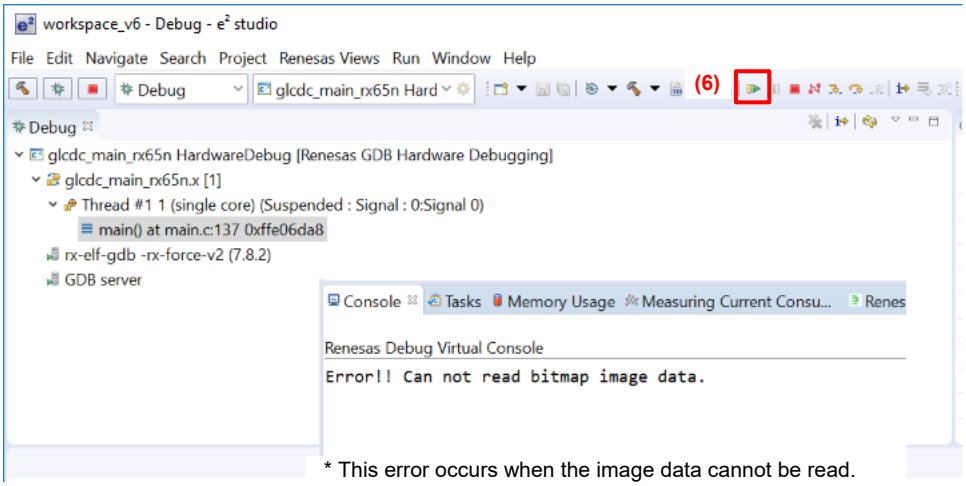
2. The window is switched to the Debug perspective. Click the Open Console icon (**3**) on the right side of the Console window. Select *1. Renesas Debug Virtual Console* (**4**) on the sub menu.



3. When the console is switched to the Renesas Debug Virtual Console, click the Pin Console icon (**5**).



4. Click the Resume icon (**6**) to execute the program. The image data is read from the SerialFlash and displayed on the LCD panel. If the image data cannot be read from the SerialFlash, the message is displayed in the Renesas Debug Virtual Console and LED3 on the RSK turns on.



* This error occurs when the image data cannot be read.

5. Changed Information in the Projects

In the projects of the sample program, settings in the configuration files, source folders and the projects have been changed for each FIT module. The details are described in the sub-sections.

Refer to the information provided in this section when setting up a new project. When using the imported project, go to 6. Hardware.

5.1 Modifying the Configuration File

The following describes the changed settings in the configuration file. Changed items are the same in the glcdc_main_rx65n project for displaying images and the serialflash_writer_rx65n project for writing the SerialFlash. For items of the configuration options and their settings, refer to the manual of each FIT module. For the configuration options which are not described here, their default settings are used.

(1) Modifying settings in the RSPI clock synchronous single master control module

Specify as follows to enable Channel 1.

File: 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) Modifying settings in the clock synchronous control module for serial flash memory access

Specify as follows to use Channel 1 for communication with DEV0.

File: 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   */
```

Specify the communication rate for communicating with DEV0 as follows.

File: 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.      */
```

Specify the port number to assign the #SS pin for DEV0 as follows.

File: 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) Modifying settings in the simple I²C module

Specify as follows to enable channel 7.

File: 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 Configuring the Project for Writing the SerialFlash

5.2.1 Project Properties

The following settings are specified in the project properties.

- Section “IMAGE” is added at address 0xFFE80000
- “binary” option is added to the compile option (see below).

-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)

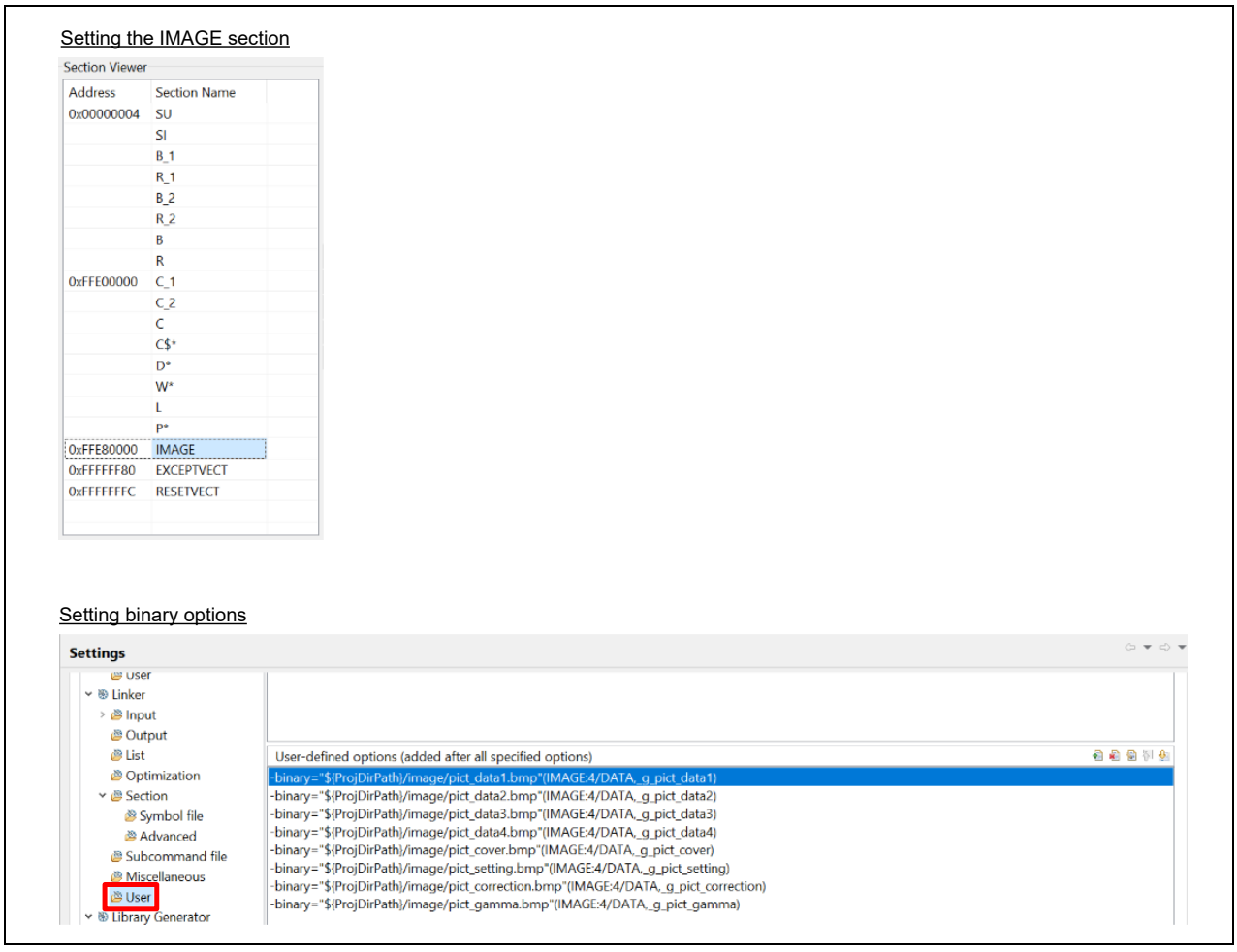


Figure 5.1 Settings of the IMAGE Section and the binary Option

5.2.2 Placing the Image Files

The image files are placed in the root directory of the project. The name of the folder to store the image files is “image” in lower case.

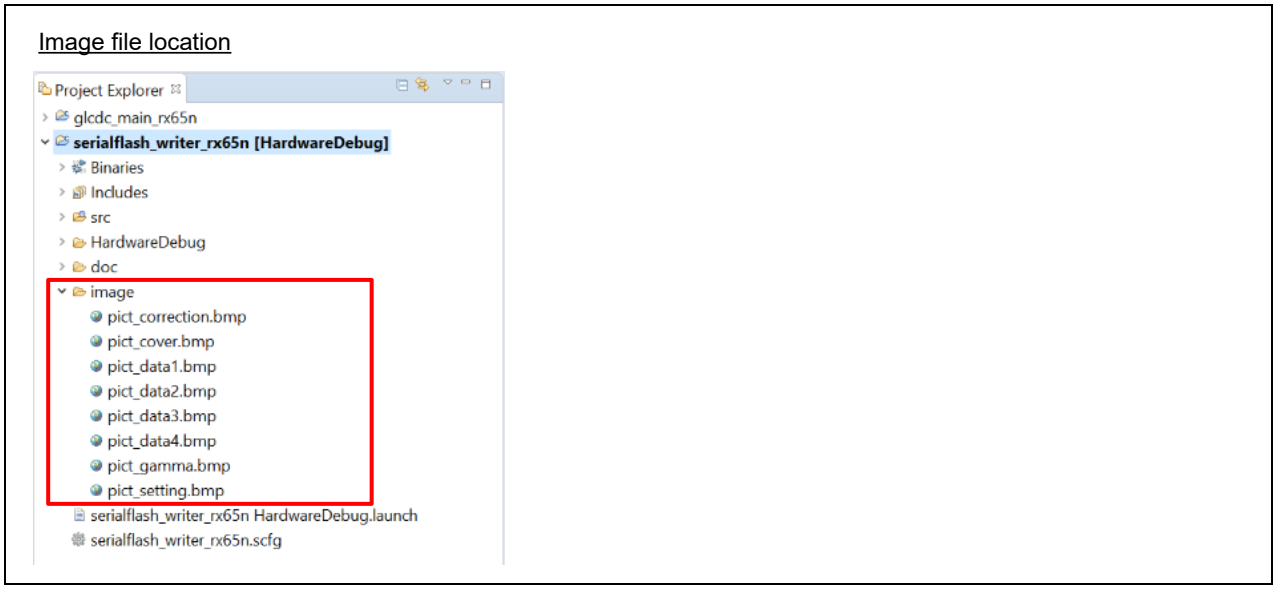


Figure 5.2 Placing the Image Files

5.3 Setting of the Project for Displaying Images

5.3.1 Project Properties

The following setting is specified in the project properties.

- Section “RIMAGE” is added at address 0x00800000.

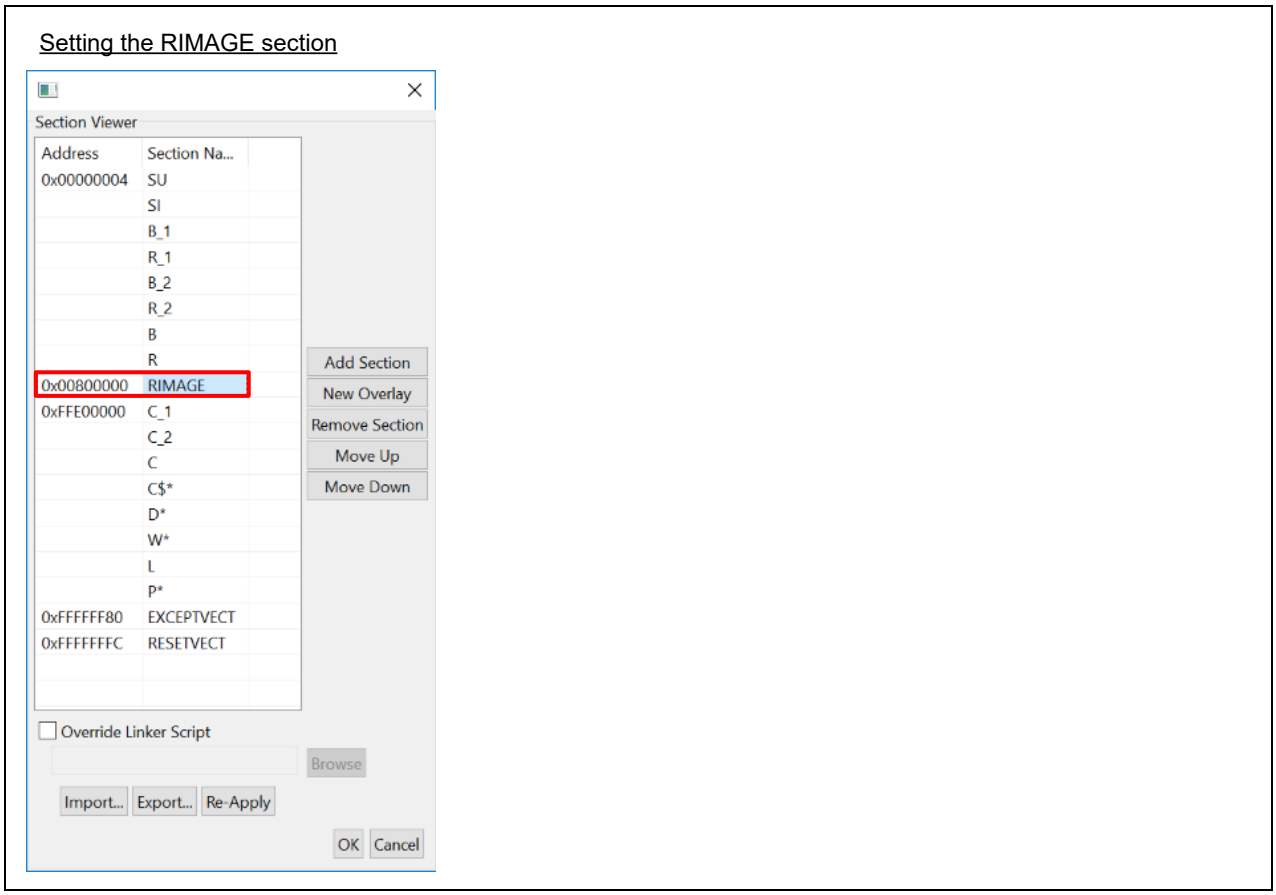


Figure 5.3 Setting the RIMAGE Section

5.4 Preventing a Build Error in CS+

5.4.1 Deleting Unnecessary Folders

Some files in the clock synchronous control module for serial flash memory access have the same name as files in other folders in the project. In CS+, if multiple files have the same name in the project, a build error occurs. Therefore, in the sample program, folders shown in Figure 5.4 have been deleted in the clock synchronous control module for serial flash memory access.

Folders used (folders not deleted) are as follows:

- using_iodefine
- rx_fit_rspi

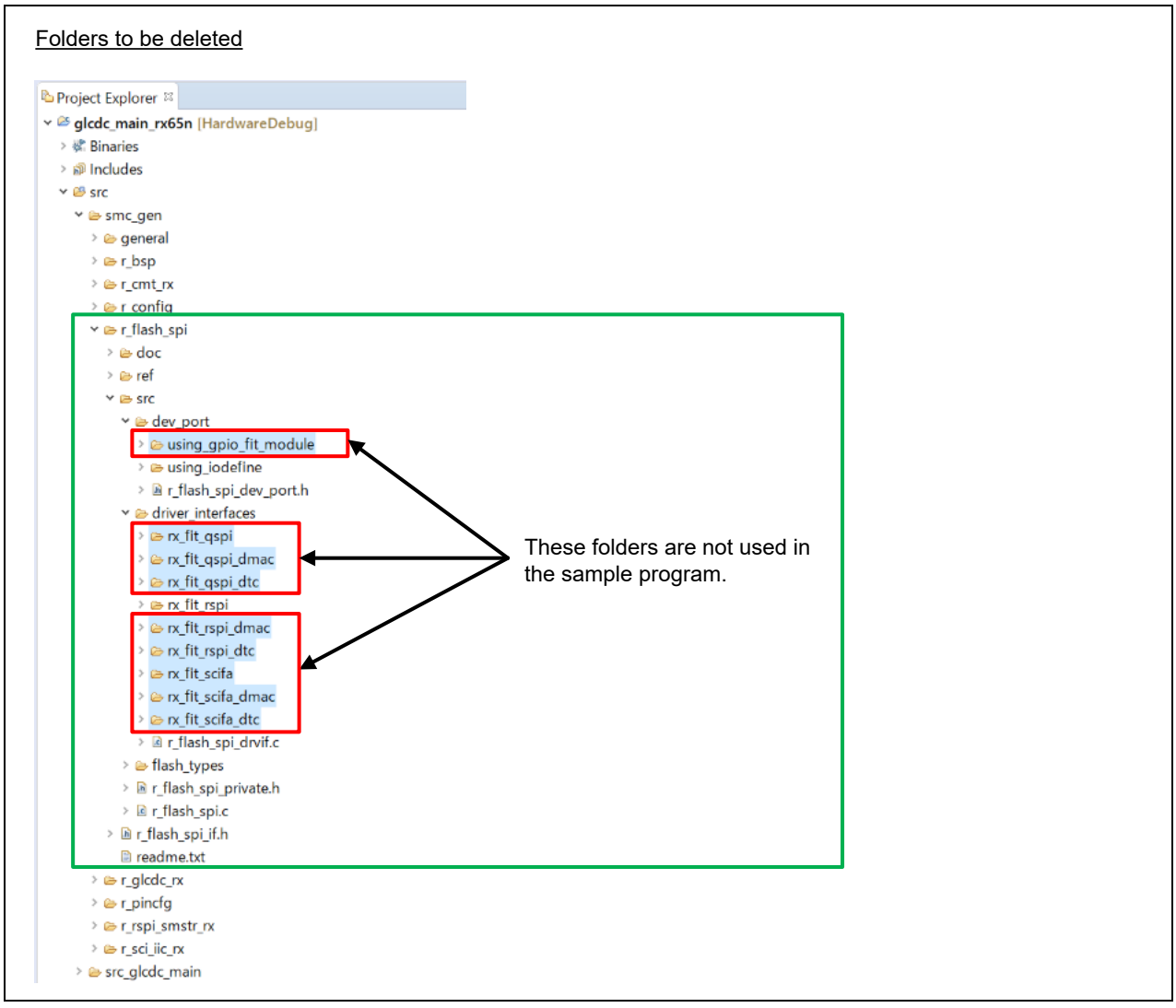


Figure 5.4 Folders to be Deleted

6. Hardware

6.1 Hardware Configuration and Jumper Setting

The devices used in this application note are listed in Table 6.1 with jumper settings.

To use the LCD panel, jumpers have to be set. Set jumpers listed in Table 6.1 before executing the sample program.

Table 6.1 Devices Used and Jumper Setting

Device	Product Information	Jumper Setting
LCD panel	Vender: Newhaven Display Part number: NHD-4.3-480272EF-ATXL#-CTP Display size: 480 × 272 Built-in touch controller	<SW4> Pin 3: OFF Pin 4: ON (OnBoard_TFT Available)
SerialFlash	Vender: Micron Technology Part number: MX25L3233FM2I-08G Memory size: 32 Mbits (4 Mbytes)	No setting required.

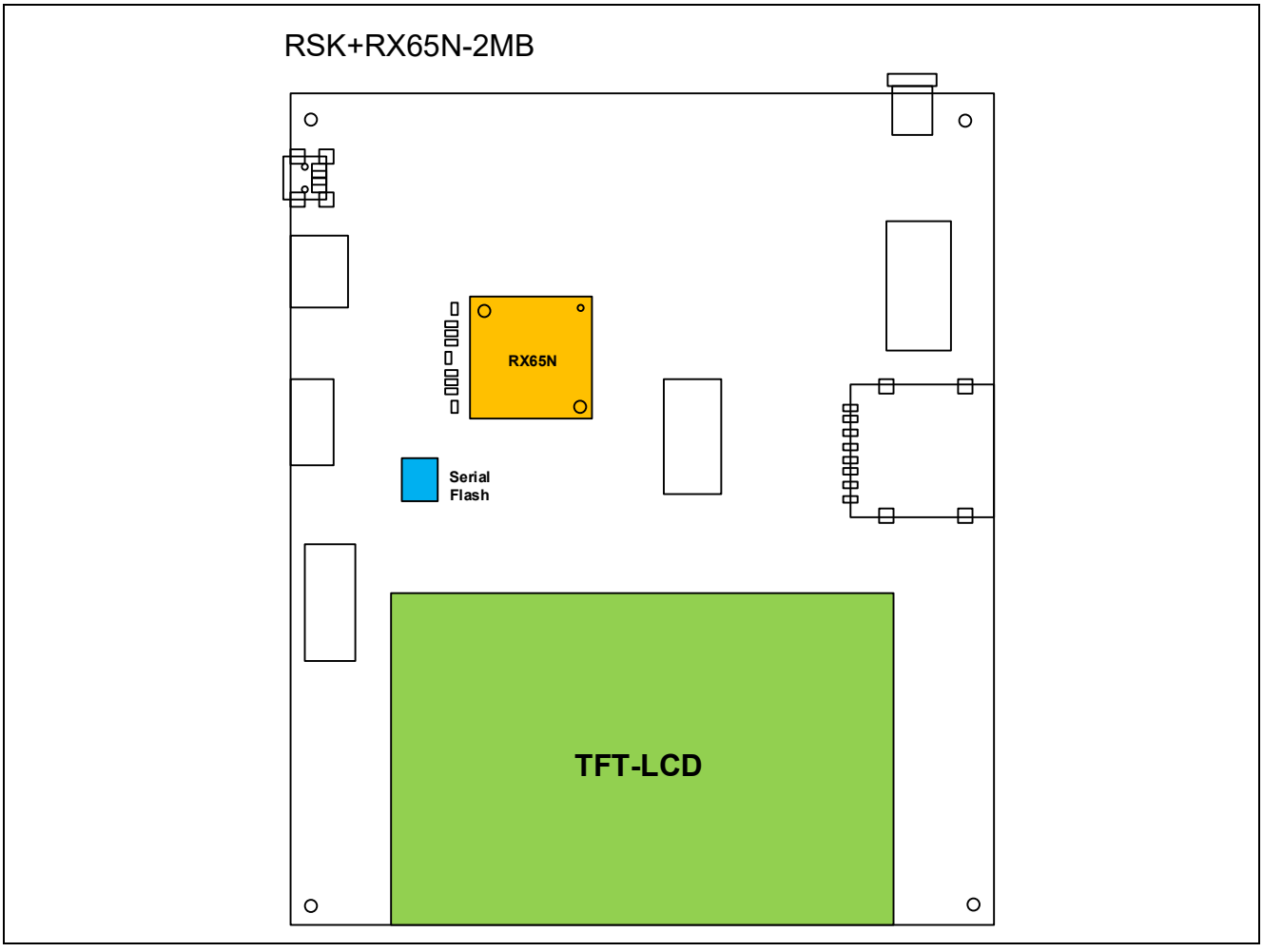


Figure 6.1 RSKRX65N-2MB

Table 6.2 lists Pins Used and Their Functions.

The pins listed here assume a product with176 pins is used. When a product with less than 176 pins is used, select pins appropriate to the product used.

Table 6.2 Pins Used and Their Functions

Device Connected	Pin Name	I/O	Description
LCD panel (GLCDC)	PB5/LCD_CLK-B	Output	Outputs the panel clock
	PB4/LCD_TCON 0-B	Output	Outputs the synchronizing signal (VSYNC)
	PB2/LCD_TCON 2-B	Output	Outputs the synchronizing signal (HSYNC)
	PB1/LCD_TCON 3-B	Output	Outputs the synchronizing signal (DE)
	PB0/LCD_DATA 0-B	Output	Outputs the LCD signal R[3]
	PA7/LCD_DATA 1-B	Output	Outputs the LCD signal R[4]
	PA6/LCD_DATA 2-B	Output	Outputs the LCD signal R[5]
	PA5/LCD_DATA 3-B	Output	Outputs the LCD signal R[6]
	PA4/LCD_DATA 4-B	Output	Outputs the LCD signal R[7]
	PA3/LCD_DATA 5-B	Output	Outputs the LCD signal G[2]
	PA2/LCD_DATA 6-B	Output	Outputs the LCD signal G[3]
	PA1/LCD_DATA 7-B	Output	Outputs the LCD signal G[4]
	PA0/LCD_DATA 8-B	Output	Outputs the LCD signal G[5]
	PE7/LCD_DATA 9-B	Output	Outputs the LCD signal G[6]
	PE6/LCD_DATA 10-B	Output	Outputs the LCD signal G[7]
	PE5/LCD_DATA 11-B	Output	Outputs the LCD signal B[3]
	PE4/LCD_DATA 12-B	Output	Outputs the LCD signal B[4]
	PE3/LCD_DATA 13-B	Output	Outputs the LCD signal B[5]
	PE2/LCD_DATA 14-B	Output	Outputs the LCD signal B[6]
	PE1/LCD_DATA 15-B	Output	Outputs the LCD signal B[7]
	PB7	Output	Backlight
	P97	Output	Panel ON
LCD touch panel controller (SCI-I^{2}C)	P92/SSCL7	I/O	Inputs/outputs the clock
	P90/SSDA7	I/O	Inputs/outputs data
	P42	Input	Trigger input pin
SerialFlash (RSPI)	P31	Output	Slave select
	P30/MISOB-A	I/O	Inputs/outputs data output from the slave
	P27/RSPCKB-A	I/O	Inputs/outputs the clock
	P26/MOSIB-A	I/O	Inputs/outputs data output from the master
Switch	P03	Input	SW1
	P05	Input	SW2
	P07	Input	SW3

7. Software (Project for Writing the SerialFlash)

7.1 Operation Overview

7.1.1 Settings for the Peripheral Module and Devices

The RSPI is used for writing the SerialFlash. Table 7.1 lists Settings for the RSPI. For details on the RSPI settings, refer to the manual for the RSPI clock synchronous single master control module.

Table 7.1 Settings for the RSPI

Item	Setting	Remarks
Channel used	Channel 1	Used for communicating with the SerialFlash. Write only
Communication method	Clock synchronous mode	
Transfer speed (bit rate)	30 Mbps	

7.1.2 Writing the SerialFlash

In the sample program, the image data are placed in the ROM of the RX65N for the project to write them to the SerialFlash. Table 7.2 lists the address map of image data in RX65N and the SerialFlash. The total size is 1,072,304 bytes. For placing the image data in the ROM, refer to 5.2 Configuring the Project for Writing the SerialFlash.

Table 7.2 Address Map of Image Data in RX65N and the SerialFlash (Project for Writing the SerialFlash)

Device	Image Data	Start Address
RX65N (Section: 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
	End address	0xFFF5F7BD
SerialFlash (to write data)	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 File Composition

Table 7.3 lists the Files Used in the Sample Program. Files generated by the FIT module and files generated by the integrated development environment are not included in the table.

Table 7.3 Files Used in the Sample Program

Fine Name	Outline	Remarks
main.c	Main processing	
r_serial_flash_write.c	RSPI initialization, Processing for writing data to the SerialFlash	
r_serial_flash_write.h	Header file for r_serial_flash_write.c	

7.3 Option Setting Memory

Table 7.4 lists the state of the option setting memory used in the sample program. Please specify values appropriate to your system as required.

Table 7.4 Option Setting Memory Used in the Sample Program

Symbol	Address	Setting Value	Description
OFS0	FE7F 5D04h to FE7F 5D07h	FFFF FFFFh	WDT/IWDT stopped after a reset
OFS1	FE7F 5D08h to FE7F 5D0Bh	FFFF FFFFh	Voltage monitor 0 reset disabled after a reset HOCO oscillation disabled after a reset
MDE	FE7F 5D00h to FE7F 5D03h	FFFF FFFFh	Little endian

7.4 Constants

Table 7.5 lists the Constants Used in the Sample Program.

Table 7.5 Constants Used in the Sample Program

Constant	Setting Value	Description
LED_ON	(0)	LED turned on
LED_OFF	(1)	LED turned off
LED0	(PORT7.PODR.BIT.B3)	PODR register bit for LED0(P73)
LED3	(PORTG.PODR.BIT.B5)	PODR register bit for LED3(PG5)
LED0_PDR	(PORT7.PDR.BIT.B3)	PDR register bit for LED0(P73)
LED3_PDR	(PORTG.PDR.BIT.B5)	PDR register bit for LED3(PG5)
IMAGE_NUM	(8)	Number of images to be written in the SerialFlash
BMP_SIZE	(114422)	Bit map file size (byte)

7.5 Variables

Table 7.6 lists the Global Variables.

Table 7.6 Global Variables

Type	Variable Name	Description
extern uint8_t	g_pict_data1[]	Pointer to image 1
extern uint8_t	g_pict_data2[]	Pointer to image 2
extern uint8_t	g_pict_data3[]	Pointer to image 3
extern uint8_t	g_pict_data4[]	Pointer to image 4
extern uint8_t	g_pict_cover[]	Pointer to the superimposed image (switch button)
extern uint8_t	g_pict_setting[]	Pointer to the superimposed image (screen for setting selection)
extern uint8_t	g_pict_correction[]	Pointer to the superimposed image (screen for brightness and contrast setting)
extern uint8_t	g_pict_gamma[]	Pointer to the superimposed image (screen for gamma setting)
uint8_t *	gp_pict_table[IMAGE_NUM]	Pointer table for each image
uint32_t	g_pict_addr_table[IMAGE_NUM]	SerialFlash image allocation (address) table

7.6 Functions

Table 7.7 and Table 7.8 list functions.

Table 7.7 Function (main.c)

Function Name	Outline
main	Main processing

Table 7.8 Function (r_serial_flash_write.c)

Function Name	Outline
serialflash_write_initialize	Initialization for SerialFlash communication
data_write	Writing data to the SerialFlash

7.7 Function Specifications

This section describes function specifications.

main	
Outline	Main processing
Header	None
Declaration	void main(void)
Description	Writes the image data to the SerialFlash.
Parameters	None
Return Values	None

serialflash_write_initialize	
Outline	Initialization for SerialFlash communication
Header	r_serial_flash_write.h
Declaration	flash_spi_status_t serialflash_write_initialize(void)
Description	Initializes communication with the SerialFlash and erases all data in the SerialFlash to make it ready for writing.
Parameters	None
Return Values	FLASH_SPI_SUCCESS: Processing completed successfully FLASH_SPI_ERR_HARD: Hardware error FLASH_SPI_ERR_OTHER: Other errors

data_write	
Outline	Writing data to the SerialFlash
Header	r_serial_flash_write.h
Declaration	flash_spi_status_t data_write(uint8_t *p_src_addr, uint32_t dest_addr, uint32_t data_size)
Description	Writes the specified image data to the specified address in the SerialFlash.
Parameters	uint8_t *p_src_addr: Address of the image data to be written uint32_t dest_addr: Address to place the image data in the SerialFlash uint32_t data_size: Size of the image data to be written
Return Values	FLASH_SPI_SUCCESS: Processing completed successfully FLASH_SPI_ERR_PARAM: Parameter error FLASH_SPI_ERR_HARD: Hardware error FLASH_SPI_ERR_OTHER: Other errors

7.8 Flowcharts

7.8.1 Main Processing

Figure 7.1 shows the flowchart of main processing.

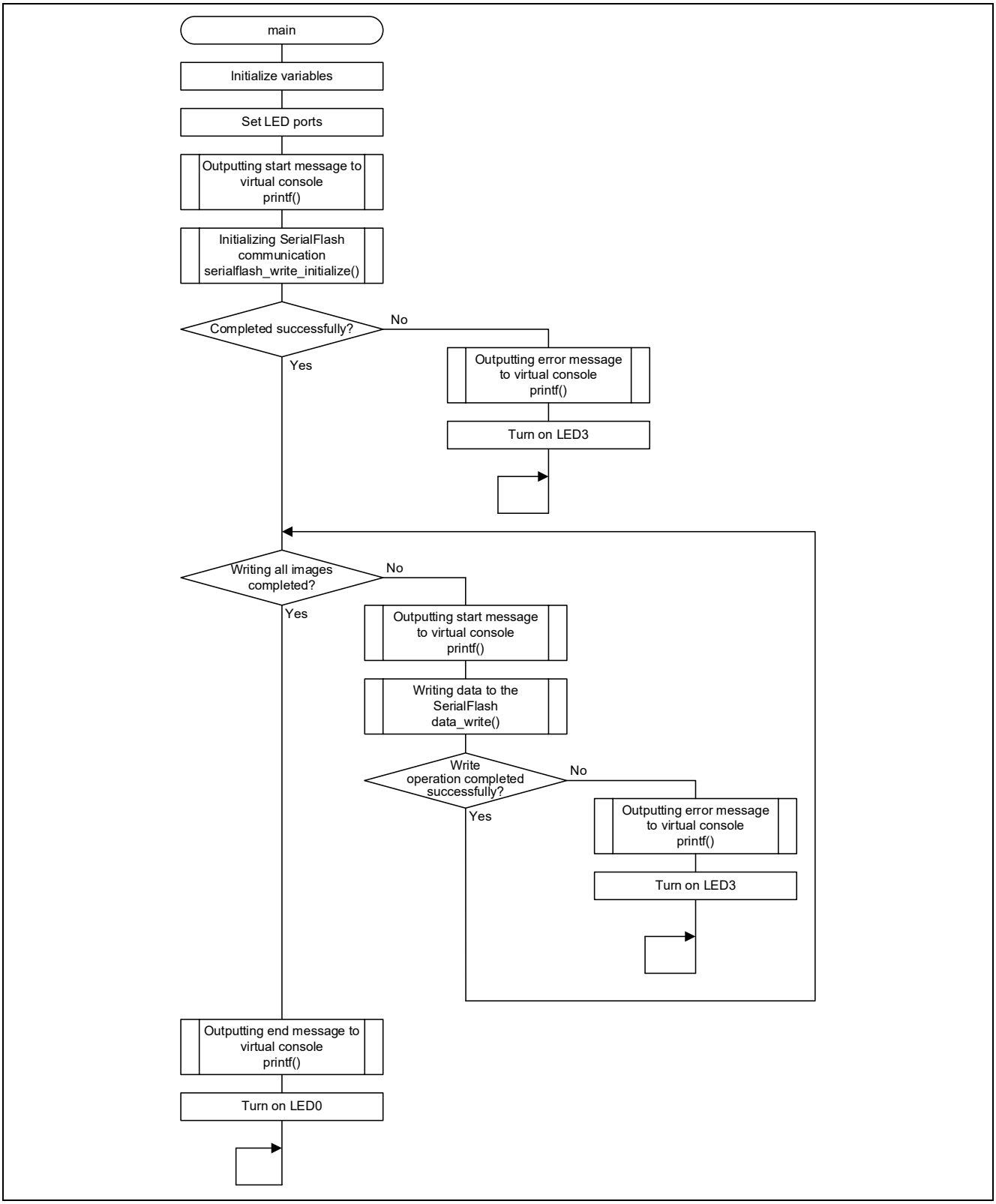


Figure 7.1 Main Processing

7.8.2 Initialization for SerialFlash Communication

Figure 7.2 shows the flowchart of initialization for SerialFlash communication.

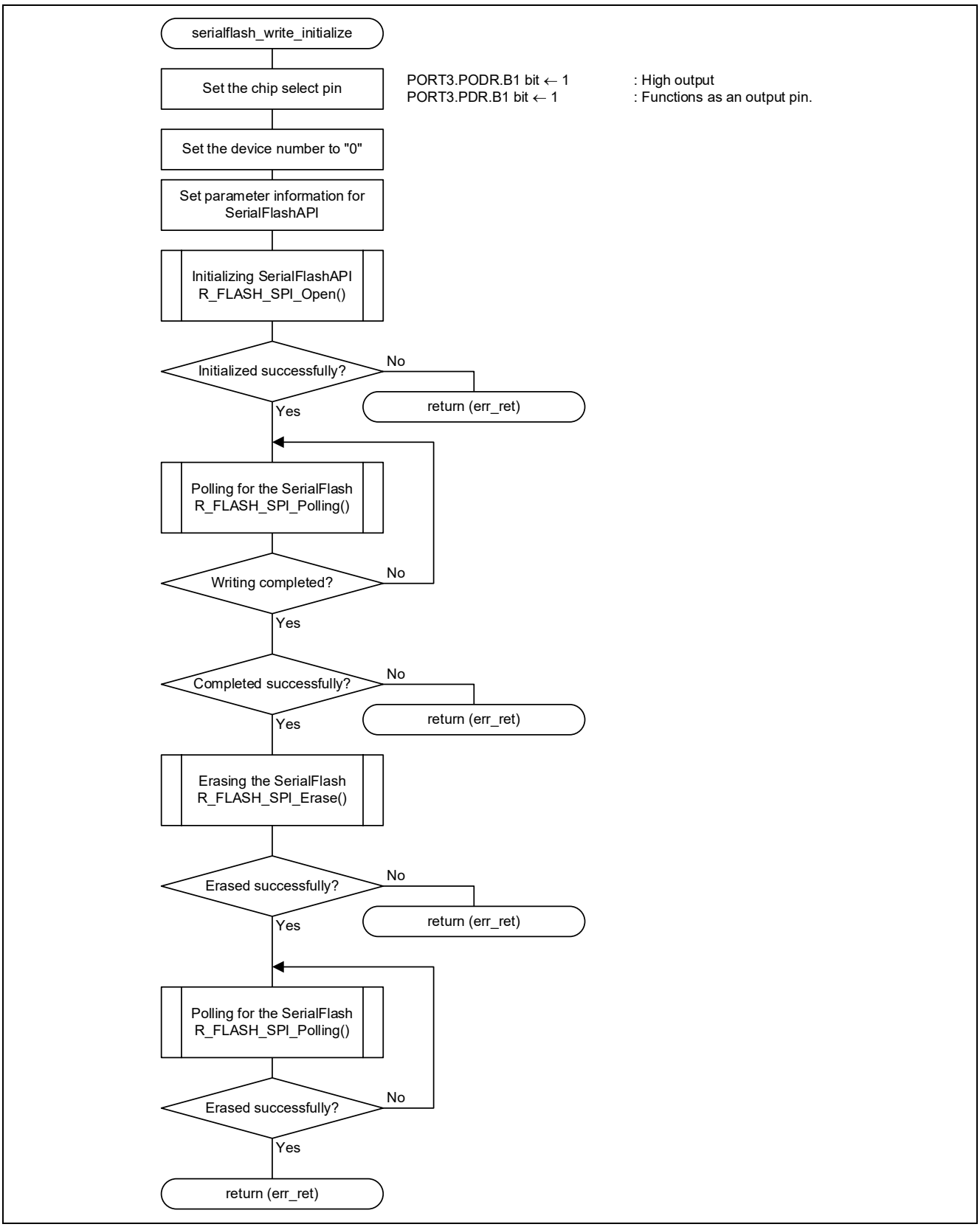


Figure 7.2 Initialization for SerialFlash Communication

7.8.3 Writing Data to the SerialFlash

Figure 7.3 shows the flowchart of processing for writing data to the SerialFlash.

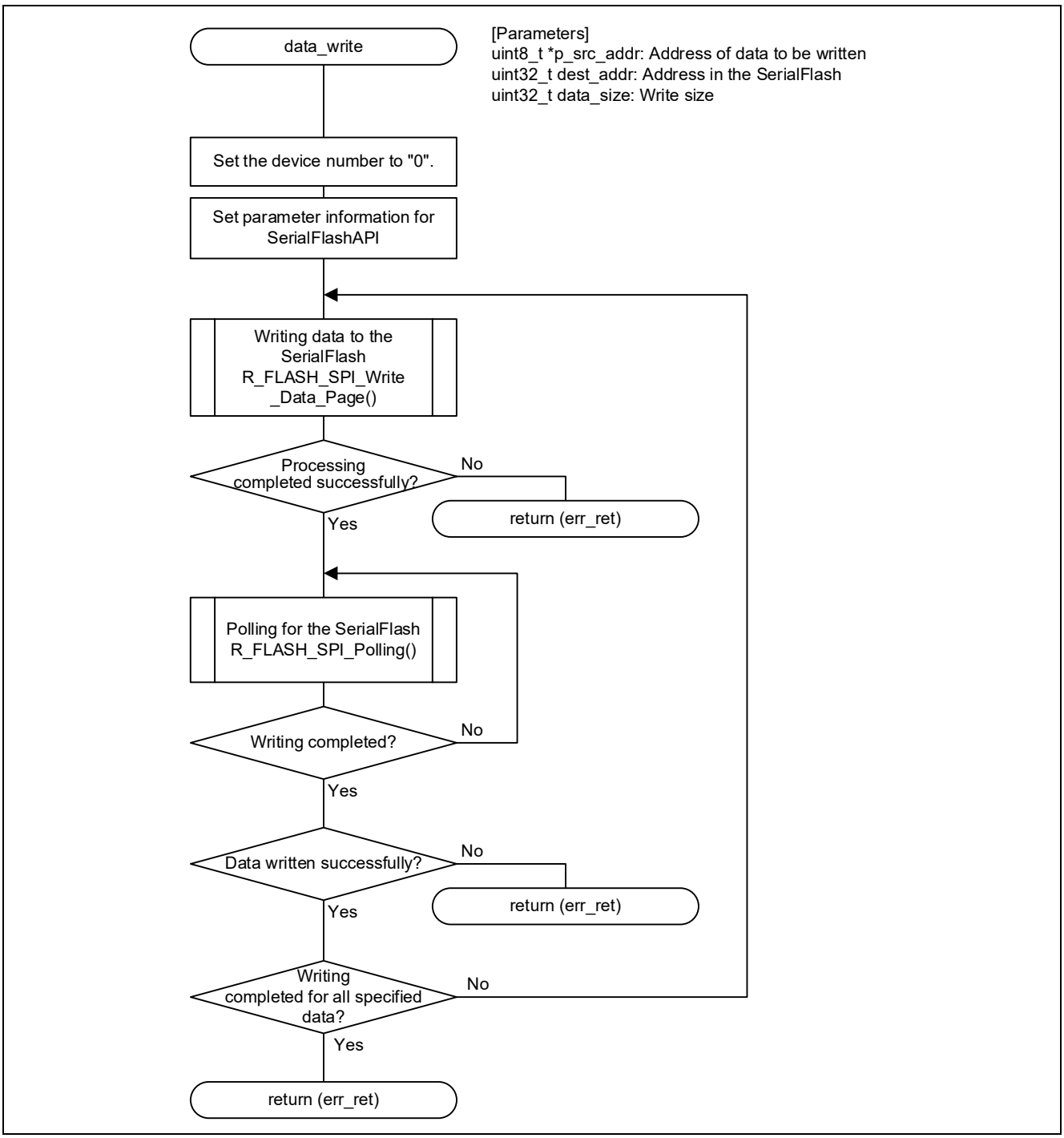


Figure 7.3 Writing Data to the SerialFlash

8. Software (glcdc_main_rx65n Project for Displaying Images)

8.1 Operation Overview

This section describes processing of the sample program project for displaying images. The sample program includes the following processing: Displaying images, touch detection and determination, changing settings, and reading the SerialFlash.

8.1.1 Settings for the Peripheral Modules and Devices

Table 8.1 lists the Settings for the GLCDC, Table 8.2 lists the Settings for the CMT, Table 8.3 lists the Settings for the SCI, and Table 8.4 lists the Settings for the RSPI. For details on each peripheral module, refer to the manual of each FIT module.

Table 8.1 Settings for the GLCDC

Item		Setting	Remarks
Graphics	Graphic screen	Graphic 1: Superimposed image Graphic 2: Displayed image Background screen: Gray screen	Settings are changed depending on the operation.
	Color format	CLUT(8) progressive format (CLUT index: 8 bits (256 entries)	Setting is specified according to the image data.
	CLUT memory (color palette)	Specifies the color palette for the image.	Color palette is distilled for the image data.
	Alpha blending	Blending view mode: Rectangle alpha blend area (The rectangle area (448 px × 253 px) is specified based on the image data. For the alpha value, the value specified in the setting mode is used.) Image view mode Per-pixel alpha blending (The alpha value of the image is used.)	Appropriate setting is used depending on the operation mode of the sample program.
Screen format		Output size: 480 px × 272 px Horizontal front porch: 3 pixels Horizontal back porch: 2 pixels Horizontal sync pulse width: 41 pixels Vertical front porch: 2 lines Vertical back porch: 2 lines Vertical sync pulse width: 10 lines	Settings are specified according to the LCD panel specifications.
Data format conversion	Output data format	RGB (565) (parallel 16 bits)	Settings are specified according to the LCD panel specifications.
	Dithering	Dithering with 2x2 pattern	Roughness of images are reduced.
	Sync signal output	TCON0: VSYNC (polarity inverted) TCON1: Not used TCON2: HSYNC (polarity inverted) TCON3: DE (polarity not inverted)	Settings are specified according to the LCD panel specifications.
Panel adjustment processing	Brightness/ contrast	Brightness: Specify the value from -512 to +512 in five levels. Contrast: Specify the value from x0 to x2 in five levels.	Settings are changed depending on the operation.
	Gamma correction	Specify the table (5 tables) corresponding to the gamma value in five levels.	Setting is changed depending on the operation.
Interrupts		Specified line detection interrupt (VPOS) Graphic 1 underflow interrupt (GR1UF) Graphic 2 underflow interrupt (GR2UF)	Interrupt for underflow detection occurs when the image data cannot be read in a specific time (high bus load)

Table 8.2 Settings for the CMT

Item	Setting	Remarks
Channel used	Channel 0	Used as the timer for execution period of main processing.
Count clock	PCLKB divided by 128	PCLKB = 60 MHz
Compare match period	100 ms (23,912 counts)	

Table 8.3 Settings for the SCI

Item	Setting	Remarks
Channel used	Channel 7	Used for communicating with the touch controller of the LCD panel.
Communication method	Simple I^2C mode	
Transfer speed (bit rate)	384 kbps	

Table 8.4 Settings for the RSPI

Item	Setting	Remarks
Channel used	Channel 1	Used for communicating with the SerialFlash. Read only
Communication method	Clock synchronous mode	
Transfer speed (bit rate)	30 Mbps	

8.1.2 Displaying Images

The GLCDC is used to display images. By enabling GLCDC operation after the necessary registers are set, the image is read from the memory area at the specified address and necessary processing is performed. Then the image is output to the LCD panel. The GLCDC repeatedly reads the image data from that specified address while operating and continues to output waveforms to the LCD panel.

In this application note, the address from which the GLCDC reads the image data is fixed. So the image displayed on the LCD panel can be changed by directly overwriting the image data at that address.

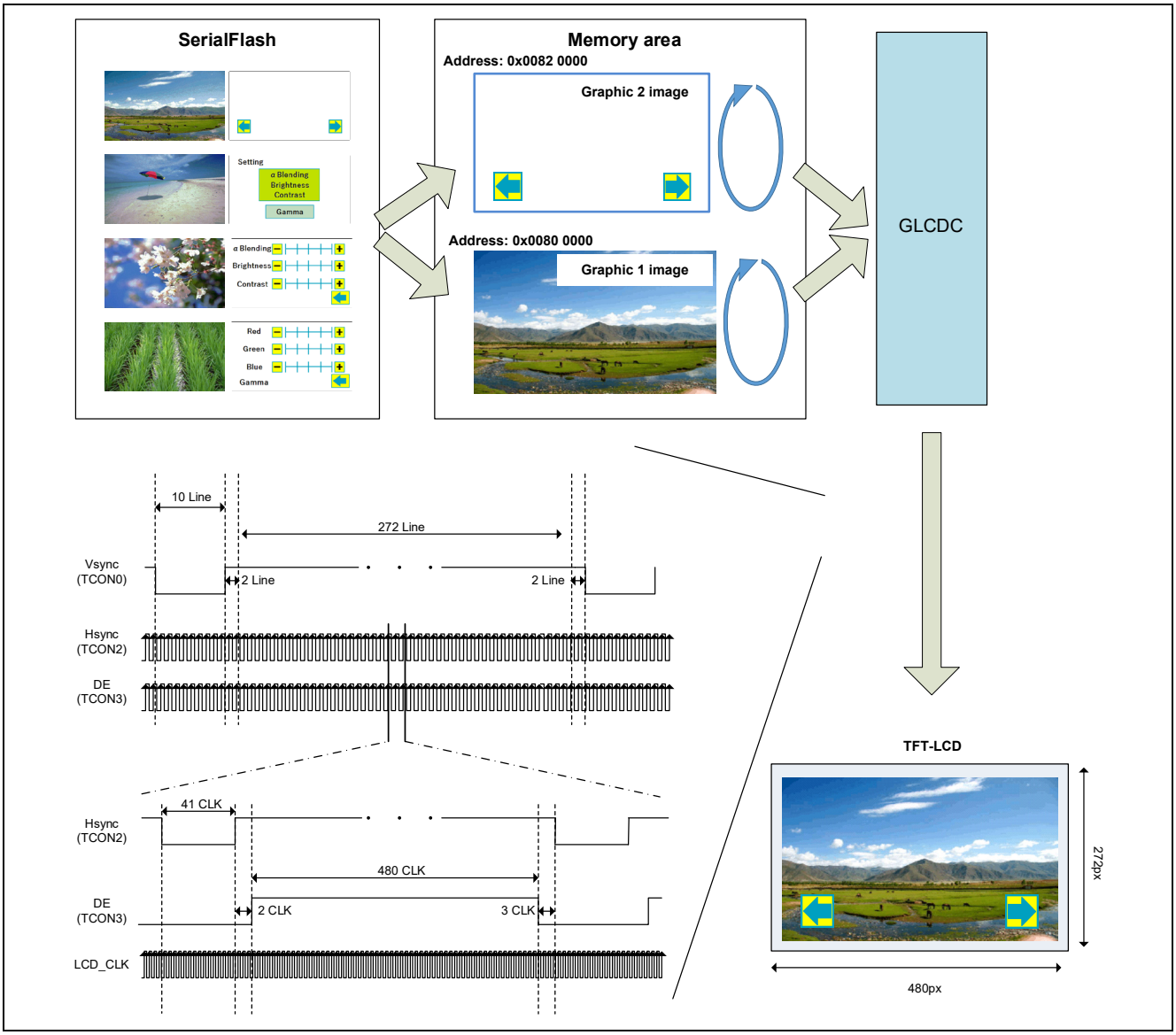


Figure 8.1 Outputting the LCD Panel

8.1.3 Touch Detection and Determination Processing

I^2C protocol is used to communicate with the touch controller on the LCD panel and this communication is performed every 100 ms using CMT0. The information is read from the touch controller every 100 ms to determine whether the button on the LCD panel is touched.

The following describes touch detection and determination processing.

1. Touch detection processing (obtaining the information from the touch controller).

The touch controller has the status registers which store the state of the touch detection. When the master transmits the address, the value of the correspondent status register can be obtained. In this sample program, the following registers are read to determine whether the buttons on the panel are touched.

For details on the specifications of the touch controller on the LCD panel, refer to the datasheet for the LCD panel.

Table 8.5 Touch Controller Registers to be Referenced

Register Name	Address [Bit]	Description
Touch Points register	02h [3:0 bits]	000b: Not touched 001b to 101b: Touch detected (Up to 5 points can be detected.)
Touch 1 Event Flag register	03h [7:6 bits]	00b: Pressed 01b: Released 10b: Being touched
TOUCH1_XH register	03h [3:0 bits]	Upper 4 bits of X coordinate
TOUCH1_XL register	04h [7:0 bits]	Lower 8 bits of X coordinate
TOUCH1_YH register	05h [3:0 bits]	Upper 4 bits of Y coordinate
TOUCH1_YL register	06h [7:0 bits]	Lower 8 bits of Y coordinate

2. Determination processing

This processing determines whether the buttons on the LCD panel are pressed based on the information obtained from the touch controller.

Touch input is determined with the following condition based on the condition in Table 8.5.

- Value in the Touch Points register > 000b
- Value in the Touch 1 Event Flag register == 10b

Then, when the obtained coordinates are within the range shown in Figure 8.2, the flag information corresponding to each button is set.

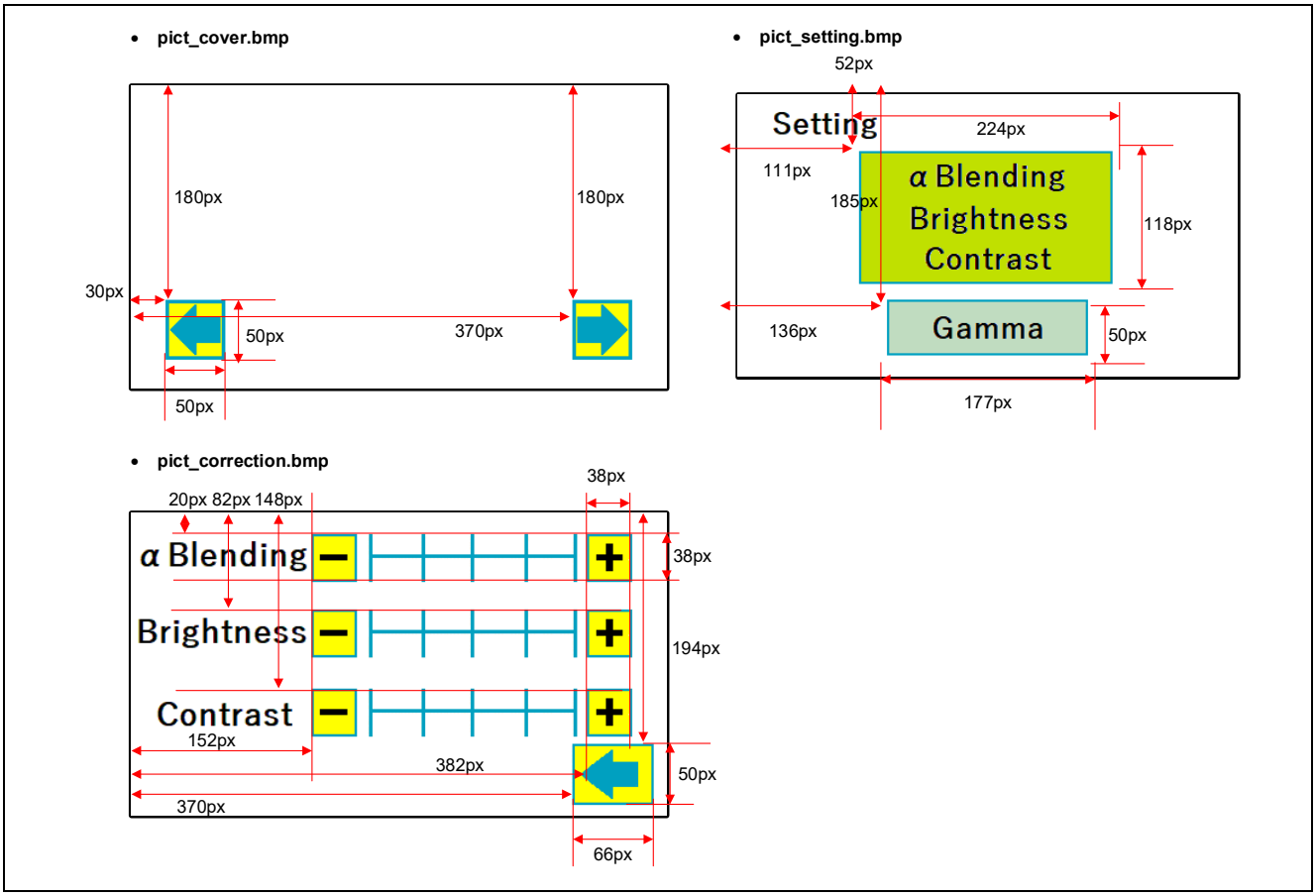


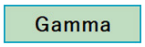
Figure 8.2 Coordinates of Buttons

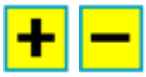
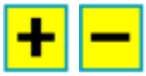
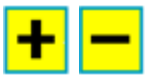
8.1.4 Changing the Setting

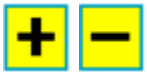
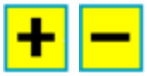
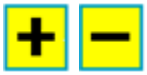
This processing changes the screen mode and displayed image, and updates the information of brightness/contrast/gamma value based on the switch (SW1 to SW3) information and the result of touch detection/determination processing.

Table 8.6 lists Processing Corresponding to Each Button Operation.

Table 8.6 Processing Corresponding to Each Button Operation

Screen Mode	Button or Switch	Item to Be Changed	Processing
Image view mode	arrow (←) 	Image of Graphic 1	Obtains the previous image from the SrialFlash and overwrites the current displayed image data with it. Display example: 1 → 4 → 3 → 2 → 1 → 4 … (numbers indicate the image numbers)
	arrow (→) 		Obtains the next image from the SrialFlash and overwrites the current displayed image data with it. Display example: 1 → 2 → 3 → 4 → 1 … (numbers indicate the image numbers)
Blending view mode	Whole screen	Images of Graphic 1 and Graphic 2	Obtains the next image from the SrialFlash and overwrites the current displayed image data with it. Display example: 1, 2 → 2, 3 → 3, 4 → 4, 1 → 1, 2 … (numbers indicate the image numbers)
Setting mode	α blending Brightness Contrast 	Image of Graphic 2	Obtains the alpha value/brightness/contrast setting image (pict_correction.bmp) from the SerialFlash and overwrite the current displayed image data with it. Executes processing to reflect the setting value in the horizontal bar on the setting screen. (“■” is shown at the intersection of the horizontal bar with a vertical line.)
	Gamma 	Image of Graphic 2	Obtains the gamma setting image (pict_gamma.bmp) from the SerialFlash and overwrite the current displayed image data with it. Executes processing to reflect the setting value in the horizontal bar on the setting screen. (“■” is shown at the intersection of the horizontal bar and a vertical line.)

Screen Mode	Button or Switch	Item to Be Changed	Processing
Setting mode (alpha value, brightness, contrast)	"+" and "-" for alpha blending 	Alpha value of Graphic 2 Register: GR2AB7 register - ARCDEF bit	Changes the alpha value (value set in the GR2AB7.ARCDEF bit) of Graphic 2 in five levels. (Available value set is 0 to 255.) 5: 255 (image on upper layer is displayed) 4: 191 3: 128 (initial value for the sample program) 2: 64 1: 0 (image on lower layer is displayed) Executes processing to reflect the setting value in the horizontal bar on the setting screen. ("■" is shown at the intersection of the horizontal bar and a vertical line.)
	"+" and "-" for brightness 	Brightness value Registers: BRIGHT1 register - BRTG bit BRIGHT2 register - BRTR bit - BRTB bit	Changes the brightness values (values set in the BRIGHT1.BRTG bit, and BRIGHT2.BRTR and BRTB bits) in five levels. (Available value set is 0 to 1023.) 5: 1023 (bright) 4: 767 3: 512 (initial value for the sample program) 2: 256 1: 0 (dark) * The same brightness value is used for RGB. Executes processing to reflect the setting value in the horizontal bar on the setting screen. ("■" is shown at the intersection of the horizontal bar and a vertical line.)
	"+" and "-" for contrast 	Contrast value Register: CONTRAST register - CONTR bit - CONTB bit - CONTG bit	Changes the contrast value (value set in the CONTRAST.CONTR, CONTB, and CONTG bits) in five levels. (Available value set is 0 to 255.) 5: 255 (light) 4: 191 3: 128 (initial value for the sample program) 2: 64 1: 0 (dark) * The same contrast value is used for RGB. Executes processing to reflect the setting value in the horizontal bar on the setting screen. ("■" is shown at the intersection of the horizontal bar and a vertical line.)
	Back button 	Image of Graphic 2	Loads the image for setting mode (pict_setting.bmp) from the SerialFlash and replaces the current displayed image with it.

Screen Mode	Button or Switch	Item to Be Changed	Processing
Setting mode (Gamma value)	"+" and "-" for R value 	Gamma value for R Registers: Registers GAMRLUT1 to GAMRLUT8 Registers GAMRAREA1 to GAMRAREA4	Changes the gamma values of R (values set in registers GAMRLUT1 to GAMRLUT8 and registers GAMRAREA1 to GAMRAREA4). For each setting value, five table data are switched according to γ value. 5: Table 5 → γ = 1.30 4: Table 4 → γ = 1.10 3: Table 3 → γ = 0.90 (initial value for the sample program) 2: Table 2 → γ = 0.70 1: Table 1 → γ = 0.50 Executes processing to reflect the setting value in the horizontal bar on the setting screen. ("■" is shown at the intersection of the horizontal bar and a vertical line.)
	"+" and "-" for G value 	Gamma value for G Registers: Registers GAMGLUT1 to GAMGLUT8 Registers GAMGAREA1 to GAMGAREA4	Changes the gamma values of G (values set in registers GAMGLUT1 to GAMGLUT8 and registers GAMGAREA1 to GAMGAREA4). For each setting value, five table data are switched according to γ value. 5: Table 5 → γ = 1.30 4: Table 4 → γ = 1.10 3: Table 3 → γ = 0.90 (initial value for the sample program) 2: Table 2 → γ = 0.70 1: Table 1 → γ = 0.50 Executes processing to reflect the setting value in the horizontal bar on the setting screen. ("■" is shown at the intersection of the horizontal bar and a vertical line.)
	"+" and "-" for B value 	Gamma value for B Registers: Registers GAMBLUT1 to GAMBLUT8 Registers GAMBAREA1 to GAMBAREA4	Changes the gamma values of B (values set in registers GAMBLUT1 to GAMBLUT8 and registers GAMBAREA1 to GAMBAREA4). For each setting value, five table data are switched according to γ value. 5: Table 5 → γ = 1.30 4: Table 4 → γ = 1.10 3: Table 3 → γ = 0.90 (initial value for the sample program) 2: Table 2 → γ = 0.70 1: Table 1 → γ = 0.50 Executes processing to reflect the setting value in the horizontal bar on the setting screen. ("■" is shown at the intersection of the horizontal bar and a vertical line.)

Screen Mode	Button or Switch	Item to Be Changed	Processing
Setting mode (Gamma value)	Back button 	Image of Graphic 2	Loads the image for setting mode (pict_setting.bmp) from the SerialFlash and replaces the current displayed image with it.
All screens mode	SW1	Images of Graphic 1 and Graphic 2 Alpha blending and alpha value of Graphic 2 Register: GR2AB1 register - ARCON bit	Saves the information of images in the current mode and then switches the mode to image view mode. Loads images of Graphic 1 and Graphic 2 from the SerialFlash based on the saved information and replaces the images. Changes the alpha blending setting to per-pixel alpha blending.
	SW2	Images of Graphic 1 and Graphic 2 Alpha blending and alpha value of Graphic 2 Register: GR2AB1 register - ARCON bit GR2AB7 register - ARCDEF bit	Saves the information of images in the current mode and then switches the mode to blending view mode. Loads images of Graphic 1 and Graphic 2 from the SerialFlash based on the saved information and replaces the images. Changes the alpha blending setting to rectangle alpha blending area and the alpha blending value to the value for blending view mode.
	SW3	Image of Graphic 1 Alpha blending and alpha value of Graphic 2 Register: GR2AB1 register ARCON bit	Saves the information of images in the current mode and then switches the mode to setting mode. Loads the image (pict_setting.bmp) for setting mode from the SerialFlash and replaces the current displayed image with it. Changes the alpha blending setting to per-pixel alpha blending.

8.1.5 Reading the SerialFlash

The image data are stored in the SerialFlash and loaded each time when necessary. The data is stored in BMP (Windows Bitmap Image) in the SerialFlash. Thus the image data and its color palette are loaded separately.

Table 8.7 shows the Address Map of Image Data in RX65N and the SerialFlash (Project for Displaying Images). For images used, refer to 1.2 Image Data Used in the Sample Program.

Table 8.7 Address Map of Image Data in RX65N and the SerialFlash (Project for Displaying Images)

Device	Image Data/Descriptions		Start Address
RX65N (Section: RIMAGE)	Address to place the data for Graphic 1		0x00800000
	Address to place the data for Graphic 2		0x00820000
	Address to place the data for the buffer (Only when big endian is used)		0x00840000
SerialFlash (for loading the data)	pict_data1.bmp	Header (54 bytes)	0x00000000
		Color palette (1,024 bytes)	0x00000036
		Image data (113,344 bytes)	0x00000436
	pict_data2.bmp	Header (54 bytes)	0x00025800
		Color palette (1,024 bytes)	0x00025836
		Image data (113,344 bytes)	0x00025C36
	pict_data3.bmp	Header (54 bytes)	0x0004B000
		Color palette (1,024 bytes)	0x0004B036
		Image data (113,344 bytes)	0x0004B436
	pict_data4.bmp	Header (54 bytes)	0x00070800
		Color palette (1,024 bytes)	0x00070836
		Image data (113,344 bytes)	0x00070C36
	pict_cover.bmp	Header (54 bytes)	0x00096000
		Color palette (1,024 bytes)	0x00096036
		Image data (113,344 bytes)	0x00096436
	pict_setting.bmp	Header (54 bytes)	0x000BB800
		Color palette (1,024 bytes)	0x000BB836
		Image data (113,344 bytes)	0x000BBC36
	pict_correction.bmp	Header (54 bytes)	0x000E1000
		Color palette (1,024 bytes)	0x000E1036
		Image data (113,344 bytes)	0x000E1436
	pict_gamma.bmp	Header (54 bytes)	0x00106800
		Color palette (1,024 bytes)	0x00106836
		Image data (113,344 bytes)	0x00106C36

1. Loading the color palette

The color palette is 1024 bytes and consists of B (1 byte), G (1 byte), R (1 byte), and A (1 byte). The information of them for each image is stored in the Color Look-up Table (GRnCLUTm[k]) register (n = 1, 2, m = 0, 1, k = 0 to 255) in the GLCDC. Note that A (1 byte) in BMP is normally “00h” in 256 colors (8 bits). If “00h” is stored into the register as it is, the alpha value becomes “00h” (lower layer is displayed for per-pixel alpha blending). Therefore all alpha values are overwritten with “FFh” before being stored in registers.

Values of the BMP 256-color palette (8 bits) are the same for any images. Thus, in this application note, the color lookup table register is configured only once when the GLCDC is initialized. After initialization, read values of the color palette are discarded. When the image format is other than BMP or a specific color palette is used for each image, the color lookup table register also needs to be changed for each image.

2. Loading the image data

The image data is directly loaded into the data storage address for Graphic 1 (0x00800000) or Graphic 2 (0x00820000) (see in Table 8.7). The GLCDC reads the image data from the specified base address. Thus overwriting the image at that address can directly change the image to be displayed.

Note that the endianness of the data read from the SerialFlash is little endian regardless of project settings or device settings. When using big endian, store the color palette and image data at the data storage address for the buffer (0x00840000) once to convert the endianness, and then store the image at the data storage address for Graphic 1 or Graphic 2.

8.2 File Composition

Table 8.8 lists the Files Used in the Sample Program and Table 8.9 lists the Standard Include File. Files in the FIT modules and files generated by the integrated development environment are not included in these table.

Table 8.8 Files Used in the Sample Program

Fine Name	Outline	Remarks
main.c	Main processing	
r_screen.c	GLCDC initialization, control processing for displayed images, processing for touch control, etc.	
r_screen.h	Header file for r_screen.c	
r_serial_flash_read.c	RSPI initialization and processing for reading data from the SerialFlash	
r_serial_flash_read.h	Header file for r_serial_flash_read.c	

Table 8.9 Standard Include File

File Name	Outline
stdbool.h	Defines macros regarding the Boolean type and the Boolean value.
stdint.h	Defines macros by declaring the integer type of the specified width.
machine.h	Defines formats of built-in functions for RX Family.
string.h	Library for processing such as string comparison and copy.
stddef.h	Defines a common macro name used by each standard include file.

8.3 Option Setting Memory

Table 8.10 shows the state of the option setting memory used in the sample program. Please specify values most appropriate to your system as required.

Table 8.10 Option Setting Memory Used in the Sample Program

Symbol	Address	Setting Value	Description
OFS0	FE7F 5D04h to FE7F 5D07hh	FFFF FFFFh	WDT/IWDT stopped after a reset
OFS1	FE7F 5D08h to FE7F 5D0Bh	FFFF FFFFh	Voltage monitor 0 reset disabled after a reset HOCO oscillation disabled after a reset
MDE	FE7F 5D00h to FE7F 5D03h	FFFF FFFFh	Little endian

8.4 Constants

Table 8.11 and Table 8.12 list constants used in the sample program.

Table 8.11 Constants Used in the Sample Program (main.c)

Constant	Setting Value	Description
SW1_PIDR	(PORT0.PIDR.BIT.B3)	PIDR register bit for SW1(P03)
SW2_PIDR	(PORT0.PIDR.BIT.B5)	PIDR register bit for SW2(P05)
SW3_PIDR	(PORT0.PIDR.BIT.B7)	PIDR register bit for SW3(P07)
SW1_PDR	(PORT0.PDR.BIT.B3)	PDR register bit for SW1(P03)
SW2_PDR	(PORT0.PDR.BIT.B5)	PDR register bit for SW2(P05)
SW3_PDR	(PORT0.PDR.BIT.B7)	PDR register bit for SW3(P07)
PUSH_SW1	(1)	Defined value when SW1 is pressed
PUSH_SW2	(2)	Defined value when SW2 is pressed
PUSH_SW3	(3)	Defined value when SW3 is pressed
PUSH_NONE	(-1)	Defined value when no switch is pressed
LED_ON	(0)	LED turned on
LED_OFF	(1)	LED turned off
LED3	(PORTG.PODR.BIT.B5)	PODR register bit for LED3(PG5)
LED3_PDR	(PORTG.PDR.BIT.B5)	PDR register bit for LED3(PG5)

Table 8.12 Constants Used in the Sample Program (r_screen.h)

Constant	Setting Value	Description
BMP_HEADER_CODE	(0xBEF64D42)	Character code (“BM”) on the top of the bitmap file
BMP_SIZE	(114422)	Bitmap file size
BMP_HEADER_SIZE	(54)	Header size of the bitmap file
BMP_COLOR_PALLETE_SIZE	(1024)	Color palette size of the bitmap file
BMP_IMAGE_DATA_SIZE	(113344)	Image data size of the bitmap file
BMP_IMAGE_DATA_OFFSET	(BMP_HEADER_SIZE + BMP_COLOR_PALLETE _SIZE)	Offset from the start of the bitmap file to the image data.
BMP_IMAGE_WIDTH	(448)	Horizontal width of the bitmap image
BMP_IMAGE_HEIGHT	(253)	Vertical width of the bitmap image
IMAGE_RELOCATION_ADDR	(0x00800000)	Address to place the data for Graphic 1
IMAGE_RELOCATION_ADDR2	(0x00820000)	Address to place the data for Graphic 2
IMAGE_BUFFER_ADDR	(0x00840000)	Address to place the data for the buffer (Used only when big endian is used.)
IMAGE_BASE_ADDR	(IMAGE_RELOCATION_ ADDR + (BMP_IMAGE_ DATA_SIZE - BMP_ IMAGE_WIDTH))	Graphic 1 base address to be set in the GLCDC
IMAGE_BASE_ADDR2	(IMAGE_RELOCATION_ ADDR2 + (BMP_IMAGE_ DATA_SIZE - BMP_ IMAGE_WIDTH))	Graphic 2 base address to be set in the GLCDC
IMAGE_COUNT_NUM	(4)	Number of images to be displayed

Constant	Setting Value	Description
IMAGE_COVER_NUM	(4)	Number of images for the superimposed image
CORRECTION_LEVEL_NUM	(5)	Number of correction setting levels
INITIAL_CORRECTION_LEVEL	(2)	Setting value at startup of the application
TOUCH_READ_INFO_SIZE	(5)	Data size obtained from the touch panel controller
TOUCH_BUTTON_NUM	(12)	Number of button types used in the touch panel
IMAGE_X_OFFSET	(16)	Start position to display an image (X-axis)
IMAGE_Y_OFFSET	(9)	Start position to display an image (Y-axis)
RECTANGLE_BLOCK_SIZE	(28)	Vertical/horizontal pixel size of a rectangle drawing
RECTANGLE_BLOCK_COLOR	(0x80)	Color of rectangle drawing (color palette number)
RECTANGLE_BUF_SIZE	(32)	Buffer used for rectangle drawing

8.5 Structures and Enumerations

This section describes structures and enumerations used for application control such as changing screen mode and touch detection/determination.

r_screen.h

```
/* Button size structure */
typedef struct
{
    uint16_t width;                /* Button width */
    uint16_t height;               /* Button height */
} touch_button_size_t;

/* Button information structure */
typedef struct
{
    touch_event_t event;           /* Button event */
    glcdc_coordinate_t pos;        /* Top left coordinate of the button */
    touch_button_size_t size;      /* Button size */
    uint8_t mode;                  /* Mode to enable the button */
} touch_button_t;

/* Register definition information structure for the touch controller  */
#pragma bit_order left
#pragma unpack
typedef struct st_touch_info {
    struct {
        uint8_t :5;
        uint8_t touch_points:3;    /* Number of touch detections */
    } td_status;
    struct {
        uint8_t event_flag:2;      /* Touch state (pressed, released, being touched) */
        uint8_t :2;
        uint8_t x_pos_msb:4;       /* Upper 4 bits of X coordinate */
    } touch1_xh;
    uint8_t touch1_xl;             /* Lower 8 bits of X coordinate */
    struct {
        uint8_t :4;
        uint8_t y_pos_msb:4;       /* Upper 4 bits of Y coordinate */
    } touch1_yh;
    uint8_t touch1_yl;             /* Lower 8 bits of Y coordinate */
}touch_info_t;
#pragma bit_order
#pragma packoption
```

Figure 8.3 Structures

```
/* Screen mode */
typedef enum
{
    IMAGE_VIEW_MODE = 0x01,                 /* Image view mode */
    BLEND_VIEW_MODE = 0x02,                 /* Blending view mode */
    SETTING_MODE = 0x04,                    /* Setting mode (setting selection) */
    SETTING_CORRECTION_MODE = 0x08,   /* Setting mode (alpha value/brightness/contrast setting) */
    SETTING_GAMMA_MODE = 0x10,          /* Setting mode (gamma setting) */
    ALL_MODE = 0x1F                         /* Common in modes */
} view_mode_t;

/* Touch event */
typedef enum
{
    NO_TOUCH = 0U,
   /* Image view mode */
    TOUCH_LEFT,                 /* Left button */
    TOUCH_RIGHT,                /* Right button */
   /* Setting mode (setting selection) */
    TOUCH_CORRECTION,     /* Button for setting alpha value/brightness/contrast */
    TOUCH_GAMMA,              /* Button for setting gamma */
   /* Setting mode (alpha value/brightness/contrast or gamma setting) */
    TOUCH_BAR1_DEC,         /* Decrement button for bar 1 */
    TOUCH_BAR1_INC,         /* Increment button for bar 1 */
    TOUCH_BAR2_DEC,         /* Decrement button for bar 2 */
    TOUCH_BAR2_INC,         /* Increment button for bar 2 */
    TOUCH_BAR3_DEC,         /* Decrement button for bar 3 */
    TOUCH_BAR3_INC,         /* Increment button for bar 3 */
    TOUCH_BACK,               /* Back button */
   /* Common in modes */
    TOUCH_OTHER,              /* Other than button */
} touch_event_t;
```

Figure 8.4 Enumerations

8.6 Variables

Table 8.13 and Table 8.14 list static variables and Table 8.15 lists the const variables. Variables included in the Fit modules are not listed in these tables.

Table 8.13 staic Variable (main.c)

Type	Variable Name	Description	Function
static volatile bool	scan_period_flag	Check flag for switch/touch panel detection period	main set_scan_period_flag

Table 8.14 staic Variables (r_screen.c)

Type	Variable Name	Description	Function
sci_iic_info_t	siic_info	Simple I^2C module configuration	touch_initialize scan_touch
uint8_t	touch_data[TOUCH_READ_INFO_SIZE]	Receive data from the touch controller	scan_touch
glcdc_cfg_t	glcdc_init_cfg	GLCDC initialization	screen_mode_change screen_update glcdc_initialize
uint32_t	gr_clut_table[256]	Color palette data	screen_mode_change screen_update glcdc_initialize
bool	first_interrupt_flag	Check flag for the first interrupt in the GLCDC	glcdc_initialize glcdc_callback
view_mode_t	current_mode	Current mode	get_current_mode screen_mode_change
int8_t	image_count	Image number of the image being displayed	screen_mode_change screen_update
int8_t	blend_image1_count	Image number 1 in current blending view	screen_mode_change screen_update
int8_t	blend_image2_count	Image number 2 in current blending view	screen_mode_change screen_update
uint8_t	current_blend_level	Current value in blending view	screen_mode_change screen_update
uint8_t	current_bright_level	Current value of brightness	screen_mode_change screen_update
uint8_t	current_contrast_level	Current value of contrast	screen_mode_change screen_update
uint8_t	current_gamma_r_level	Current value of gamma (R value)	screen_mode_change screen_update
uint8_t	current_gamma_g_level	Current value of gamma (G value)	screen_mode_change screen_update
uint8_t	current_gamma_b_level	Current value of gamma (B value)	screen_mode_change screen_update

Table 8.15 const Variables (r_screen.c)

Type	Variable Name	Description	Function
uint32_t	gp_pict_table[IMAGE_COUNT_NUM]	SerialFlash image allocation (address) table for image 1 to image 4	main glcdc_initialize screen_mode_change screen_update bmp_image_check
uint32_t	gp_pict_cover_table[IMAGE_COVER_NUM]	SerialFlash image allocation (address) table for superimposed image	glcdc_initialize screen_mode_change bmp_image_check
uint32_t	g_common_level_table[CORRECTION_LEVEL_NUM]	Setting table for ratio of contrast/blending	screen_mode_change screen_update
uint32_t	g_bright_level_table[CORRECTION_LEVEL_NUM]	Setting table for brightness	screen_update
gamma_correction_t	g_gamma_table	Gamma correction data	glcdc_initialize
glcdc_coordinate_t	g_cross_point_bar_table[3][5]	Table for coordinates of intersection of bars	screen_mode_change
touch_button_t	g_touch_button_table[TOUCH_BUTTON_NUM]	Table for coordinates of button allocation of the LCD panel	scan_touch

8.7 Functions

Table 8.16 to Table 8.18 list functions.

Table 8.16 Functions (main.c)

Function Name	Outline
main	Main processing
check_sw	Checking switches
set_scan_period_flag	Setting the flag for switch/touch panel detection period

Table 8.17 Functions (r_screen.c)

Function Name	Outline
get_current_mode	Obtaining the current screen mode
screen_mode_change	Screen mode switching
screen_update	Updating the displayed image and the GLCDC settings
glcdc_initialize	Initializing and starting the GLCDC
glcdc_port_setting	Configuring pins used for the LCD panel
glcdc_callback	Callback processing for the GLCDC API functions
touch_initialize	Initializing touch panel communication
scan_touch	Obtaining touch information from the touch panel
touch_callback	Callback processing for the simple I^2C FIT module
fill_clut_alpha_data	Overwriting the alpha value of the color palette
draw_rectangle	Rectangle drawing
bmp_image_load	Reading image data from the SerialFlash
bmp_image_check	Checking the image stored in the SerialFlash
convert_endian	Endian conversion (Used only when the endianness in the CPU is big endian)

Table 8.18 Functions (r_serial_flash_read.c)

Function Name	Outline
serialflash_read_initialize	Initializing the SerialFlash communication
data_read	Reading data from the SerialFlash

8.8 Function Specifications

This section describes function specifications.

8.8.1 Functions (main.c)

main	
Outline	Main processing
Header	None
Declaration	void main(void)
Description	After the GLCDC initialization, switches displayed images or screen modes according to switch pressed or touch input on the LCD panel.
Parameters	None
Return Values	None

check_sw	
Outline	Checking switches
Header	None
Declaration	static int8_t check_sw (void)
Description	When a switch is pressed, this function returns the value corresponding to the switch pressed.
Parameters	None
Return Values	Number for the switch pressed 1: SW1 2: SW2 3: SW3 -1: Others

set_scan_period_flag	
Outline	Setting the flag for switch/touch panel detection period
Header	None
Declaration	static void set_scan_period_flag (void)
Description	Sets the flag for detection period to “true”.
Parameters	None
Return Values	None

8.8.2 Functions (r_screen.c)

get_current_mode	
Outline	Obtaining the current screen mode
Header	screen.h
Declaration	view_mode_t get_current_mode (void)
Description	Returns the status of the current screen mode.
Parameters	None
Return Values	IMAGE_VIEW_MODE: Image view mode BLEND_VIEW_MODE: Blending view mode SETTING_MODE: Setting mode (setting selection) SETTING_CORRECTION_MODE: Setting mode (alpha value/brightness/contrast setting) SETTING_GAMMA_MODE: Setting mode (gamma setting)

screen_mode_change	
Outline	Screen mode switching
Header	screen.h
Declaration	void screen_mode_change(view_mode_t mode)
Description	Changes the screen mode to the mode specified by the parameter.
Parameters	view_mode_t mode: Mode to change
Return Values	None

screen_update	
Outline	Updating the displayed image and the GLCDC settings
Header	screen.h
Declaration	void screen_update(view_mode_t mode, touch_event_t event)
Description	Updates the displayed image or changes the GLCDC settings (alpha value/brightness/contrast, gamma) according to the values of the first and second parameters.
Parameters	view_mode_t mode: Current screen mode touch_event_t: Event
Return Values	None

glcdc_initialize	
Outline	Initializing and starting the GLCDC
Header	screen.h
Declaration	static void glcdc_initialize (void)
Description	Prepares for image data, performs GLCDC pin configuration and initialization, and starts GLCDC operation.
Parameters	None
Return Values	None

glcdc_port_setting	
Outline	Configuring pins used for the LCD panel
Header	None
Declaration	static void glcdc_port_setting (void)
Description	Performs GLCDC pin configuration.
Parameters	None
Return Values	None

glcdc_callback	
Outline	Callback processing for the GLCDC API functions
Header	None
Declaration	static void glcdc_callback(void * pdata)
Description	This is the function called from interrupt handlers in the GLCDC. Callback functions are registered with the API function.
Parameters	void * pdata: Information of an interrupt occurred - GLCDC_EVENT_GR1_UNDERFLOW: Graphic 1 underflow - GLCDC_EVENT_GR2_UNDERFLOW: Graphic 2 underflow - GLCDC_EVENT_LINE_DETECTION: Graphic 2 line detection
Return Values	None
Remarks	After the GLCDC software reset is released, unintended Graphic 2 specified line notification, Graphic 1 underflow, and Graphic 2 underflow are detected only for the first time. Therefore the sample program performs nothing in the interrupt handler for the first detection of Graphic 2 specified line notification after the R_GLCDC_Open function is executed. The user processing is executed in the next interrupt handler. Refer to the manual of the graphic LCD controller module for details. When implementing this function into the user system, an appropriate processing should be implemented for each interrupt as required.

touch_initialize	
Outline	Initializing touch panel communication
Header	screen.h
Declaration	void touch_initialize(void)
Description	Initializes the simple I^2C FIT module and configures the communication.
Parameters	None
Return Values	None

scan_touch	
Outline	Obtaining touch information from the touch panel
Header	screen.h
Declaration	touch_event_t scan_touch(view_mode_t mode)
Description	Communicates with the touch panel controller and obtains the information of the button touched.
Parameters	view_mode_t mode: Mode information
Return Values	**All mode:** NO_TOUCH: Not touched TOUCH_OTHER: Touched other than buttons **When in image view mode:** TOUCH_LEFT: Left button (for image view mode) TOUCH_RIGHT: Right button (for image view mode) **When in setting mode (setting selection):** TOUCH_CORRECTION: Button for setting alpha value/brightness/contrast TOUCH_GAMMA: Button for gamma setting **When in setting mode** **(alpha value/brightness/contrast setting or gamma setting):** TOUCH_BAR1_DEC: Decrement button for bar 1 TOUCH_BAR1_INC: Increment button for bar 1 TOUCH_BAR2_DEC: Decrement button for bar 2 TOUCH_BAR2_INC: Increment button for bar 2 TOUCH_BAR3_DEC: Decrement button for bar 3 TOUCH_BAR3_INC: Increment button for bar 3 TOUCH_BACK: Back button

touch_callback	
Outline	Callback processing for the simple I^2C FIT module
Header	None
Declaration	void touch_callback(void)
Description	Callback processing for the simple I^2C FIT module
Parameters	None
Return Values	None

fill_clut_alpha_data	
Outline	Overwriting the alpha value of the color palette
Header	None
Declaration	static void fill_clut_alpha_data(uint8_t * p_clut_data)
Description	Write “0xFF” over the alpha value of the color palette specified by the parameter.
Parameters	uint8_t * p_clut_data: Pointer to the color palette
Return Values	None

draw_rectangle	
Outline	Rectangle drawing
Header	None
Declaration	static void draw_rectangle (glcdc_coordinate_t draw_point)
Description	Draws a rectangle (28 px × 28 px) on the coordinates of Graphic 2 specified by the parameter. The start point of the drawing is a point of “specified coordinates – 14” (half the value of the RECTANGLE_BLOCK_SIZE constant). This function is used when displaying the current levels of brightness, contrast, and gamma values.
Parameters	glcdc_coordinate_t draw_point: Coordinates to draw a rectangle
Return Values	None
Remarks	This function is used to draw a rectangle in the 256-color BMP image format.

bmp_image_load	
Outline	Reading image data from the SerialFlash
Header	None
Declaration	static void bmp_image_load (uint32_t image_data, uint8_t * p_dest_addr, uint8_t * p_clut_data)
Description	Distills the color palette data and image data from the bitmap image (256 colors) at the address specified in the SerialFlash and copy them to the specified address.
Parameters	uint32_t image_data: Start address of the bitmap image in the SerialFlash uint8_t * p_dest_addr: Location to copy the image data uint8_t * p_clut_data: Location to copy the color palette
Return Values	None
Remarks	When big endian is used, the endian must be converted before copying.

bmp_image_check	
Outline	Checking the image stored in the SerialFlash
Header	None
Declaration	bool bmp_image_check (void)
Description	Checks whether the image data is stored in the SerialFlash. The function reads the first 4 bytes where each image data is supposed to be stored and checks whether the read bytes match the start character code (“BM”) of the bitmap file.
Parameters	None
Return Values	true: Image data present false: No image data present

convert_endian	
Outline	Endian conversion
Header	None
Declaration	static void convert_endian (uint32_t *p_src_addr, uint32_t size)
Description	Converts the endianness of the specified data.
Parameters	uint32_t * p_src_addr: Pointer to the data to be converted uint32_t size: Size of data to be converted (multiple of 4 bytes)
Return Values	None
Remarks	This function is enabled when big endian is selected in the compiler option.

8.8.3 Functions (r_serial_flash_read.c)

serialflash_read_initialize	
Outline	Initializing the SerialFlash communication
Header	r_serial_flash_read.h
Declaration	void serialflash_read_initialize (void)
Description	Initializes the communication with the SerialFlash.
Parameters	None
Return Values	None

data_read	
Outline	Reading data from the SerialFlash
Header	r_serial_flash_read.h
Declaration	flash_spi_status_t data_read(uint8_t *p_dest_addr, uint32_t src_addr, uint32_t data_size)
Description	Reads the specified image data from the SerialFlash and places it at the specified address.
Parameters	uint8_t * p_dest_addr: Address of the image data to read uint32_t src_addr: Address of the image data stored in the SerialFlash uint32_t size: Size of the image data to read
Return Values	FLASH_SPI_SUCCESS: Processing completed successfully FLASH_SPI_ERR_PARAM: Parameter error FLASH_SPI_ERR_HARD: Hardware error FLASH_SPI_ERR_OTHER: Other errors

8.9 Process Flowcharts

This section describes process flowcharts in this application note.

8.9.1 Main Processing

In main processing, each peripheral module is initialized first. Then switch input or touch input on the RSK is checked regularly, and then the screen mode is changed or processing according to touch input is performed.

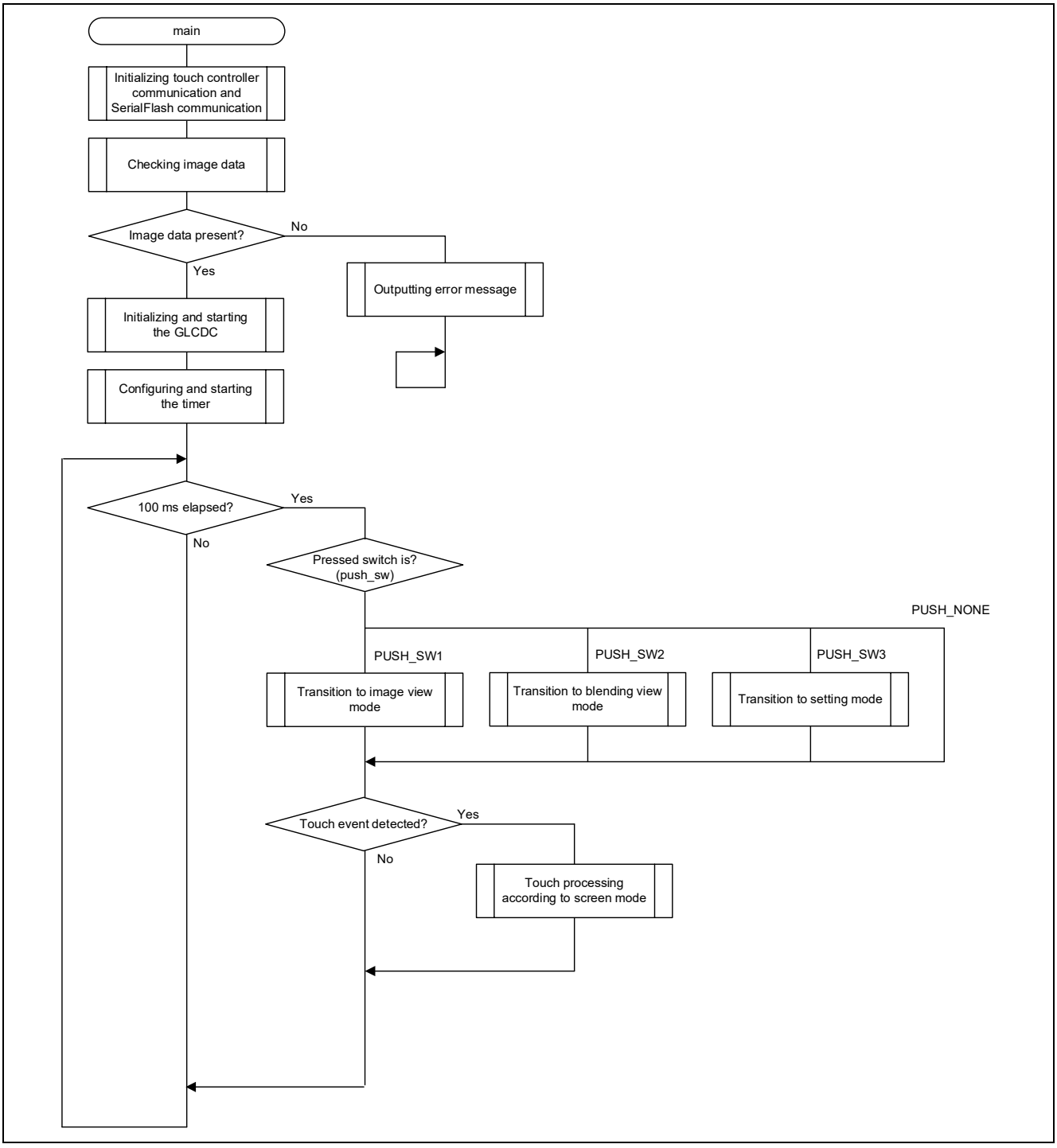


Figure 8.5 Flowchart of Main Processing

8.9.2 Initializing and Starting the GLCDC

Before starting the GLCDC, the image data is read from the SerialFlash and stored into the memory area from which the GLCDC reads images. Then the GLCDC is initialized with the R_GLCDC_Open function and GLCDC operation is started with the R_GLCDC_Control function. The image data then can be read from the memory area and displayed on the LCD panel.

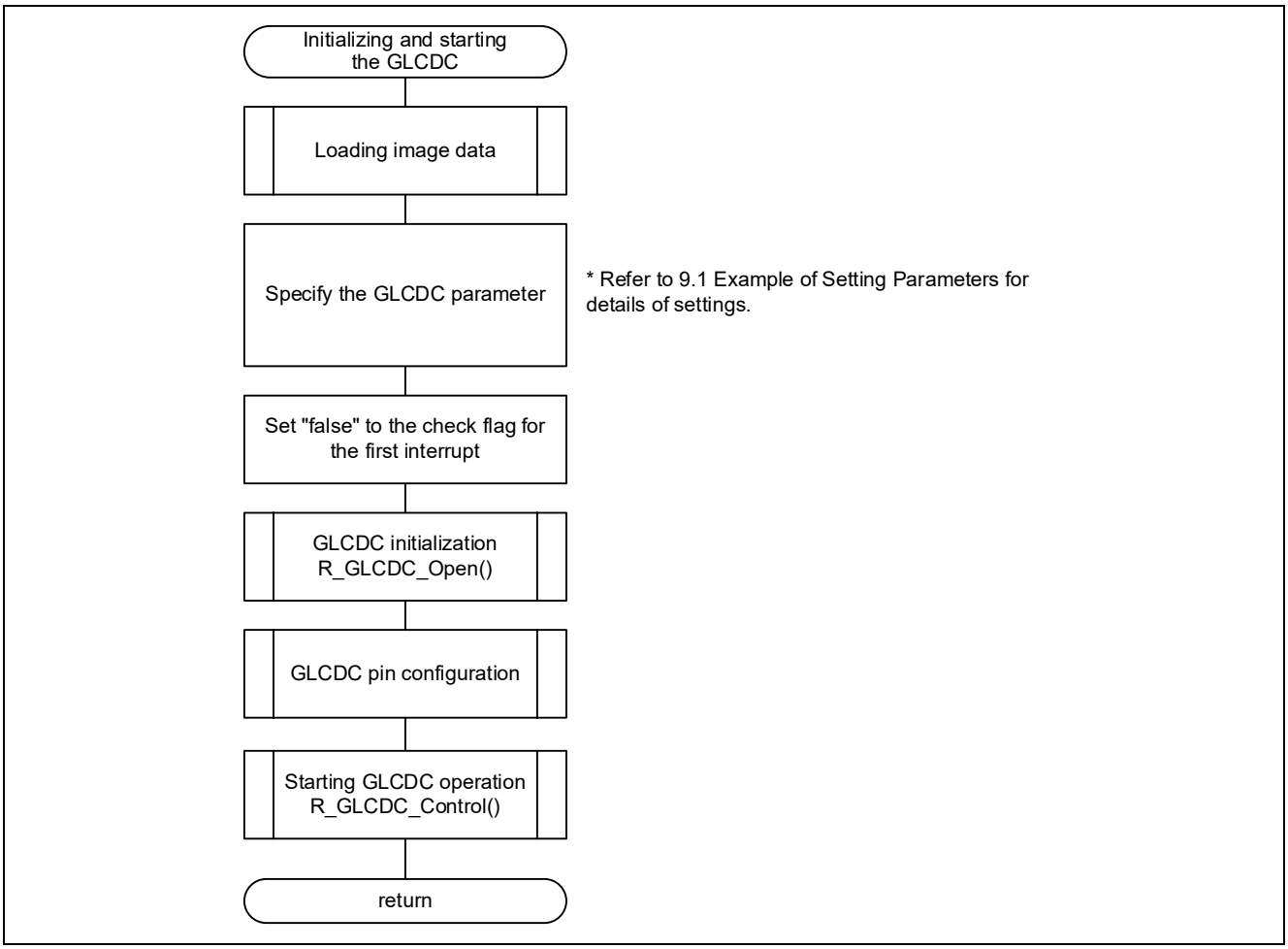


Figure 8.6 Flowchart of GLCDC Initialization

8.9.3 Mode Transition

In this processing, processing for each mode is performed depending on the mode to transition. The image data corresponding to each mode is read from the SerialFlash and written over the area where the current displayed data is placed. The GLCDC reads the image data repeatedly from the specified area while operating. Thus the image displayed on the LCD panel can be changed by overwriting the image data directly in the area where the GLCDC reads the image data. The GLCDC settings are updated and switched when necessary for each mode. The R_GLCDC_LayerChange function is used to change settings for alpha blending, chroma keying, or screen configuration of Graphic 1 and Graphic 2.

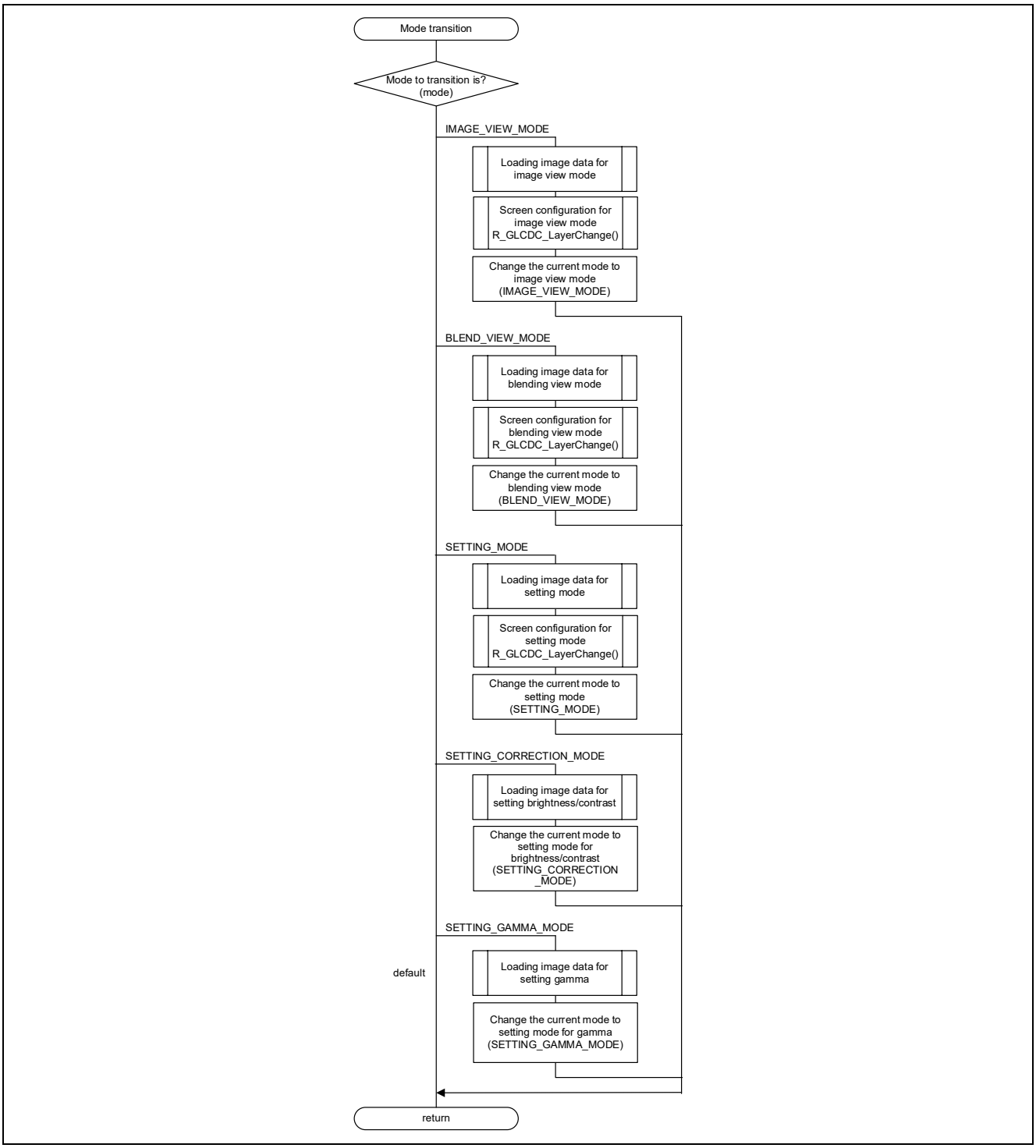


Figure 8.7 Flowchart of Mode Transition

8.9.4 Processing Responding to Touch Input on Each Screen Mode

1. Processing responding to touch input on image view mode

When touch input occurs in image view mode, the displayed image is switched. When a touch event is detected, the image data is read from the SerialFlash and the memory area where the current displayed image data is stored is overwritten with the read data. Then the displayed image is changed.

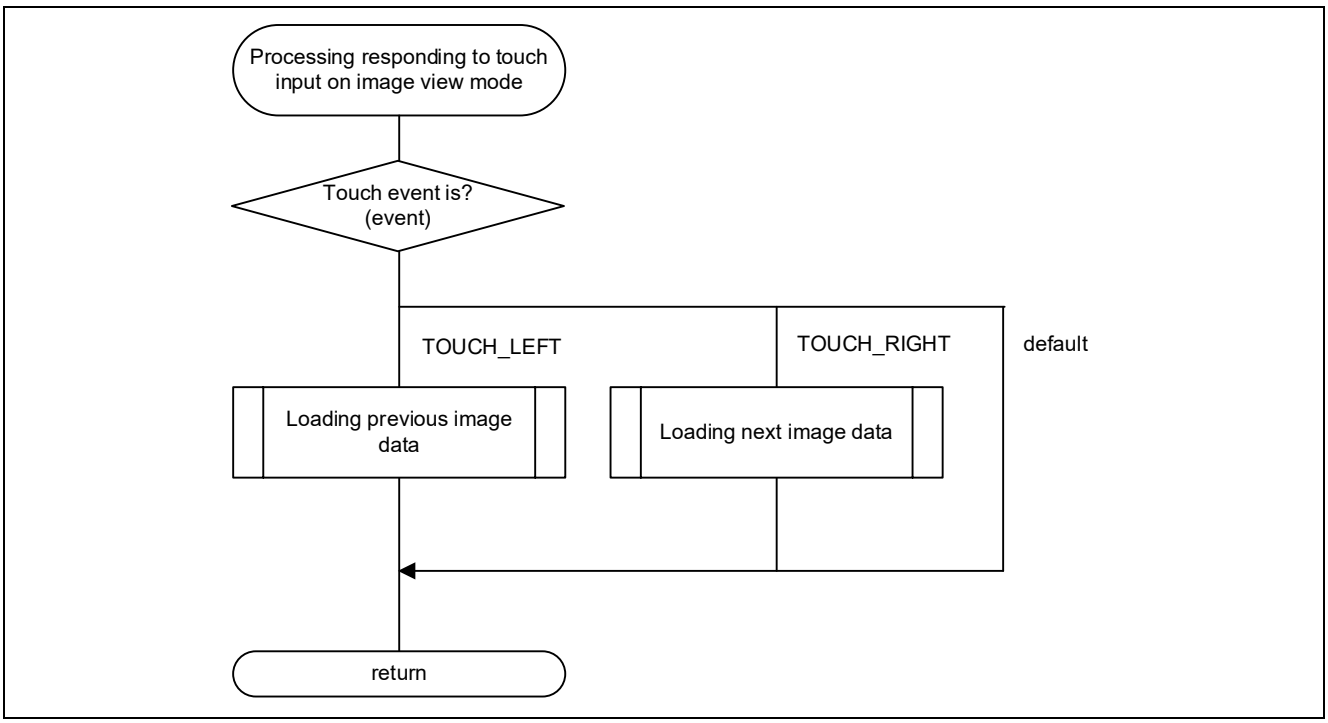


Figure 8.8 Flowchart of Processing Responding to Touch Input on Image View Mode

2. Processing responding to touch input on blending view mode

When touch input occurs in blending view mode, the image for blending is switched. When a touch event is detected, the image data is read from the SerialFlash and the memory area where the current displayed image data is stored is overwritten with the read data. Then the displayed image is changed.

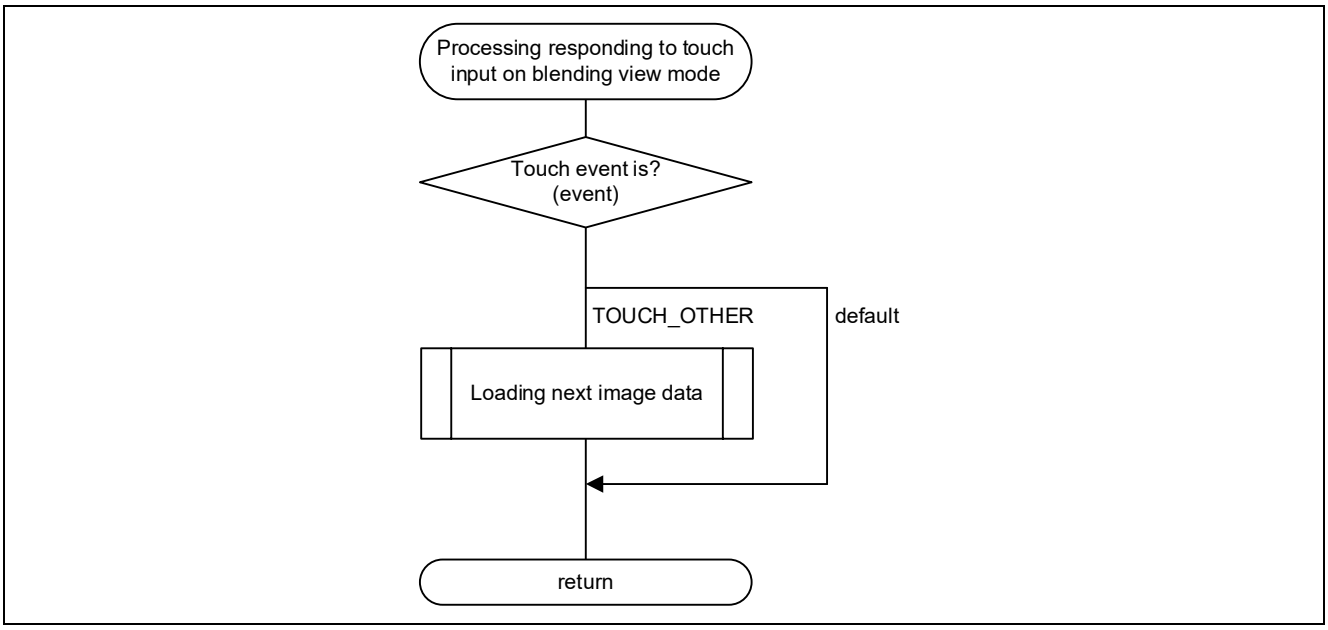


Figure 8.9 Flowchart of Processing Responding to Touch Input on Blending View Mode

3. Processing responding to touch input on setting mode

When touch input occurs in setting mode, mode transition is made and the displayed image is switched. When a touch event is detected, mode transition is made according to the event, the image for the mode is read from the SerialFlash, and the memory area where the current displayed image data is stored is overwritten with the read data. Then the displayed image is changed.

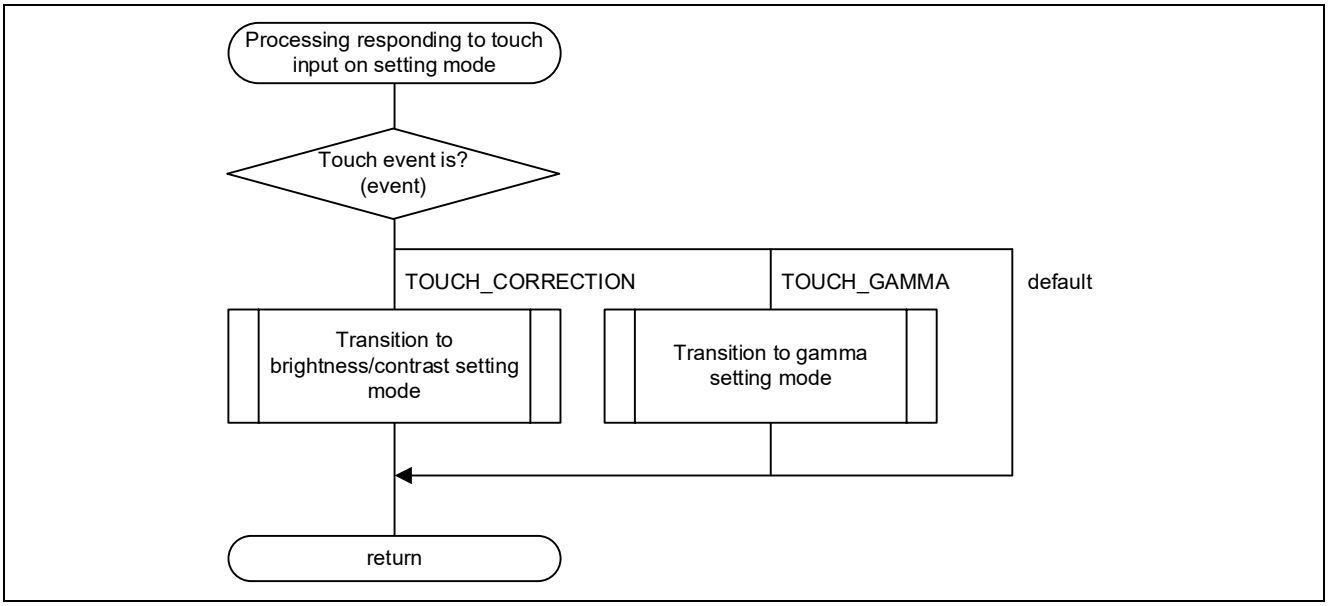


Figure 8.10 Flowchart of Processing Responding to Touch Input on Setting Mode

4. Processing responding to touch input on brightness/contrast setting mode

When touch input occurs in brightness/contrast setting mode, settings for alpha blending or brightness/contrast are changed. The change in the alpha blending value is reflected on the transition to blending view mode. Brightness and contrast values can be changed with the R_GLCDC_ColorCorrection function. The changes made are reflected to the screen after the function is executed. Then processing for transition to brightness/contrast setting mode (update display screen) is performed to reflect the changed setting to the horizontal bar on the screen.

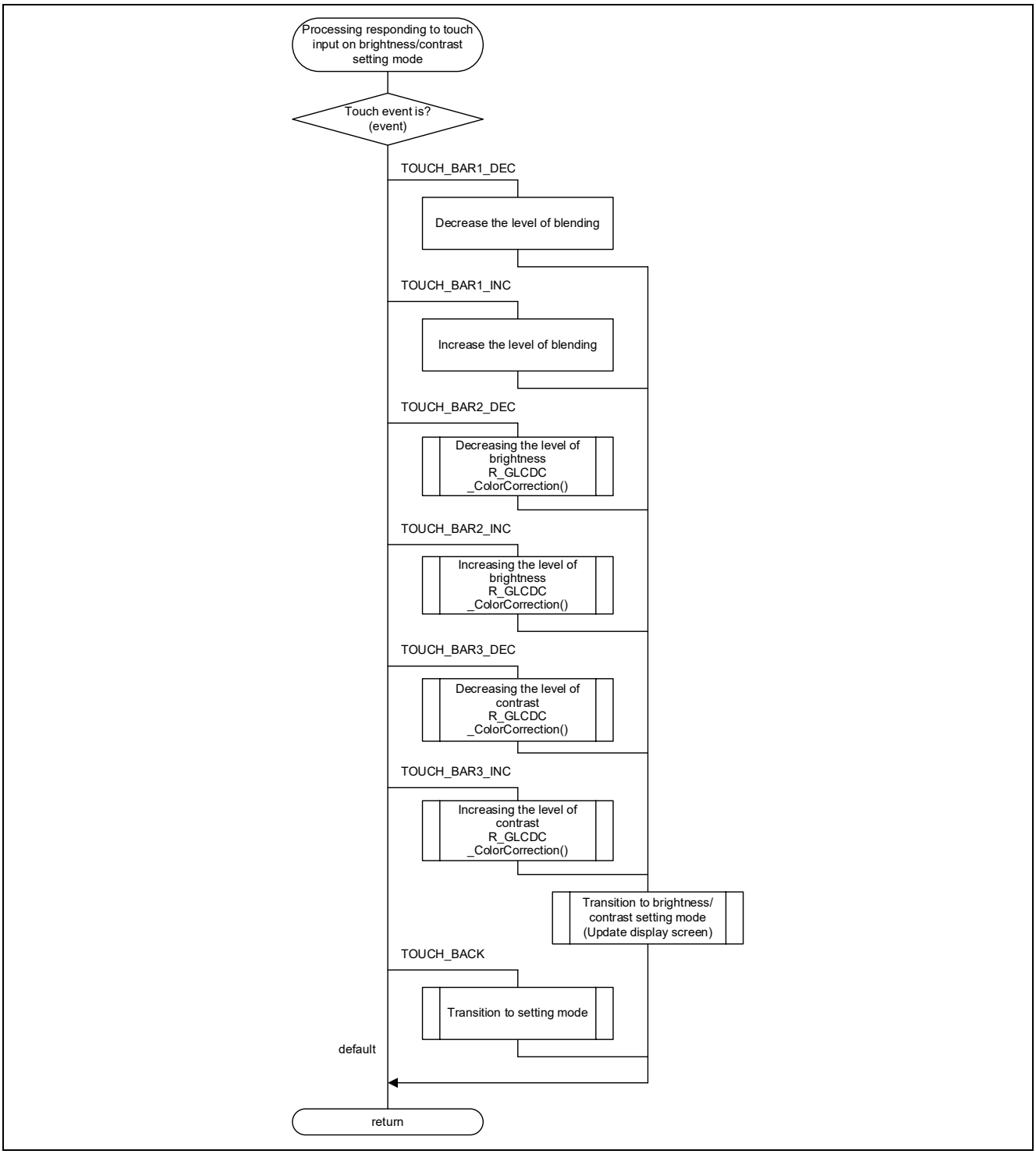


Figure 8.11 Flowchart of Processing Responding to Touch Input on Brightness/Contrast Setting Mode

5. Processing responding to touch input on gamma setting mode

When touch input occurs in gamma setting mode, RGB gamma values are changed. The gamma values can be changed with the R_GLCDC_ColorCorrection function. The change made is reflected to the screen after the function is executed. Then transition to gamma setting mode is made (update the display screen) to reflect the changed settings to the horizontal bars on the screen.

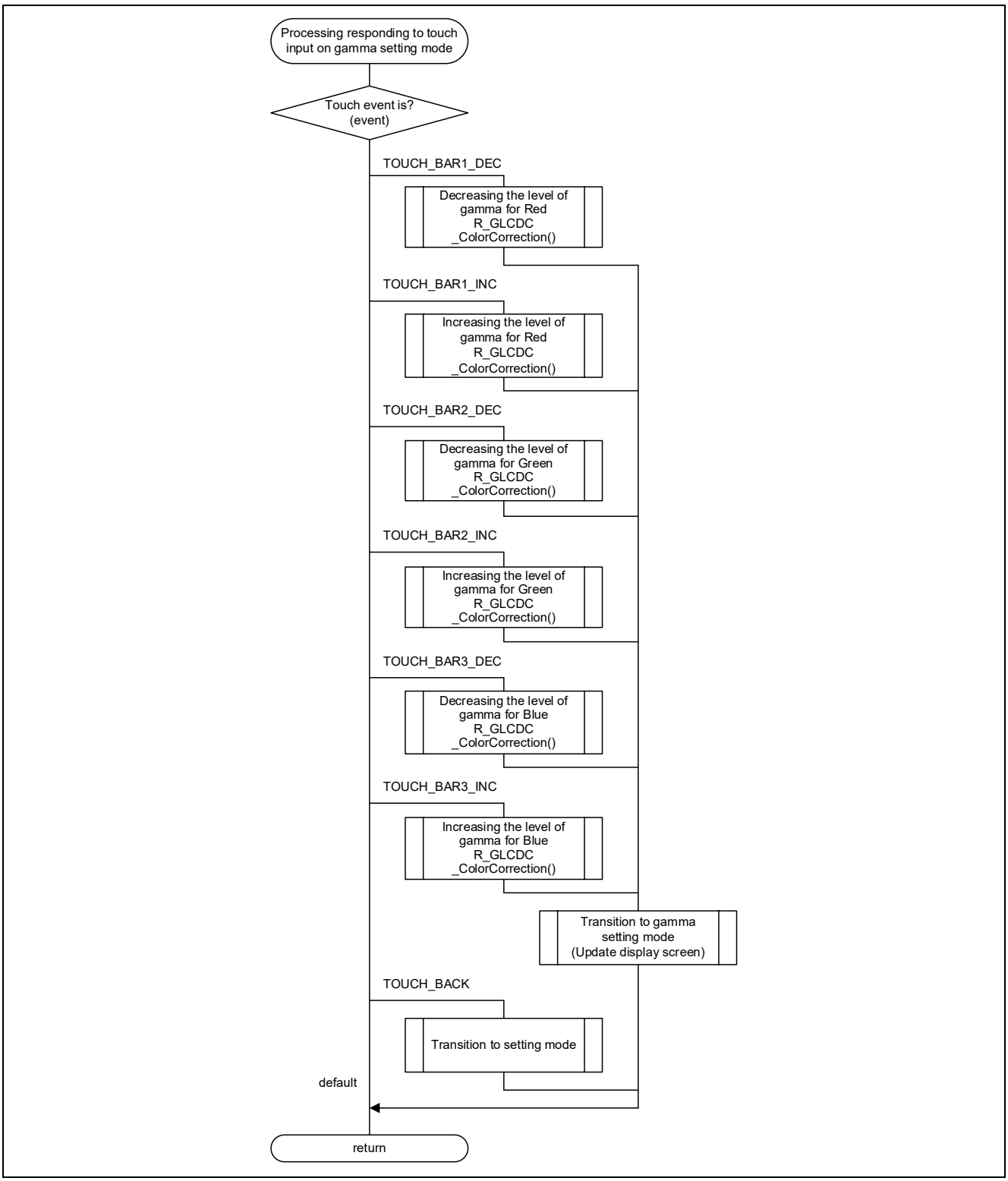


Figure 8.12 Flowchart of Processing Responding to Touch Input on Gamma Setting Mode

9. Appendices

9.1 Example of Setting Parameters

This section shows an example of arguments set to the glcdc_cfg_t p_cfg parameter in the R_GLCDC_Open function. The settings shown here are the same settings in the glcdc_initialize function in r_screen.c.

```
/* 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;
```

Figure 9.1 Example of Parameter Settings (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;
```

Figure 9.2 Example of Parameter Settings (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;
```

Figure 9.3 Example of Parameter Settings (3/3)

10. Importing a Project

The sample code is provided as the e² studio project. This section describes importing a project into the e² studio and CS+. After importing a project, confirm that the build settings and the debug settings are correct.

10.1 Importing a Project into the e² studio

Follow the steps below to import your project into the e² studio.
(Windows/dialogs may differ depending on the e² studio version used.)

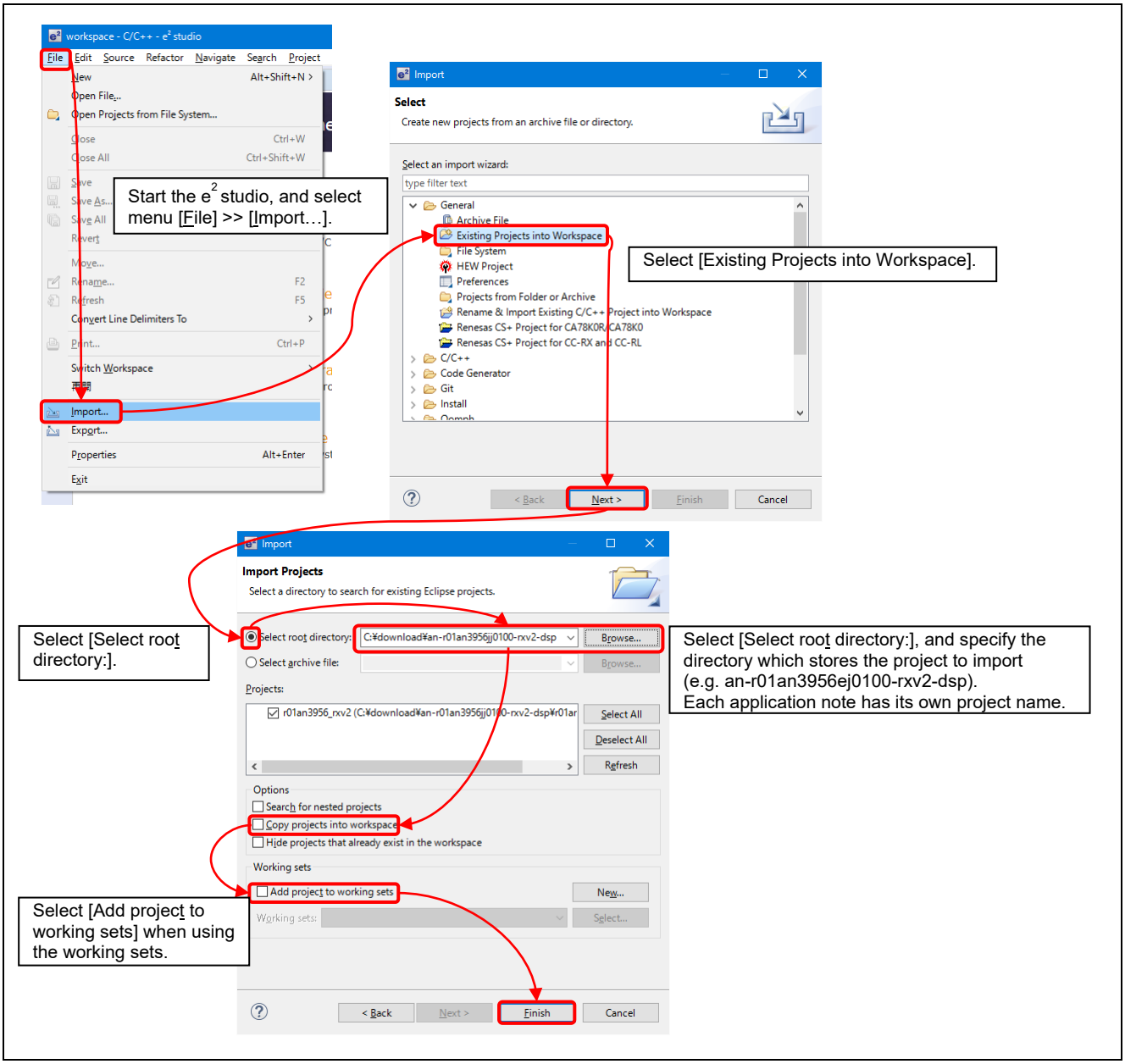


Figure 10.1 Importing a Project into the e² studio

10.2 Importing a Project into CS+

Follow the steps below to import your project into CS+.
(Windows/dialogs may differ depending on the CS+ version used.)

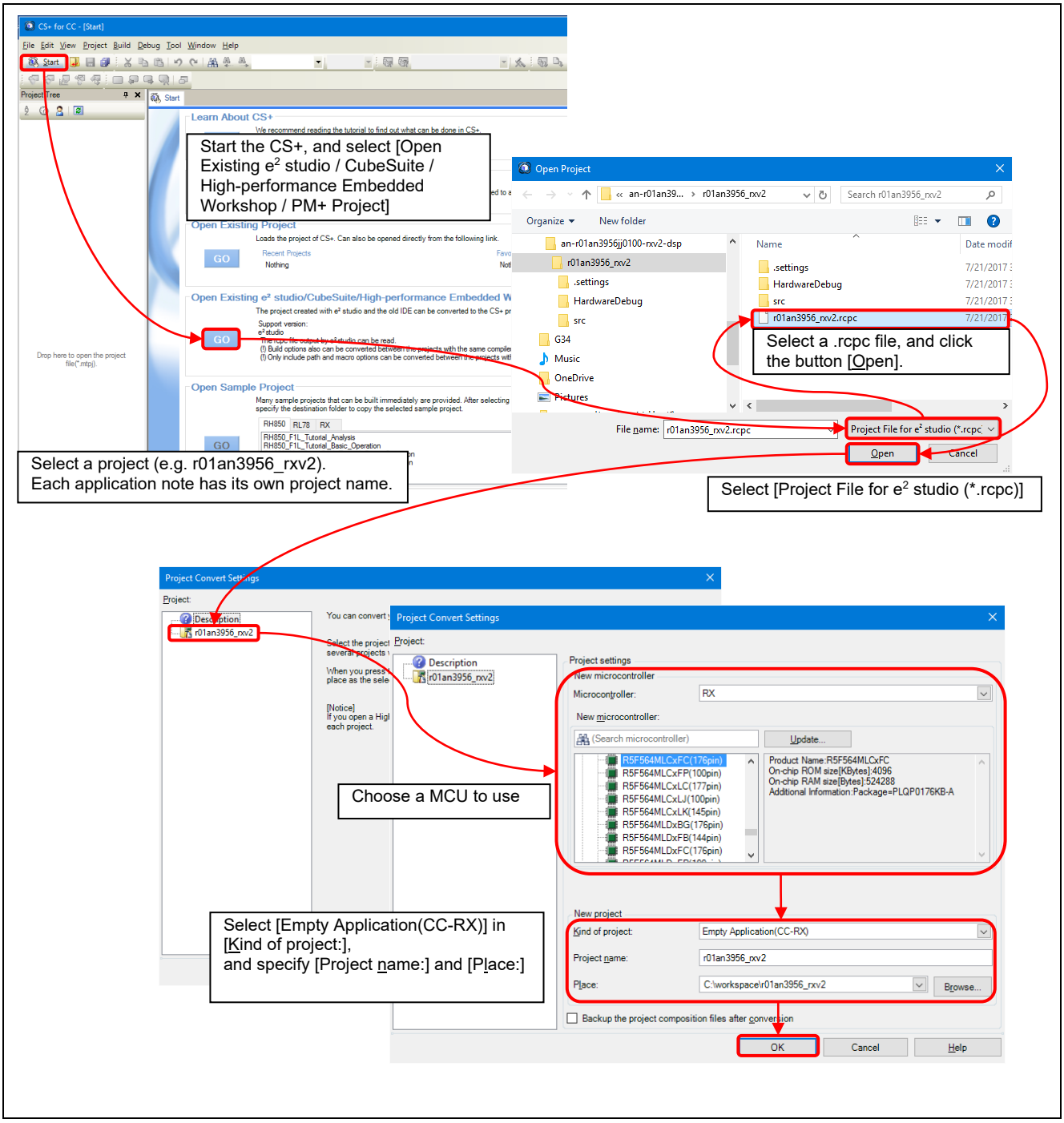


Figure 10.2 Importing a Project into CS+

11. Reference Document

User's Manual: Hardware
RX65N Group, RX651 Group User's Manual: Hardware (R01UH0590)
(The latest version can be downloaded from the Renesas Electronics website.)

Technical Update/Technical News
(The latest version can be downloaded from the Renesas Electronics website.)

User's Manual: Development Tools
RX Family CC-RX Compiler User's Manual (R20UT3248)
(The latest version can be downloaded from the Renesas Electronics website.)

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Oct. 20, 2017	—	First edition issued

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.3.0-1 November 2016)

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 6.0