

RX Family

emWin v6.26 Module Firmware Integration Technology

Introduction

This application note describes the emWin v.6.26 module which uses Firmware Integration Technology (FIT). This module is hereinafter referred to as “the emWin FIT module”.

The emWin FIT module is the modularized emWin (<https://www.segger.com/products/user-interface/emwin/add-ons/emwin-support-renesas-rx-mcu/>) by SEGGER by using FIT

When the emWin FIT module is used with the RX family (products supported by the FIT modules), mass production is possible with no license required. For details about the license, refer to “1.1 emWin FIT Module”.

For the details of “emWin” and GUI design tool, “AppWizard”, contact SEGGER (<https://www.segger.com/>).

This module is linked with QE for Display[RX], the development assistance tool for display, (<https://www.renesas.com/software-tool/qe-display-development-assistance-tool-display-applications>). When you use the emWin FIT module, we recommend using QE for Display[RX] together.

Applicable devices

- RX family

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

Target Compilers

- Renesas Electronics C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

For details of the confirmed operation contents of each compiler, refer to “2.9 Adding the FIT Module to Your Project”.

Related Documents

- Firmware Integration Technology User’s Manual (R01AN1833)
- RX Family Board Support Package Module Firmware Integration Technology (R01AN1685)

Contents

1. Overview.....	4
1.1 emWin FIT Module	4
1.2 Overview of emWin FIT module.....	4
1.3 Limitation	5
1.4 Structure of Product Files.....	6
1.5 API Overview.....	8
1.6 Software Configuration.....	10
2. API Information.....	11
2.1 Hardware Requirements	11
2.2 Software Requirements.....	11
2.3 Supported Toolchain	12
2.4 Header Files	12
2.5 Integer Types.....	12
2.6 Configuration while Compiling.....	13
2.7 Code Size	16
2.8 Parameter.....	18
2.9 Adding the FIT Module to Your Project	18
2.10 for, while, and do while Expressions	19
3. Functions Called from emWin	20
3.1 GUI_X_Config().....	20
3.2 LCD_X_Config ()	21
3.3 LCD_X_DisplayDriver ()	22
3.4 GUI_X_Init ().....	24
3.5 GUI_X_Delay ()	25
3.6 GUI_X_ExecIdle ()	26
3.7 GUI_X_GetTime ()	27
3.8 GUI_X_ErrorOut ()	28
3.9 GUI_X_Warn ().....	29
3.10 GUI_X_Log ()	30
3.11 GUI_X_InitOS ()	31
3.12 GUI_X_Unlock ()	32
3.13 GUI_X_Lock ().....	33
3.14 GUI_X_GetTaskId ().....	34
3.15 GUI_X_WaitEvent ().....	35
3.16 GUI_X_SignalEvent ()	36
3.17 GUI_X_WaitEventTimed ().....	37
3.18 PID_X_SetLayerIndex ()	38
3.19 PID_X_Init ().....	39

3.20	GUI_TOUCH_X_ActiveX ()	40
3.21	GUI_TOUCH_X_ActiveY ()	41
3.22	GUI_TOUCH_X_MeasureX ()	42
3.23	GUI_TOUCH_Y_MeasureY ()	43
3.24	APPW_X_FS_Init ()	44
4.	API Functions Called from the Application	45
4.1	R_EMWIN_GetBufferAddr()	45
4.2	R_EMWIN_GetD2 ()	46
4.3	R_EMWIN_EnableDave2D ()	47
4.4	R_EMWIN_DisableDave2D ()	48
4.5	R_EMWIN_GetDaveActive ()	49
4.6	R_EMWIN_GetVersion ()	50
4.7	_VSYNC_ISR ()	51
5.	Pin Setting	52
6.	Notation to implement the emWin FIT module	53
6.1	Selecting the library file	53
6.2	Setting the section when using RX65N	53
6.3	Setting the RAM size required for system operation	53
6.4	Setting the heap memory size when DRW2D is enabled	53
6.5	Setting the compile option	54
6.6	Image format	54
6.7	Setting multi-buffering when using AppWizard	54
6.8	When using an SPI (serial interface) LCD	54
6.9	Color depth setting when using an SPI (serial interface) LCD	54
6.10	When using the SPI (serial interface) touch function	54
6.11	Implementing to the environment exclusive of operation confirmed	55
7.	Sample Application	57
8.	Appendix	58
8.1	Confirmed Operation Environment	58
8.2	Troubleshooting	59
9.	Reference Documents	60
	Revision History	61

1. Overview

1.1 emWin FIT Module

emWin is a GUI library by Skegger. It allows GUIs to be easily designed with API-level programming, and combined use of emWin and various tools such as AppWizard improves development efficiency.

Generally, you need a license agreement with Segger when using emWin. However, when using the emWin FIT module with the RX family (products supported by the FIT modules), mass production is possible with no license required on condition that the default configuration is used.

Note, however, that if a display driver, functions, or library source codes that are not included in the emWin FIT module are required, a license agreement with Segger is required.

For details about the license agreement, contact Segger or the local agent EmbiTeck (<https://www.embitek.co.jp/>).

1.2 Overview of emWin FIT module

The emWin FIT module enables emWin to be easily implemented in a user's program with Smart Configurator by making emWIN (V.6.26) by SEGGER correspond to FIT. We will continue to support the upgraded version of emWin V6.26.

For the details of emWin, refer to the document below.

- emWin Graphic Library with Graphical User Interface User Guide & Reference Manual (<https://www.segger.com/downloads/emwin/UM03001>)

The emWin FIT module supports the following interfaces:

LCD interfaces:

- RGB (parallel interface)
Available on RX products with GLCDC, such as the RX65N group and RX72N group products
emWin provides control for the GUIDRV_LIN driver.
- SPI (serial interface)
Available on RX products with the RSPI or SCI (simple SPI mode)
emWin provides control for the GUIDRV_FlexColor driver.

Note: "GUIDRV_LIN" and "GUIDRV_FlexColor" are the names of display driver groups defined in emWin.

Touch panel interfaces:

- I²C
- SPI

1.3 Limitation

The emWin FIT module has the limitations mentioned below.

- Use of the DRW2D FIT module is recommended (DRW2D can be used only with the RGB (parallel interface) with GLCDC).
- OS: supports only FreeRTOS and OS-less
- Does not support emFILE or embOS by SEGGER
- Only the GUIDRV_LIN driver and GUIDRV_FlexColor driver are supported.
- Operation of LCDs has not been confirmed except for those described in “8.1 Confirmed Operation Environment”. To use an LCD without operation confirmed, refer to “6.11 Implementing to the environment exclusive of operation confirmed”.
- Setting color depth 1bpp (RGB by GLCDC only(parallel interface)) is prohibited in this version.
- When using the IAR C/C++ Compiler for Renesas RX, the SCI FIT module cannot be used to control the touch panel in this version. (Can be used with Renesas Electronics C/C++ Compiler Package for RX Family and GCC for Renesas RX)

1.4 Structure of Product Files

This product includes the files listed in Table 1.1 below.

Table 1.1 Structure of Product Files

File/Directory (Bold) Names	Description
r01an6452ej0100-rx-emwin.pdf	emWin FIT module Application Note (English)
r01an6452jj0100-rx-emwin.pdf	emWin FIT module Application Note (Japanese)
FITModules	Folder of the FIT module
r_emwin_rx_v6.26c_100.mdf	File to set the configuration of the emWin FIT module used by Smart Configurator
r_emwin_rx_v6.26c_100.xml	File to add the emWin FIT module to the project used by Smart Configurator
r_emwin_rx_v6.26c_100.zip	emWin FIT module (The contents are shown below)
r_config	Folder to store the config .h file for emWin FIT module
r_emwin_rx_config.h	Config .h file for emWin FIT module
r_emwin_rx	Folder to store the source code, documents, and tools of emWin FIT module
readme.txt	Explanation of overview and file structure of emWin FIT module
r_emwin_rx_if.h	Declaration .h file of emWin FIT module
doc	Folder to store documents of emWin FIT module
emWin_doc	Folder to store documents about emWin library provided by SEGGER GmbH
en	Folder to store emWin FIT module Application Note (English)
r01an6452ej0100-rx-emwin.pdf	emWin FIT module Application Note (English)
ja	Folder to store emWin FIT module Application Note (Japanese)
r01an6452jj0100-rx-emwin.pdf	emWin FIT module Application Note (Japanese)
Training	Sample program to use emWin library provided by SEGGER GmbH (Please refer to emWin_Training.pdf)
lib	Folder to store emWin library and source code which is interface block as the configuration for the emWin library
Config	Folder to store source code which is interface block as the configuration for the emWin library
APPW_X_NoFS.c	.c file which includes a function to use APPWizard (This file supports the project without file system.)
GUI_X_Ex.c	.c file which includes functions to use when an RTOS is used and not used
GUIConf.c	.c file which includes a function to allocate work memory and initialization of emWin library
GUIConf.h	.h file to show the configuration of emWin library
LCDConf_glcdc_if.c	.c file which includes functions to use and initialize the display driver of the emWin library and perform control processing by GLCDC and DRW2D
LCDConf_spi_if.c	.c file which includes functions to use and initialize the display driver of the emWin library and perform control processing by SPI (RSPI or simple SPI mode)
LCDConf_user_if.c	.c file which implements user defined processing If the emWin FIT module does not operate under control of preceding GLCDC or SPI, user defined processing must be implemented in this file.
LCDConf.h	.h file of declarations about display driver

	PIDConf.c	.c file which includes functions to use touch function and initialization
	PIDConf.h	.h file of declarations about touch function
	GUI	Folder to store library files and header files of emWin library
	src	Folder to store source code which is not interface block as the configuration for the emWin library
	r_emwin_rx_if.c	.c file which includes an own function of emWin FIT module
	r_emwin_rx_pid_iic_if.c	.c file which has the IIC interface function with the touch panel
	r_emwin_rx_pid_spi_if.c	.c file which has the SPI interface function with the touch panel
	r_emwin_rx_pid_user_if.c	.c file which implements user defined processing If the emWin FIT module does not operate under control of preceding IIC or SPI, user defined processing must be implemented in this file.
	r_emwin_rx_private.h	.h file to be used by emWin FIT module internally
	tool	Folder to store tools for emWin library which includes installer of AppWizard (Please refer to doc/Training/emWin_Training.pdf for detail)

1.5 API Overview

The tables below list the API functions included in the emWin FIT module. Table 1.2 Functions which emWin calls from the Inside lists the functions which emWin calls from the inside. Table 1.3 lists the API functions which are called from the application.

For the details, refer to “3.Functions Called from emWin” and “4.API Functions Called from the Application.”

Table 1.2 Functions which emWin calls from the Inside

Function	Description
GUI_X_Config	Registers memory block which is used in emWin memory management system
LCD_X_Config	Initializes LCD and device driver
LCD_X_DisplayDriver	Calls back function of display driver
GUI_X_Init	Initializes necessary hardware
GUI_X_Delay	Waits the specified time
GUI_X_ExecIdle	Called from Window Manager when GUI is not up to date and there is no content to be processed
GUI_X_GetTime	Current system time is obtained in integer in milliseconds.
GUI_X_ErrorOut	When a fatal error occurs, called from emWin with an error string as an input
GUI_X_Warn	When a warning occurs, called from emWin with a warning string as an input.
GUI_X_Log	When a message occurs, called from emWin with a message string as an input.
GUI_X_InitOS	When using under multitask environment, generates semaphore or mutex
GUI_X_Unlock	When using under multitask environment, unlocks GUI
GUI_X_Lock	When using under multitask environment, locks GUI
GUI_X_GetTaskId	When using under multitask environment, obtains task ID
GUI_X_WaitEvent	When using under multitask environment, executes the waiting for an event
GUI_X_SignalEvent	When using under multitask environment, executes event notification
GUI_X_WaitEventTimed	When using under multitask environment, executes the waiting for an event during the specified period
PID_X_SetLayerIndex	Sets layer number
PID_X_Init	Initializes Pointer Input Device
GUI_TOUCH_X_ActiveX	Enables voltage measurement of x axis of Touch IC
GUI_TOUCH_X_ActiveY	Enables voltage measurement of y axis of Touch IC
GUI_TOUCH_X_MeasureX	Returns the voltage measurement result of x axis obtained from Touch IC
GUI_TOUCH_Y_MeasureY	Returns the voltage measurement result of y axis obtained from Touch IC
APPW_X_FS_Init	Initializes the file system access of AppWizard

Table 1.3 API Functions Called from Application

Function	Function Description
R_EMWIN_GetBufferAddr	Obtains the address of frame buffer
R_EMWIN_GetD2	Obtains the handle of Dave2D function
R_EMWIN_EnableDave2D	Turns the Dave2D function into the enable state
R_EMWIN_DisableDave2D	Turns the Dave2D function into the operation inhibition state
R_EMWIN_GetDaveActive	Obtains the operating state of Dave2D function
R_EMWIN_GetVersion	Obtains the version of emWin
_VSYNC_ISR()	Performs Vsync interrupt processing (Assumes callback function of GLCDC)

1.6 Software Configuration

The application which uses the emWin FIT module has a software configuration shown in Figure 1.1.

Application uses the emWin FIT module.

When an RGB (parallel interface) LCD is used, the emWin FIT module uses the DRW2D FIT module to create a figure, and then uses the GLCD FIT module to display the figure on the LCD.

When an SPI (serial interface) LCD is used, the emWin FIT module uses the RSPI or SCI FIT module to display figures on the LCD.

Touch panel information is controlled by using the SCI-I2C FIT module, RSPI FIT module, or SCI FIT module.

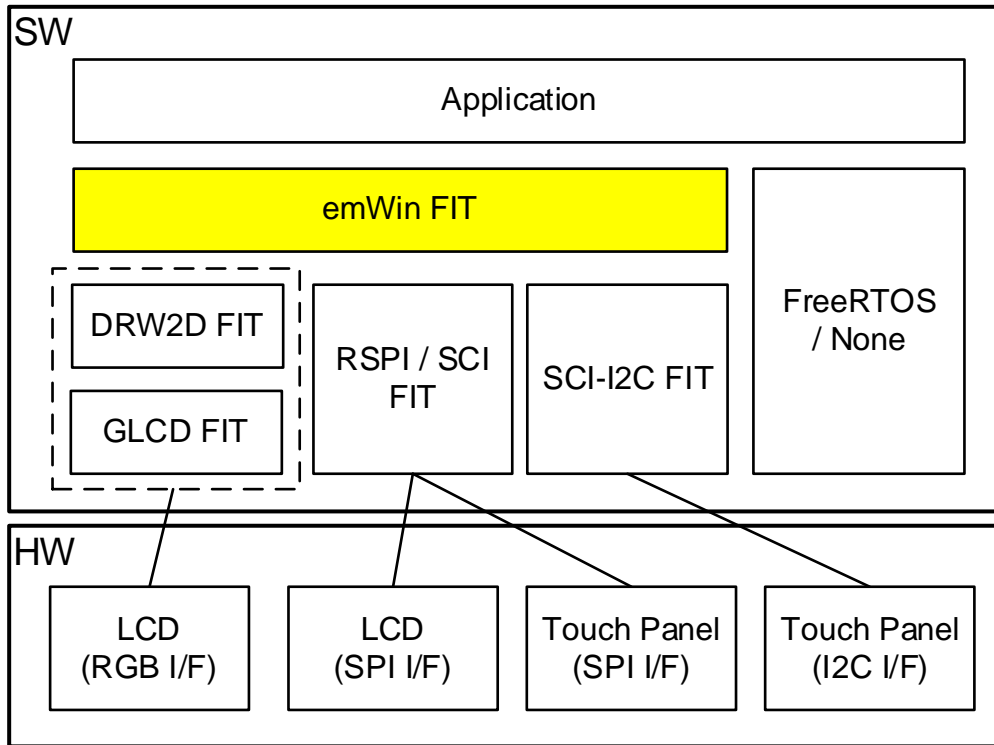


Figure 1.1 Software Configuration

2. API Information

This FIT module has been confirmed to operate under the following conditions.

2.1 Hardware Requirements

The MCU used must support the following functions:

In all cases:

- GPIO
- CMT

When using an SPI (serial interface) LCD:

One of the following is required.

- SCI
- RSPI

When using an RGB (parallel interface) LCD:

- DMAC
- GLCDC
- DRW2D

When using the touch function:

- SCI
- RSPI

2.2 Software Requirements

This driver is dependent upon the following FIT module:

In all cases:

- Board support package (r_bsp) Rev.7.20 or later
- GPIO (r_gpio_rx) Rev.4.50 or later
- CMT (r_cmt_rx) Rev.5.20 or later

When using an SPI (serial interface) LCD:

One of the following is required.

- SCI (r_sci_rx) Rev.4.40 or later
- RSPI (r_rspi_rx) Rev.3.04 or later

When using an RGB (parallel interface) LCD:

- DMAC (r_dmaca_rx) Rev.2.90 or later
- Graphic LCD controller (r_glcdc_rx) Rev.1.50 or later
- DRW2D driver (r_drw2d_rx) Rev.1.11 or later

When using the touch function:

One of the following is required.

- SCI (simple I2C mode) (r_sci_iic_rx) Rev.2.50 or later
- SCI (r_sci_rx) Rev.4.40 or later
- RSPI (r_rspi_rx) Rev.3.04 or later

2.3 Supported Toolchain

This FIT module has been confirmed to work with the toolchain listed in 8.1 Confirmed Operation Environment.

2.4 Header Files

All API calls and their supporting interface definitions are located in r_emwin_rx_if.h.

2.5 Integer Types

This driver uses ANSI C99. These types are defined in stdint.h

2.6 Configuration while Compiling

The configuration option settings of the emWin FIT module are performed in `r_emwin_rx_config.h`. The option names and setting values are listed in the table below:

Configuration options in <code>r_emwin_rx_config.h</code>	
EMWIN_GUI_NUM_BYTES	Specifies the maximum memory size used in GUI. The required memory size varies depending on the system to be developed. If the system does not operate normally with the default value, increase the memory size.
EMWIN_XSIZE_PHYS	Specifies the horizontal LCD size.
EMWIN_YSIZE_PHYS	Specifies the vertical LCD size.
EMWIN_BITS_PER_PIXEL	Selects color depth value. - When this is set to 1, color depth is set as 1 bpp. Then, DRW2D FIT module can not be used. (This setting is prohibited in this version) - When this is set to 4, color depth is set as 4 bpp. Then, DRW2D FIT module can not be used. - When this is set to 8, color depth is set as 8 bpp. Then, DRW2D FIT module can not be used. - When this is set to 16, color depth is set as 16 bpp. This value is recommended in this emWin FIT module. - When this is set to 24, color depth is set as 24 bpp. - When this is set to 32, color depth is set as 32 bpp.
EMWIN_DISPLAY_ORIENTATION	Selects the LCD orientation from the defined values. - When this is set to ORIENTATION_0, the displayed image is not rotated. - When this is set to ORIENTATION_CW, the displayed image is rotated clockwise 90 degrees. - When this is set to ORIENTATION_180, the displayed image is rotated 180 degrees. - When this is set to ORIENTATION_CCW, the displayed image is rotated counter-clockwise 90 degrees.
EMWIN_LCD_IF	Selects the interface to display images on the LCD. - When this is set to LCD_IF_GLCDC, GLCDC is used. - When this is set to LCD_IF_RSPI, the RSPI is used. - When this is set to LCD_IF_SCI_SPI, SCI (simple SPI mode) is used. - When this is set to LCD_IF_OTHER, you need to implement the interface to use.
EMWIN_USE_TOUCH	Selectable whether to use the touch function. - When this is set to 0, the touch function is not used. - When this is set to 1, the touch function is used.
Settings that are valid when using an RGB (parallel interface) LCD (EMWIN_LCD_IF = LCD_IF_GLCDC)	
EMWIN_NUM_BUFFERS	Specifies number of buffers. emWin FIT module supports 1 ~ 3. When more frame buffers are needed, please implement additionally.
EMWIN_GUI_FRAME_BUFFER1	Specifies start address of the frame buffer 1 to display image.
EMWIN_GUI_FRAME_BUFFER2	Specifies start address of the frame buffer 2 to display image.
EMWIN_GUI_FRAME_BUFFER3	Specifies start address of the frame buffer 3 to display image.
EMWIN_USE_DRW2D	Selectable whether to use DRW2D. - When this is set to 0, DRW2D is not used. - When this is set to 1, DRW2D is used.
EMWIN_DMACH_NUMBER	Specifies channel number of DMACH to use in data transfer between frame buffers.

EMWIN_INIT_DMACH	Selectable whether to initialize DMACH in emWin FIT module. - When this is set to 0, DMACH is not initialized in emWin FIT module. - When this is set to 1, DMACH is initialized in emWin FIT module.
EMWIN_USE_DISP_SIGNAL_PIN	Selectable whether to use the LCD reset pin. - When this is set to 0, the LCD reset pin is not used. - When this is set to 1, the LCD reset pin is used.
EMWIN_DISP_SIGNAL_PIN	Specifies the GPIO pin to be used as the LCD reset pin. To set this configuration, use the enumerated-type <code>gpio_port_pin_t</code> member in the GPIO FIT module.
EMWIN_USE_BACKLIGHT_PIN	Selectable whether to use the LCD backlight pin. - When this is set to 0, the LCD backlight pin is not used. - When this is set to 1, the LCD backlight pin is used.
EMWIN_BACKLIGHT_PIN	Specifies the GPIO pin to be used as the LCD backlight pin. To set this configuration, use the enumerated-type <code>gpio_port_pin_t</code> member in the GPIO FIT module.
Settings that are valid when using an SPI (serial interface) LCD (EMWIN_LCD_IF = LCD_IF_RSPI or LCD_IF_SCI_SPI)	
EMWIN_LCD_IF_NUMBER	Specifies the channel number of the interface to be used for the LCD display.
EMWIN_LCD_DRIVER_IC	Selects the product number of the LCD controller mounted on the LCD from the defined values. - When this is set to <code>LCD_DRV_IC_ST7715</code> , the ST7715 series is subject to control. - When this is set to <code>LCD_DRV_IC_ILI9341</code> , the ILI9341 series is subject to control. - When this is set to <code>LCD_DRV_IC_OTHER</code> , you need to implement the control code of the IC mounted on the LCD to be used.
EMWIN_LCD_BAUDRATE	Specifies the baud rate of the interface to be used for the LCD display. The maximum value that can be set depends on the LCD controller and RX product to be used. Refer to the data sheet of the LCD controller and the user's manual of the RX product.
EMWIN_GUI_USE_CACHE	Selectable whether to use the cache. - When this is set to 0, the cache is not used. - When this is set to 1, the cache is used. Using the cache enables high-speed internal processing. However, using the cache requires memory capacity sufficient for the LCD screen size. Therefore, allocate additional memory for one frame by using <code>EMWIN_GUI_NUM_BYTES</code> . Required memory capacity (bytes): LCD screen size x color depth For an LCD for which displayed memory cannot be read, graphic display without read is possible by enabling the cache or by using the memory device function. Note that the cache cannot be used when the color depth is set to 24. In such a case, select 0.
EMWIN_USE_DISP_SIGNAL_PIN	Selectable whether to use LCD reset pin. - When this is set to 0, LCD reset pin is not used. - When this is set to 1, LCD reset pin is used.
EMWIN_DISP_SIGNAL_PIN	Specifies GPIO pin to use as LCD reset pin. To set this configuration, use <code>gpio_port_pin_t</code> member in GPIO FIT module.
EMWIN_USE_BACKLIGHT_PIN	Selectable whether to use LCD backlight pin. - When this is set to 0, LCD backlight pin is not used. - When this is set to 1, LCD backlight pin is used.

EMWIN_BACKLIGHT_PIN	Specifies GPIO pin to use as LCD backlight pin. To set this configuration, use <code>gpio_port_pin_t</code> member in GPIO FIT module.
EMWIN_USE_DATA_CMD_PIN	Selectable whether to use the LCD data/command pin. - When this is set to 0, the LCD data/command pin is not used. - When this is set to 1, the LCD data/command pin is used.
EMWIN_DATA_CMD_PIN	Specifies the GPIO pin to be used as the LCD data/command pin. To set this configuration, use the enumerated-type <code>gpio_port_pin_t</code> member in the GPIO FIT module.
EMWIN_USE_LCD_CS_PIN	Selectable whether to use the LCD CS pin. - When this is set to 0, the LCD CS pin is not used. - When this is set to 1, the LCD CS pin is used.
EMWIN_LCD_CS_PIN	Specifies the GPIO pin to be used as the LCD CS pin. To set this configuration, use the enumerated-type <code>gpio_port_pin_t</code> member in the GPIO FIT module.
Settings that are valid when using the touch function	
EMWIN_TOUCH_IF	Specifies interface to use in touch function - When this is set to <code>TOUCH_IF_SCI_IIC</code> , SCI-I ² C is used. - When this is set to <code>TOUCH_IF_RSPI</code> , the RSPI is used. - When this is set to <code>TOUCH_IF_SCI_SPI</code> , SCI (simple SPI mode) is used. - When this is set to <code>TOUCH_IF_OTHER</code> , please implement interface to use. This option must be set when using the RSPI or SCI (simple SPI mode). For details, refer to "6.10 When using the SPI (serial interface) touch function".
EMWIN_TOUCH_IF_NUMBER	Specifies channel number of touch interface to use in communication to touch panel.
EMWIN_TOUCH_BAUDRATE	Specifies the baud rate of the interface to be used in the touch function. The maximum value that can be set depends on the touch controller or RX product to be used. Refer to the data sheet of the touch controller and the user's manual of the RX product.
EMWIN_USE_MULTITOUCH	Selectable whether to use multi-touch function. - When this is set to 0, multi touch function is not used. - When this is set to 1, multi touch function is used. The multi-touch function cannot be used if the RSPI or SCI (simple SPI mode) is selected as the interface to be used in the touch function or if the controller does not support the multi-touch function. In such cases, set 0.
EMWIN_MAX_NUM_TOUCHPOINTS	Specifies the maximum number of touch panel points. The value is used when the multi-touch function is used.
EMWIN_SLAVE_ADDRESS	Specifies slave address of touch panel. This setting is valid when I ² C is selected as the interface to be used in the touch function.
EMWIN_USE_TOUCH_IC_RESET_PIN	Selectable whether to use LCD touch IC reset pin. - When this is set to 0, LCD touch IC reset pin is not used. - When this is set to 1, LCD touch IC reset pin is used.
EMWIN_TOUCH_IC_RESET_PIN	Specifies GPIO pin to use as LCD touch IC reset pin. To set this configuration, use <code>gpio_port_pin_t</code> member in GPIO FIT module.
EMWIN_USE_TOUCH_CS_PIN	Selectable whether to use the CS pin of the touch panel. - When this is set to 0, the CS pin of the touch panel is not used. - When this is set to 1, the CS pin of the touch panel is used.
EMWIN_TOUCH_CS_PIN	Specifies the GPIO pin to be used as CS pin of the touch panel. To set this configuration, use the enumerated-type <code>gpio_port_pin_t</code> member in the GPIO FIT module.

2.7 Code Size

The sizes of ROM, RAM and maximum stack usage of the emWin FIT module are listed below.

The ROM (code and constants) and RAM (global data) sizes are determined by the build-time configuration options described in “2.6,Configuration while Compiling”

The values in the table below are confirmed under the following conditions

Module Revision: emWin Rev6.26c FIT Rev1.00

Compiler Version: Renesas Electronics C/C++ Compiler Package for RX Family V3.04.00

(The option of “-lang = c99” is added to the default settings of the integrated development environment.)

GCC for Renesas RX 8.03.00.202202

(The option of “-std=gnu99” is added to the default settings of the integrated development environment)

IAR C/C++ Compiler for Renesas RX version 4.20.1

(The default settings of the integrated development environment)

Configuration options: Default settings

ROM, RAM and Stack Code Sizes				
Device	Category	Memory Used		
		Renesas Compiler	GCC	IAR Compiler
RX130	ROM	91351 bytes	121799 bytes	32508 bytes
	RAM	3442 bytes	3453 bytes	2526 bytes
	Stack	664 bytes	-	-
[Configuration Options] EMWIN_GUI_NUM_BYTES=4 EMWIN_LCD_IF = LCD_IF_RSPI EMWIN_USE_TOUCH = 1 EMWIN_TOUCH_IF = TOUCH_IF_SCI_SPI				
RX231	ROM	90350 bytes	119437 bytes	26462 bytes
	RAM	3282 bytes	3309 bytes	1404 bytes
	Stack	664 bytes	-	-
[Configuration Options] EMWIN_GUI_NUM_BYTES=4 EMWIN_LCD_IF = LCD_IF_SCI_SPI EMWIN_USE_TOUCH = 0				
RX65N	ROM	106749 bytes	144068 bytes	34464 bytes
	RAM	3899 bytes	3929 bytes	2932 bytes
	STACK	776 bytes	-	-

ROM, RAM and Stack Code Sizes				
Device	Category	Memory Used		
		Renesas Compiler	GCC	IAR Compiler
[Configuration Options] EMWIN_GUI_NUM_BYTES=4 EMWIN_LCD_IF = LCD_IF_GLCDC EMWIN_USE_DRW2D = 1 EMWIN_USE_TOUCH = 1 EMWIN_TOUCH_IF = TOUCH_IF_SCI_IIC EMWIN_USE_MULTITOUCH = 0				
RX72N	ROM	106814 bytes	144074 bytes	34580 bytes
	RAM	3899 bytes	3929 bytes	2932 bytes
	STACK	776 bytes	-	-
[Configuration Options] EMWIN_GUI_NUM_BYTES=4 EMWIN_LCD_IF = LCD_IF_GLCDC EMWIN_USE_DRW2D = 1 EMWIN_USE_TOUCH = 1 EMWIN_TOUCH_IF = TOUCH_IF_SCI_IIC EMWIN_USE_MULTITOUCH = 0				

RAM size includes the value of EMWIN_GUI_NUM_BYTES. The RAM size listed in this table is the value when EMWIN_GUI_NUM_BYTES is set to 4 (minimum).

2.8 Parameter

This section describes the parameter structure used by the API functions in this module. The structure is located in `r_emwin_rx_if.h` as are the prototype declarations of API functions

2.9 Adding the FIT Module to Your Project

The emWin FIT module must be added to each project in which it is used. Renesas recommends the methods in (1) to (3) that use the Smart Configurator. Note, however, that QE for Display[RX] cannot be linked in CS+ and IAREW and you must specify the required settings.

- (1) Adding the FIT module to your project using the Smart Configurator in e² studio
By using the Smart Configurator in e² studio, the FIT module is automatically added to your project. Refer to the application note, "RX Smart Configurator User's Guide: e² studio (R20AN0451)" for details
Note : When there are emWin FIT modules with other versions in the directory to store downloaded FIT modules, Smart Configurator may not add the emWin FIT module precisely. Please store the latest emWin FIT module and do not leave other emWin FIT modules in the directory.
- (2) Adding the FIT module to your project using the Smart Configurator in CS+
By using the stand-alone version of Smart Configurator in CS+, the FIT module is automatically added to your project. For details, refer to the application note, "RX Smart Configurator User's Guide: CS+ (R20AN0470)".
- (3) Adding the FIT module to your project using the Smart Configurator in IAREW
By using the stand-alone version of Smart Configurator, the FIT module is automatically added to your project. For details, refer to the application note, "RX Smart Configurator User's Guide: IAREW (R20AN0535)".

2.10 for, while, and do while Expressions

This module uses *for*, *while*, and *do while* expressions (loop processing) for standby states such as waiting for register values to be updated. These instances of loop processing are indicated by the keyword `WAIT_LOOP` in the comments. Therefore, if you wish to incorporate failsafe processing into the instances of loop processing, you can locate them in the code by searching for the keyword `WAIT_LOOP`.

An example code listing is shown below.

```
Example of a while expression:
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}

Example of a for expression:
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}

Example of a do while expression:
/* Reset completion waiting */
do
{
    reg = phy_read(ether_channel, PHY_REG_CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

3. Functions Called from emWin

3.1 GUI_X_Config()

This function is a function to register memory block used in the memory management system of emWin

Format

void GUI_X_Config(void)

Parameters

None

Return Values

None

Properties

Prototyped in GUI.h

Description

Used to register memory block which is used in the memory management system of emWin

In the emWin FIT module, assigns memory by using GUI block function.

Reentrant

- No

3.2 LCD_X_Config ()

This function is a function to initialize LCD and device drivers.

Format

void LCD_X_Config(void)

Parameters

None

Return Values

None

Properties

Prototyped in LCD.h

Description

Used to initialize LCD and device drivers

The emWin FIT module uses a GUI block function to initialize the LCD. If GLCDC is selected for the interface, this function also initializes the DRW2D FIT module.

Reentrant

- No

3.3 LCD_X_DisplayDriver ()

This function is the callback function of display driver.

Format

```
int LCD_X_DisplayDriver(  
    unsigned layer_index,  
    unsigned cmd,  
    void * p_data  
)
```

Parameters

layer_index	Input	Layer number
cmd	Input	Executed command
p_data	Input	Pointer to data structure

Return Values

0:	Command has been executed normally
-1:	Command has not been executed
-2:	Error occurs

Properties

Prototyped in LCD.h

Description

Used as a callback function of the display driver. Called from display driver and executes callback routine.

In the emWin FIT module, this function initializes peripheral functions selected for the interface according to a command and performs processing according to the interface.

If GLCDC is selected for the interface, this function initializes the GLCDC FIT module, registers a figure generation function using the DRW2D FIT module, sets the Lookup Table entry, turns on and off the display, and switches the buffer.

If the RSPI or SCI (simple SPI mode) is selected for the interface, this function initializes the RSPI FIT module or SCI FIT module and turns on and off the display.

Command	Value	Meaning	Supporting status Y: Supported N: Not supported
LCD_X_INITCONTROLLER	0x01	Initializes display controller	Y
LCD_X_SETVRAMADDR	0x02	Sets Video RAM address	N
LCD_X_SETORG	0x03	Sets standard within layer	N
LCD_X_SETLUTENTRY	0x04	Sets Lookup Table entry	Y
LCD_X_ON	0x05	Switches on display	Y
LCD_X_OFF	0x06	Switches off display	Y
LCD_X_SETSIZE	0x07	Sets layer size	N
LCD_X_SETPOS	0x08	Sets layer position	N
LCD_X_SETVIS	0x09	Sets layer visualization	N
LCD_X_SETALPHA	0x0A	Sets layer alpha value	N
LCD_X_SETALPHAMODE	0x0B	Sets alpha blending mode	N
LCD_X_SETCHROMAMODE	0x0C	Sets chroma blending mode	N
LCD_X_SETCHROMA	0x0D	Sets chroma value	N
LCD_X_SHOWBUFFER	0x0E	Switches buffer	Y

Reentrant

- No

3.4 GUI_X_Init ()

This function is a function to initialize hardware necessary to GUI.

Format

void GUI_X_Init(void)

Parameters

None

Return Values

None

Properties

Prototyped in GUI.h

Description

A function to initialize necessary hardware

In the emWin FIT module, used to initialize compare match timer which is used for latency measurement

Reentrant

- No

3.5 GUI_X_Delay ()

This function is a function to wait for a specified time.

Format

```
void GUI_X_Delay(  
    int ms  
)
```

Parameters

ms Input Latency [a millisecond]

Return Values

None

Properties

Prototyped in GUI.h

Description

Waits for a specified time

In the emWin FIT module, waits for a specified time by utilizing time information obtained from compare match timer.

Reentrant

- No

3.6 GUI_X_ExecIdle ()

This function is a function called from Window Manager when there is no content to be processed because GUI is up to date.

Format

void GUI_X_ExecIdle(void)

Parameters

None

Return Values

None

Properties

Prototyped in GUI.h

Description

Called from Window Manager when GUI is up to date and there is no content to be processed.

In the emWin FIT module, performs no processing

Reentrant

- No

3.7 GUI_X_GetTime ()

This function is a function in which the current system time is obtained with integer type of millisecond unit.

Format

```
GUI_TIMER_TIME GUI_X_GetTime(  
    int ms  
)
```

Parameters

None

Return Values

System time [millisecond]

Properties

Prototyped in GUI.h

Description

The current system time is obtained with integer type of millisecond unit.

In the emWin FIT module, returns a value obtained from compare match timer

Reentrant

- No

3.8 GUI_X_ErrorOut ()

This function is a function called from emWin with an error string as an input when a fatal error occurs.

Format

```
void GUI_X_ErrorOut(  
    const char *s  
)
```

Parameters

s Input Error string

Return Values

None

Properties

Prototyped in GUI.h

Description

When a fatal error occurs, called from emWin with an error string as an input

Enabled when GUI_DEBUG_LEVEL ≥ 3

In emWin FIT module, performs no processing.

Reentrant

- No

3.9 GUI_X_Warn ()

This function is a function called from emWin with a warning string as an input when a warning occurs.

Format

```
void GUI_X_Warn(  
    const char *s  
)
```

Parameters

s Input Warning string

Return Values

None

Properties

Prototyped in GUI.h

Description

When a warning occurs, called from emWin with a warning string as an input

Enabled when GUI_DEBUG_LEVEL ≥ 4

In the emWin FIT module, performs no processing.

Reentrant

- No

3.10 GUI_X_Log ()

This function is a function called from emWin with a message string as an input when a message occurs.

Format

```
void GUI_X_Log(  
    const char *s  
)
```

Parameters

s Input Message string

Return Values

None

Properties

Prototyped in GUI.h

Description

When a message occurs, called from emWin with a message string as an input

Enabled when GUI_DEBUG_LEVEL ≥ 5

In the emWin FIT module, performs no processing.

Reentrant

- No

3.11 GUI_X_InitOS ()

This is a function to generate a semaphore or a mutex when used under multitask environment.

Format

void GUI_X_InitOS(void)

Parameters

None

Return Values

None

Properties

Prototyped in GUI.h

Description

A function to generate a semaphore or a mutex when used under multitask environment.

In the emWin FIT module, generates a semaphore and an event using FreeRTOS function when using FreeRTOS. When not using Free RTOS, performs no processing.

Reentrant

- No

3.12 GUI_X_Unlock ()

This function is a function to unlock GUI when used under multitask environment.

Format

void GUI_X_Unlock(void)

Parameters

None

Return Values

None

Properties

Prototyped in GUI.h

Description

A function to unlock GUI when used under multitask environment

In the emWin FIT module, releases a semaphore using FreeRTOS function when using FreeRTOS. When not using FreeRTOS, performs no processing.

Reentrant

- No

3.13 GUI_X_Lock ()

This function is a function to lock GUI when used under multitask environment.

Format

void GUI_X_Unlock(void)

Parameters

None

Return Values

None

Properties

Prototyped in GUI.h

Description

A function to lock GUI when used under multitask environment

In the emWIN FIT module, obtains a semaphore using FreeRTOS function when using FreeRTOS. When not using FreeRTOS, performs no processing.

Reentrant

- No

3.14 GUI_X_GetTaskId ()

A function to obtain a task ID when used under multitask environment

Format

U32 GUI_X_GetTaskId(void)

Parameters

None

Return Values

Task ID

Properties

Prototyped in GUI.h

Description

A function to obtain a task ID when used under multitask environment

In the emWin FIT module, obtains a task ID using FreeRTOS function when using FreeRTOS. When not using FreeRTOS, constantly returns 1.

Reentrant

- No

3.15 GUI_X_WaitEvent ()

A function to execute the waiting for an event when used under multitask environment

Format

void GUI_X_WaitEvent(void)

Parameters

None

Return Values

None

Properties

Prototyped in GUI.h

Description

A function to execute the waiting for an event when used under multitask environment

In the emWin FIT module, executes the waiting for an event using FreeRTOS function when using FreeRTOS. On this occasion, maximum waiting time is 60000 milliseconds. When not using FreeRTOS, performs no processing.

Reentrant

- No

3.16 GUI_X_SignalEvent ()

A function to execute event notification when used under multitask environment

Format

void GUI_X_SignalEvent(void)

Parameters

None

Return Values

None

Properties

Prototyped in GUI.h

Description

A function to execute event notification when used under multitask environment

In the emWin FIT module, executes event notification using FreeRTOS function when using FreeRTOS. When not using FreeRTOS, performs no processing.

Reentrant

- No

3.17 GUI_X_WaitEventTimed ()

This function is a function to execute the waiting for an event for a specified period when used under multitask environment.

Format

```
void GUI_X_WaitEventTimed(  
    int period  
)
```

Parameters

Period Input Specified period

Return Values

None

Properties

Prototyped in GUI.h

Description

A function to execute the waiting for an event for a specified period when used under multitask environment.

In the emWin FIT module, executes the waiting for an event for a specified period using FreeRTOS function when using FreeRTOS. In this occasion, maximum waiting time is 60000 milliseconds. When not using FreeRTOS, performs no processing.

Reentrant

- No

3.18 PID_X_SetLayerIndex ()

This function is a function to set a layer number.

Format

```
void PID_X_SetLayerIndex(  
    int layer_index  
)
```

Parameters

LayerIndex	Input	Layer number
------------	-------	--------------

Return Values

None

Properties

Prototyped in PIDConf.h

Description

Sets a layer number

In the emWin FIT module, sets a layer number to internal variable.

Reentrant

- No

3.19 PID_X_Init ()

This function is a function to initialize Pointer Input Device.

Format

void PID_X_Init(void)

Parameters

None

Return Values

None

Properties

Prototyped in PIDConf.h.

Description

Initializes Pointer Input Device

In the emWin FIT module, resets Touch IC, initializes SCI-I2C, boots compare match timer and registers callback function to obtain touch information, and enables multi-touch function.

Reentrant

- No

3.20 GUI_TOUCH_X_ActiveX ()

This function is a function to enable the voltage measurement of the X axis of Touch IC.

Format

void GUI_TOUCH_X_ActivateX(void)

Parameters

None

Return Values

None

Properties

Prototyped in GUI.h

Description

A function to enable voltage measurement of the X axis of Touch IC

In the emWin FIT module, performs no processing.

Reentrant

- No

3.21 GUI_TOUCH_X_ActiveY ()

This function is a function to enable the voltage measurement of the Y axis of Touch IC.

Format

void GUI_TOUCH_X_ActivateY(void)

Parameters

None

Return Values

None

Properties

Prototyped in GUI.h

Description

A function to enable the voltage measurement of the Y axis of Touch IC

In the emWin FIT module, performs no processing.

Reentrant

- No

3.22 GUI_TOUCH_X_MeasureX ()

This function is a function to return the X axis voltage measurement result obtained from Touch IC.

Format

int GUI_TOUCH_X_MeasureX(void)

Parameters

None

Return Values

0

Properties

Prototyped in GUI.h

Description

A function to return the X axis voltage measurement result obtained from Touch IC

In the emWin FIT module, constantly returns 0.

Reentrant

- No

3.23 GUI_TOUCH_Y_MeasureY ()

This function is a function to return the Y axis voltage measurement result obtained from Touch IC.

Format

int GUI_TOUCH_X_MeasureY(void)

Parameters

None

Return Values

0

Properties

Prototyped in GUI.h

Description

A function to return the Y axis voltage measurement result obtained from Touch IC

In the emWin FIT module, constantly returns 0.

Reentrant

- No

3.24 APPW_X_FS_Init ()

This function is a function to initialize the file system access of AppWizard.

Format

void APPW_X_FS_Init (void)

Parameters

None

Return Values

None

Properties

Prototyped in AppWizard.h

Description

A function to initialize the file system access of AppWizard

In the emWin Fit module, performs no processing.

Reentrant

- No

4. API Functions Called from the Application

4.1 R_EMWIN_GetBufferAddr()

This function is a function to obtain the address of the frame buffer which is used in the emWin FIT module.

Format

void * R_EMWIN_GetBufferAddr (void)

Parameters

None

Return Values

Frame buffer address

Properties

Prototyped in r_emwin_rx_if.h

Description

Obtains the address of the frame buffer used in the emWin FIT module.

Reentrant

- No

4.2 R_EMWIN_GetD2 ()

This function is a function to obtain the handle of the Dave2D function of the emWin FIT module.

Format

d2_device * R_EMWIN_GetD2 (void)

Parameters

None

Return Values

Handle of Dave2D

Properties

Prototyped in r_emwin_rx_if.h

Description

Obtains the handle of the Dave2D function of the emWin FIT module

This function is enabled only when the DRW2D FIT module is used.

Reentrant

- No

4.3 R_EMWIN_EnableDave2D ()

This function is a function to turn the Dave2D function of the emWin FIT module into the enable state.

Format

void R_EMWIN_EnableDave2D (void)

Parameters

None

Return Values

None

Properties

Prototyped in r_emwin_rx_if.h

Description

Turns the Dave2D function of the emWin FIT module into the enabled state.

This function is enabled only when the DRW2D FIT module is used.

Reentrant

- No

4.4 R_EMWIN_DisableDave2D ()

This function is a function to turn the Dave2D function of the emWin FIT module into the operation inhibition state.

Format

void R_EMWIN_DisableDave2D (void)

Parameters

None

Return Values

None

Properties

Prototyped in r_emwin_rx_if.h

Description

Turns the Dave2D function of the emWin FIT module into the operation inhibition state.

This function is enabled only when the DRW2D FIT module is used.

Reentrant

- No

4.5 R_EMWIN_GetDaveActive ()

This function is a function to obtain the operation state of the Dave2D function of the emWin FIT module.

Format

uint32_t R_EMWIN_GetDaveActive (void)

Parameters

None

Return Values

Dave2D operation state (0:state of forbidding operation, 1:State of enabling operation)

Properties

Prototyped in r_emwin_rx_if.h

Description

Obtains the operation state of the Dave2D function of the emWin FIT module

This function is enabled only when the DRW2D FIT module is used.

Reentrant

- No

4.6 R_EMWIN_GetVersion ()

This function is a function to obtain the version number of the emWin FIT module.

Format

```
void R_EMWIN_GetVersion(st_emwin_version_t * version)
```

Parameters

* version Output Pointer of the storage destination of a version number

Return Values

None

Properties

Prototyped in r_emwin_rx_if.h

Description

Obtains the version number of the emWin FIT module

Reentrant

- No

4.7 `_VSYNC_ISR ()`

This function is a function to perform V-sync interrupt processing.

Format

```
void _VSYNC_ISR(void * p)
```

Parameters

* p Output Callback argument from GLCDC

Return Values

None

Properties

Prototyped in `r_emwin_rx_if.h`

Description

Performs V-sync interrupt processing

Assuming the callback function of the GLCDC FIT module

Reentrant

- No

5. Pin Setting

The pin setting to use the emWin FIT module can be performed with QE for Display [RX].

The pins that require setting include the reset pin of the LCD panel, the backlight pin of the LCD panel, and the reset pin of the touch IC mounted on the LCD panel. Select the pins to be used according to the LCD connected to the RGB or SPI.

In case of e² studio, by using the pin setting function of the emWin setting dialog of the QE for Display [RX], pin setting can be performed. When using the QE for Display [RX], pin setting regarding r_emwin_rx with Smart Configurator is not required.

Information of the selected pin is applied to qe_emwin_config.h. Macro definition value shown in 2.6 Configuration while Compiling. When QE for Display [RX] is used, macro definitions in r_emwin_rx_config.h are disabled.

To perform the pin setting without using QE for Display[RX], edit r_emwin_rx_config.h included in the emWin FIT module.

6. Notation to implement the emWin FIT module

When the emWin FIT module is implemented, please note the following matters.

6.1 Selecting the library file

The emWin FIT module includes following library files. Please select the library correspond to the MCU and compiler. Note that when the Smart Configurator is used, the library is automatically configured according to the device and compiler to be used.

Library files	
emWinLib_RXv1_CCRX.lib	A library file to be used with Renesas Electronics C/C++ Compiler for RX Family.
emWinLib_RXv2_CCRX.lib	
emWinLib_RXv3_CCRX.lib	
libemWinLib_RXv1_GCC.a	A library file to be used with GCC for Renesas RX.
libemWinLib_RXv2_GCC.a	
libemWinLib_RXv3_GCC.a	
emWinLib_RXv1_IAR.a	A library file to be used with IAR C/C++ Compiler for Renesas
emWinLib_RXv2_IAR.a	
emWinLib_RXv3_IAR.a	

6.2 Setting the section when using RX65N

When using GLCDC as the interface, the emWin FIT module requires two frame buffers to be allocated. When using RX65N, the two frame buffers must be placed separately due to the relation of address placement. Therefore, if 256 Kbytes from 0x00000100 and 256 Kbytes from 0x00800000 are allocated respectively for frame buffers, set the location of the SU section (or later section) that was originally set to 0x00840000 or later.

6.3 Setting the RAM size required for system operation

The emWin FIT module has a relatively high ratio of RAM in the system, and RAM requirements vary depending on the system to be developed. Therefore, some RX products might not operate correctly with the default RAM size.

If necessary, adjust the following sizes by using the Smart Configurator.

Items set for r_bsp

- User stack (BSP_CFG_USTACK_BYTES)
- Interrupt stack (BSP_CFG_ISTACK_BYTES)
- Heap memory size (BSP_CFG_HEAP_BYTES)

Item set in the emWin FIT module

- Maximum memory size used for the GUI (EMWIN_GUI_NUM_BYTES)

6.4 Setting the heap memory size when DRW2D is enabled

When DRW2D is enabled, change the value of "Heap size" of "r_bsp" to 0x4000 by using the Smart Configurator. Note that 0x4000 is just a reference and the required heap memory size depends on the system to be developed.

6.5 Setting the compile option

You must set the compile option (-nostuff=C), which allocates const qualified variables to sections having 4-byte boundary alignment.

When using e² studio, open the property screen by selecting [Project] > [C/C++ Project Settings], open the [Tool Settings] tab by selecting [C/C++build] > [Settings], select [Compiler] > [Object], and then select the check box of [Allocates const qualified variables to 4-byte boundary alignment sections] (-nostuff=C). Note that when the Smart Configurator is used, this option is automatically set.

6.6 Image format

When using images with the emWin FIT module, use the data in bit map format (.bmp).

6.7 Setting multi-buffering when using AppWizard

To ensure that the widget configured in AppWizard operates smoothly without flicker, the multi-buffering option must be enabled.

For convenience of the use of AppWizard in the Renesas environment, the [Selected BSP] option in the project properties of AppWizard must be set to [None]. In this case, enable the multi-buffering option [Enable Multibuffering] that is disabled by default.

Note that the operation for enabling multi-buffering is not required in QE for Display [RX] version 3.01.00 or later. This is because an AppWizard project file with multi-buffering enabled is created when AppWizard is started from QE for Display[RX].

6.8 When using an SPI (serial interface) LCD

Using an SPI (serial interface) LCD with default settings might cause the LCD display to flicker. Use the following methods to ensure smooth operation without flicker:

- Enable the cache (a buffer for one frame is required).
- Use the memory device function# (setting in the user application program is required).

#: The memory device function provides a temporary buffer (mounted on emWin) used for drawing operation and performs various operations. For details, refer to the emWin user guide.

6.9 Color depth setting when using an SPI (serial interface) LCD

The emWin FIT module supports the following LCD controllers. The maximum color depth of these controllers is 18 bits, so if EMWIN_BITS_PER_PIXEL is set to "24" (24 bits of RGB888), the colors will be reduced to 18 bits.

- ST7715 Series
- ILI9341 Series

6.10 When using the SPI (serial interface) touch function

In the SCI FIT module or RSPI FIT module settings, specify 0x00 for the dummy data to be sent during read operation.

For the SCI FIT module: SCI_CFG_DUMMY_TX_BYTE

For the RSPI FIT module: RSPI_CFG_DUMMY_TXDATA

6.11 Implementing to the environment exclusive of operation confirmed

When the emWin FIT module is implemented to the environment exclusive of operation confirmed, please note following matters. The environment which operation is confirmed are described in 8.1.

- Use of LCD

In the emWin FIT module, the interface with the LCD can be configured by using the following methods:

1. QE for Display[RX]

When using e² studio, add emWin FIT module with Smart Configurator. Then, input necessary settings to QE for Display[RX]. Please refer to the below URL to know details.

<https://www.renesas.com/jp/en/software-tool/qe-display-development-assistance-tool-display-applications>

2. Smart Configurator

When using e² studio, input necessary settings to Smart Configurator.

Note: When QE for Display[RX] is used, the settings are overwritten.

3. Implement setting data structure (when GLCDC is used for the interface with the LCD)

The setting data of GLCDC can be implemented without Smart Configurator and QE for Display. The source code to be implemented is in LCDConf_glcdc_if.c. In r_emwin_lcd_open function, the setting data structure is set and substituted for R_GLCDC_Open function.

4. Edit a macro definition and source code (when the RSPI or SCI (simple SPI mode) is selected for the interface with the LCD)

The LCDConf_spi_if.c file contains the macro definition and source code appropriate for the LCD connected to the SPI. Operation of LCDs has not been confirmed except for those described in 8.1. If your LCD does not operate normally with the existing settings or source codes, modifications are required according to your LCD. If LCD_DRV_IC_OTHER is selected for EMWIN_LCD_DRIVER_IC in r_emwin_rx_config.h, the macro definition or source code appropriate for your LCD must be implemented. When using e² studio for implementation, the places in which the macro definition or source code must be implemented are marked with warnings.

When the GLCDC FIT module, RSPI FIT module, or SCI FIT module is not used, implement the process in LCDConf_user_if.c. When using e² studio for implementation, the places in which the interface must be implemented can be marked with warnings by specifying LCD_IF_OTHER for EMWIN_LCD_IF in r_emwin_rx_config.h.

- Use of touch panel

In the emWin FIT module, the process to use the touch panel with the I²C and SPI interfaces is implemented. When using other touch panels or other interfaces, the necessary processes must be implemented as follows.

1. Process to get touch data and pass to emWin library in PIDConf.c. (pidconf_cb_single function)

2. Process to interface with the touch panel in r_emwin_rx_pid_user_if.c

When the process in 2 is implemented by using e² studio, the code within the file is enabled by specifying TOUCH_IF_OTHER for EMWIN_TOUCH_IF in r_emwin_rx_config.h.

- Use of OS

emWin FIT module supports FreeRTOS (BSP_CFG_RTOS_USED == 1) and no OS (BSP_CFG_RTOS_USED == 0). When using other OS, alternative process must be implemented in GUI_X_Ex.c.

When using e² studio, the places to be modified can be shown with BSP_CFG_RTOS_USED is other values. Then, warnings are activated in these places. Please refer to the below document for details. Board Support Package Module Using Firmware Integration Technology (R01AN1685)

- Setting of emWin library

In emWin FIT module, supported number of frame buffers is 3 or below, supported display driver is GUIDRV_Lin. When increase frame buffers to reduce flickering, or applying other display drivers, following places must be implemented.

1. To increase number of frame buffers more than 4, set EMWIN_NUM_BUFFERS to the value which acceptable value is 16 or below. Then, necessary implementation must be applied in LCDConf_glcdc_if.c. When using e² studio, the places to be modified can be shown with setting EMWIN_NUM_BUFFERS in r_emwin_rx_config.h to the value. Then, warnings are activated in these places.
2. To use a display driver other than GUIDRV_Lin or GUIDRV_FlexColor, implement the process within LCDConf_user_if.c. When using e² studio for implementation, the code within the file is enabled by specifying GUIDRV_OTHER for EMWIN_DISPLAY_DRIVER in r_emwin_rx_config.h.

7. Sample Application

Sample application is stored in doc/Training folder. For the detail, refer to the document below.

- emWin Training

8. Appendix

8.1 Confirmed Operation Environment

This section describes confirmed operation environment for the emWin FIT module.

Table 8.1 Confirmed Operation Environment

Item	Contents
Integrated Development Environment	Renesas Electronics e ² studio 2022-07
C compiler	Renesas Electronics C/C++ Compiler for RX Family(CC-RX) V3.04.00 Compile option: Add the option below to the default setting of the Integrated Development Environment. -lang = c99
	GCC for Renesas RX 8.03.00.202202 Compile option: Add the option below to the default setting of the Integrated Development Environment -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.20.1 Compile option: The default setting of the Integrated Development Environment
Endian	Little endian
Version of the Module	Ver.1.00
OS	FreeRTOS Release Release RX MCUs FreeRTOS v1.0.5 comes from original 10.4.3 - renesas/FreeRTOS-Kernel - GitHub
	Without OS
Board used	Renesas Envision KIT RPBRX65N (Product No.: RTK5RX65N2C00000BR) Renesas Envision Kit RPBRX72N (Product No.: RTK5RX72N0C00000BJ) Renesas Starter Kit+ for RX65N-2MB (RTK50565N2S10010BE) Renesas Starter Kit+ for RX72N (Product No.: RTK5572NNHS10000BE) Renesas Starter Kit for RX140 (Product No.: RTK551406BS00000BE) Target Board for RX130 (Product No.: RTK5RX1300C00000BR) Target Board for RX671 (Product No.: RTK5RX6710C00000BJ)

Table 8.2 Operation Confirmed LCDs

LCD product No.	Resolution	LCD controller	Touch controller	Remarks
RGB interface				
NHD-4.3-480272EF-ATXL#-CTP (Newheaven Display)	480x272	HX8257-A (Himax)	FT5306 (FocalTech)	Mounted on Renesas Starter Kit+ for RX65N-2MB Mounted on Renesas Starter Kit+ for RX72N
ER-TFT043-3 (East Rising)	480x272	NV3047 (TDK)	FT5206 (FocalTech)	Mounted on Renesas Envision KIT RPBRX65N Renesas Envision Kit RPBRX72N
Serial interface				
RH128128T-1x44WN-B2 (OKAYA)	128x128	ST7715R (Sitronix)	—	Pmod connection LCD supplied with Renesas Starter Kit
MSP2807 (Kuongshun Electronic Limited)	240x320	ILI9341 (ILITEK)	XPT2046 (Xptek)	Used for operation confirmation

8.2 Troubleshooting

- (1) Q: I have added the FIT module to the project and built it. Then I got the error: Could not open source file "platform.h".

A: The FIT module may not be added to the project properly. Check if the method for adding FIT modules is correct with the following documents:

- When using e² studio
Application note "Adding Firmware Integration Technology Modules to Projects (R01AN1723)"

When using this FIT module, the board support package FIT module (BSP module) must also be added to the project. Refer to the application note "Board Support Package Module Using Firmware Integration Technology (R01AN1685)"

9. Reference Documents

User's manual: Software

- emWin Graphic Library with Graphical User Interface User Guide & Reference Manual

(<https://www.segger.com/downloads/emwin/UM03001>)

User's manual: Hardware

(The latest version of each device can be downloaded from the Renesas Electronics website.)

Technical Update/Technical News

(The latest information can be downloaded from the Renesas Electronics website.)

User's Manual: Development Environment

RX Family C/C++ Compiler CC-RX User's Manual (R20UT3248)

(The latest version can be downloaded from the Renesas Electronics website.)

Related Technical Update

This module has no technical update.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Sep.7.22	—	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. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity.

Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. 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 the 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.

5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.

6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of 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 claims 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, drawings, charts, programs, algorithms, and 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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer 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, copying or reverse engineering.
6. 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; 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.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; 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 any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using 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 dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. 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. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a 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. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using 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.
11. 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 comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.