RE01 1500KB, 256KB Group

SMIP Driver Detailed Specification

Summary

This document describes the detailed specifications of the SMIP driver provided in the RE01 1500KB and 256KB Group CMSIS software package (hereinafter referred to as the SMIP driver).

Target Device

RE01 1500KB Group

RE01 256KB Group

Contents

1. Overview

The SMIP driver allows the RE01 1500KB and 256KB group products to control the MIP-LCD by using the SPI. The operation of the SMIP driver has been confirmed with the following LCDs.

Product Number	Manufacturer	Resolution	Display Color
LPM013M126A	Japan Display Inc. (JDI)	176(H) x 176(V)	8 colors
TN0181ANVNANN-*N*03	KYOCERA Corporation	256(H) x 256(V)	Monochrome (black-white binary)
LS013B7DH03	Sharp Corporation	128(H) x 128(V)	Monochrome (black-white binary)

The LCD products indicated by the product numbers will be hereafter referred to by the corresponding manufacturer's names. The default values for setting each LCD defined in the configuration correspond to the LCDs listed above.

This driver uses the peripheral functions shown in Figure 1-1.

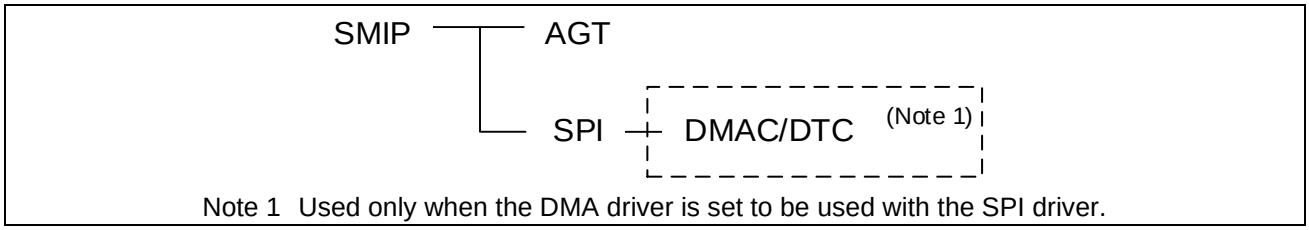


Figure 1-1 Block Diagram of SMIP Driver Peripheral Functions

The following table shows the functions used with this driver.

Peripheral Function	Description	Operation Mode
AGT	VCOM (EXTCOMIN) output	Timer mode/pulse output mode
SPI (Note1)	Data output	Master transmit mode, MSB/LSB first(Note2), 8-bit data length
DMAC/DTC	SPI data write (only when the DMA driver is set to be used with the SPI driver.)	Normal transfer

Note1 Setting the SPI driver is required. For details of the SPI driver, see the RE01 1500KB, 256KB Group CMSIS Driver R_SPI Specifications (R01AN4728).

Note2 MSB/LSB is set in "SMIP_CFG_SPI_BIT_ORDER" of r_smip_cfg.h.

1.1 Setting SPI Driver Used with SMIP

1.1.1 Setting DMA to Use

Whether to use the DMA driver with the SPI driver is set by the SPIn_TRANSMIT_CONTROL (n = 0,1) definition in r_spi_cfg.h. Figure 1-2 shows an example to use the DTC driver.

```
#define SPI0_TRANSMIT_CONTROL   SPI_USED_DTC     ///< SPI0 transmit control
```

Figure 1-2 Example of Setting DTC Driver to Use with r_spi_cfg.h (SPI0 used)

1.1.2 Setting Interrupts (NVIC)

To operate the SPI in master transmit mode with the SMIP driver, the interrupts to be used must be registered using r_system_cfg.h. For details of interrupts (NVIC), refer to "Setting Interrupts (NVIC)" in the RE01 1500KB, 256KB Group Getting Started Guide to Development Using CMSIS Package (r01an4660). Table 1-1 shows the interrupt definition to be used with the SPI driver.

Table 1-1 NVIC Registration Definitions (n = 0 or 1, m = 0 to 3)

Mode	Used for	NVIC Registration Definition
Master	Used only for transmission	[When using interrupts and DTC for transmission control] SYSTEM_CFG_EVENT_NUMBER_SPIn_SPTI [When using DMAC for transmission control] SYSTEM_CFG_EVENT_NUMBER_DMACm_INT
		SYSTEM_CFG_EVENT_NUMBER_SPIn_SPII

```
#define SYSTEM_CFG_EVENT_NUMBER_SCI0_TXI
          (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)    /*!< Numbers 1/5/9/13/17/21/25/29 only */
#define SYSTEM_CFG_EVENT_NUMBER_SPI0_SPTI
          (SYSTEM_IRQ_EVENT_NUMBER1)              /*!< Numbers 1/5/9/13/17/21/25/29 only */
#define SYSTEM_CFG_EVENT_NUMBER_SOL_DL
          (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)    /*!< Numbers 1/9/17/25 only */
...
#define SYSTEM_CFG_EVENT_NUMBER_SCI0_TEI
          (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)    /*!< Numbers 2/6/10/14/18/22/26/30 only */
#define SYSTEM_CFG_EVENT_NUMBER_SPI0_SPII
          (SYSTEM_IRQ_EVENT_NUMBER2)              /*!< Numbers 2/6/10/14/18/22/26/30 only */
#define SYSTEM_CFG_EVENT_NUMBER_SPI0_SPTEND
          (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)    /*!< Numbers 2/6/10/14/18/22/26/30 only */
...
```

Figure 1-3 Example of Registering Interrupts to NVIC with r_system_cfg.h (SPI0 used)

1.1.3 Setting SPI Pins

The pins to be used by the SPI driver are set with the R_RSPI_Pinset_CHn (n = 0, 1) function and released with the R_RSPI_Pinclr_CHn function in pin.c.

Select the pin to be used by editing the R_RSPI_Pinset_CHn and R_RSPI_Pinclr_CHn functions of pin.c. The names of the pins used with the SPI functions each have a suffix _A, _B, _C, or _D. When assigning an SPI function, select function pins having the same suffix (Notes 1 and 2).

Figure 1-4 shows an example for setting of the SPI0.

Note 1. Signals with the same suffix belong to a group in which the signals are adjusted in timing. Signals having different suffixes cannot be used at the same time. The exceptions are the RSPCKA_C and MOSIA_C signals for the SPI and the SSLB0_D signal, which can be used at the same time as signals from group B.

Note 2. The RST and SCS signals used by this driver are controlled with port outputs. Do not assign the RST and SCS control pins specified with r_smip_cfg.h to peripheral functions including the SPI. Similarly, when AGTOn is not used for the VCOM signal (when SMIP_CFG_AGTO_EN is set to 1), do not assign the VCOM control pin to a peripheral function.

```
/*****************************************************************************//**
* @brief This function sets Pin of RSPI0.
* @note  Several pin names have added _A, _B, and _C suffixes.@n
*       When assigning the SPI functions, select the functional pins with the same suffix.@n
*       Comment out the terminal of unused suffix.@n
*       When using “RSPCKA_C, MOSIA_C” added by the SPI, select the pair of RSPCKA_B and RSPCKA_C
*       and the pair of MOSIA_B and MOSIA_C. When using “SSLB0_D” added by SPI,
*       select the pair of SSLB0_D and SSLB0_B.
*****************************************************************************/
/* Function Name : R_RSPI_Pinset_CH0 */
void R_RSPI_Pinset_CH0(void)  // @suppress("API function naming") @suppress("Function length")
{
    /* Disable protection for PFS function (Set to PWPR register) */
    R_SYS_RegisterProtectDisable(SYSTEM_REG_PROTECT_MPC);
//    /* MISOA_A : P105 */
//    PFS->P105PFS_b.PMR  = 0U;
//    PFS->P105PFS_b.ASEL = 0U;
//    PFS->P105PFS_b.ISEL = 0U;
//    PFS->P105PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
//    PFS->P105PFS_b.PMR  = 1U;

    /* Set P500 as MISOA pin. */
    /* MISOA_B : P500 */
    PFS->P500PFS_b.ASEL = 0U;
    PFS->P500PFS_b.ISEL = 0U;
    PFS->P500PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
    PFS->P500PFS_b.PMR  = 1U;

/* When using P103 for the SCS (RST) signal, do not assign P103 to the SSLA0A pin */
//    /* SSLA0_A : P103 */
//    PFS->P103PFS_b.ASEL = 0U;  /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
//    PFS->P103PFS_b.ISEL = 0U;  /* 0: Do not use as an IRQn input pin,  1: Use as an IRQn input pin. */
//    PFS->P103PFS_b.PSEL = R_PIN_PRV_RSPI_PSEL;
//    PFS->P103PFS_b.PMR  = 1U;  /* 0: Use the pin as a general I/O port,
                                                                  1: Use the pin as a peripheral module. */
...
}/* End of function R_RSPI_Pinset_CH0() */

/*****************************************************************************//**
* @brief This function clears the pin setting of RSPI0.
*****************************************************************************/
/* Function Name : R_RSPI_Pinclr_CH0 */
void R_RSPI_Pinclr_CH0(void)  // @suppress("API function naming")
{
    /* Disable protection for PFS function (Set to PWPR register) */
    R_SYS_RegisterProtectDisable(SYSTEM_REG_PROTECT_MPC);

//    /* MISOA_A : P105 */
//    PFS->P105PFS &= R_PIN_PRV_CLR_MASK;

    /* Release MISOA pin */
    /* MISOA_B : P500 */
    PFS->P500PFS &= R_PIN_PRV_CLR_MASK;

/* When using P103 for the SCS (RST) signal, do not assign P103 to the SSLA0A pin */
//    /* SSLA0_A : P103 */
//    PFS->P103PFS &= R_PIN_PRV_CLR_MASK;
...
}
```

Figure 1-4 Example of Setting Pins

1.2 Setting AGT Used with SMIP

1.2.1 Registering AGTI Interrupts to NVIC

It is required to set the AGT timer interrupts (AGTn_AGTI) if VCOM (EXTCOMIN) output is generated using the AGT timer interrupts. For details of interrupts (NVIC), refer to "Setting Interrupts (NVIC)" in the RE01 1500KB, 256KB Group Getting Started Guide to Development Using CMSIS Package (r01an4660).

Table 1-2 AGTI Interrupt Setting Combinations shows for which condition combinations the AGT timer interrupt setting is required.

Table 1-2 AGTI Interrupt Setting Combinations

SMIP_CFG_EXTMODE (Note 1)	Transmit Processing	AGTI Interrupt Setting
0	KYOCERA (Note 2)	Required
0	JDI and SHARP (Note 3)	Not required
1	KYOCERA (Note 2)	Required
1	JDI and SHARP (Note 3)	Required

Note 1. EXTMODE pin setting definition to set EXTMODE pin input level

Note 2. When b4 of the st_smip_cfg_t structure *type* member is 1, transmission control synchronizes with VCOM output.

Note 3. When b4 of the st_smip_cfg_t structure *type* member is 0, transmission control does not synchronize with VCOM output.

When it is required to set the AGT timer interrupts, register the interrupts to NVIC in _system_cfg.h. Figure 1-5 shows an example of registering the AGT timer interrupts to the NVIC.

```
...
#define SYSTEM_CFG_EVENT_NUMBER_QSPI_INTR
    (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers 3/7/11/15/19/23/27/31 only */
#define SYSTEM_CFG_EVENT_NUMBER_AGT0_AGTI
    (SYSTEM_IRQ_EVENT_NUMBER11)         /*!< Numbers 3/11/19/27 only */
#define SYSTEM_CFG_EVENT_NUMBER_USBFS_USBR
    (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)  /*!< Numbers 3/11/19/27 only */
...
```

Figure 1-5 Example of Registering Interrupts to NVIC with r_sysytem_cfg.h (AGT0 used)

1.2.2 Setting AGTO Pins

When AGTO output controlled by VCOM (EXTCOMIN) is enabled (Note), the AGT pin setting function and pin release function are executed. The pins to be used by the AGT are set with the R_AGT_Pinset_CHn (n = 0, 1) function and released with the R_AGT_Pinclr_CHn function.

Select the pin to be used by editing the R_AGT_Pinset_CHn and R_AGT_Pinclr_CHn functions of pin.c (Note 2). Figure 1-6 shows an example for setting of the AGTO0 pin.

Note. When SMIP_CFG_AGTO_EN is set to 1, AGTO output controlled by VCOM (EXTCOMIN) is enabled. For details, see section 4.4.5, EXTMODE Pin Setting Definition.

```
/*****************************************************************************//**
* @brief This function sets Pin of AGT0.
*****************************************************************************/
/* Function Name : R_AGT_Pinset_CH0 */
void R_AGT_Pinset_CH0(void)  // @suppress("API function naming")
{
...
    /* AGTO0 : P808 */
//    PFS->P808PFS_b.ASEL = 0U;  /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
//    PFS->P808PFS_b.ISEL = 0U;  /* 0: Do not use as an IRQn input pin,  1: Use as an IRQn input pin. */
//    PFS->P808PFS_b.PSEL = R_PIN_PRV_AGT_PSEL;
//    PFS->P808PFS_b.PMR  = 1U;
                  /* 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* Set P111 as AGTO0 */
    /* AGTO0 : P111 */
    PFS->P111PFS_b.ASEL = 0U;  /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    PFS->P111PFS_b.ISEL = 0U;  /* 0: Do not use as an IRQn input pin,  1: Use as an IRQn input pin. */
    PFS->P111PFS_b.PSEL = R_PIN_PRV_AGT_PSEL;
    PFS->P111PFS_b.PMR  = 1U;
                  /* 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* AGTOA0 : P807 */
//    PFS->P807PFS_b.ASEL = 0U;  /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
//    PFS->P807PFS_b.ISEL = 0U;  /* 0: Do not use as an IRQn input pin,  1: Use as an IRQn input pin. */
//    PFS->P807PFS_b.PSEL = R_PIN_PRV_AGT_PSEL;
//    PFS->P807PFS_b.PMR  = 1U;
                  /* 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */
...
}/* End of function R_AGT_Pinset_CH0() */

/*****************************************************************************//**
* @brief This function clears the pin setting of AGT0.
*****************************************************************************/
/* Function Name : R_AGT_Pinclr_CH0 */
void R_AGT_Pinclr_CH0(void)  // @suppress("API function naming")
{
  ...
    /* AGTO0 : P808 */
//    PFS->P808PFS &= R_PIN_PRV_CLR_MASK;

    /* Release AGTO0 pin */
    /* AGTO0 : P111 */
    PFS->P111PFS &= R_PIN_PRV_CLR_MASK;

    /* AGTOA0 : P807 */
//    PFS->P807PFS &= R_PIN_PRV_CLR_MASK;
...
}/* End of function R_AGT_Pinclr_CH0() */
```

Figure 1-6 Example of Setting Pins

2. Driver Configuration

2.1 File Configuration

This SMIP driver conforms to the CMSIS HAL driver package and consists of three files: "r_smip_api.c", "r_smip_api.h", and "r_smip_cfg.h" in the vendor-specific file storage directory. The functions of the files are shown in Table 2-1, and the file configuration is shown in Figure 2-1.

Table 2-1 Functions of SMIP Driver Files

File Name	Description
r_smip_api.c	Driver source file This file provides the detail of the driver function. To use the SMIP driver, it is necessary to build this file.
r_smip_api.h	Driver header file The macro, type, and prototype definitions to be used in the driver are defined. To use the SMIP driver, it is necessary to include this file.
r_smip_cfg.h	Configuration definition file This provides configuration definitions that can be modified by the user.

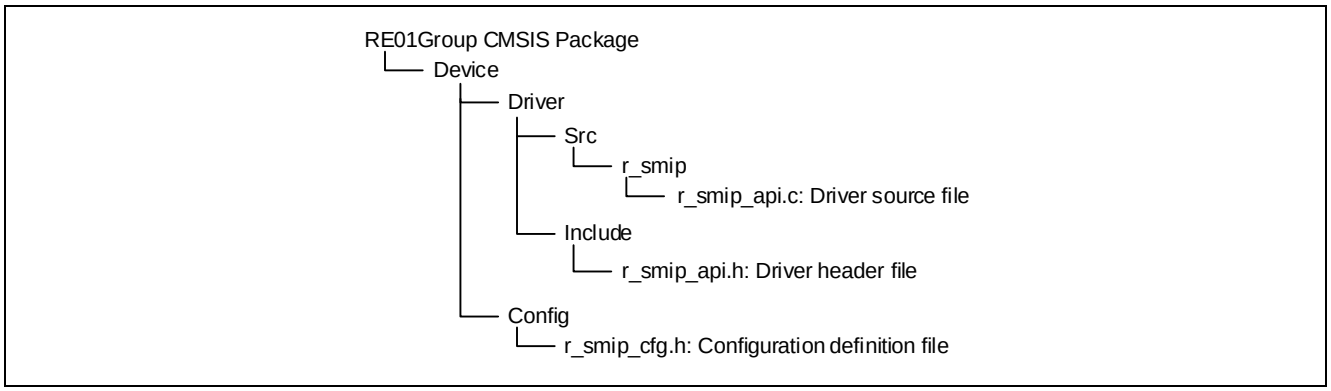


Figure 2-1 File Configuration of SMIP Driver

2.2 Driver APIs

The list of SMIP driver APIs is shown in Table 2-2.

Table 2-2 SMIP Driver APIs

API	Description
R_SMIP_GetVersion	Acquires the version of the SMIP driver.
R_SMIP_Open	Opens the SMIP driver (initializes RAM and SPI driver). If the AGTI interrupt is used (Note 1), it will also set the interrupt.
R_SMIP_Close	Releases the SMIP driver (releases the SPI driver). If the AGTI interrupt is used (Note 1), it will also disable the interrupt.
R_SMIP_PowerOn	Executes MIP-LCD power-on sequence.
R_SMIP_PowerOff	Executes MIP-LCD power-off sequence.
R_SMIP_AllZero	Transmits 0 to full screen.
R_SMIP_AllOne	Transmits 1 to full screen.
R_SMIP_Send	Transmits monochrome data.
R_SMIP_SendColor	Transmits color data.
R_SMIP_SendCommand (Note 2)	Transmits a command.
R_SMIP_Suspend	Clears the SPI communication settings and releases the pins used for SPI
R_SMIP_Resume	Reconfigure SPI communication settings and assign pins to SPI
R_SMIP_ReconfigSpiSpeed	Change SPI communication speed

Note 1. Refers to the case in which SMIP_CFG_EXTMODE is 1 or transmission control synchronizes with VCOM output.
For detail, see section 1.2.1, Registering AGTI Interrupts to NVIC.

Note 2. Cannot be used if KYOCERA's LCD has been specified by the Open function.

2.3 Setting SMIP

For this driver, specify the LCD type to use by the Open function. When using the JDI's, KYOCERA's or SHARP's LCD, specify the g_smip_tbl_lcd_info[XXX] pointer as the third argument. For XXX, specify the LCD type definition. For details of the SMIP LCD type definition, see 2.4.1, SMIP LCD Type Definitions.

Specify the SPI channel to use and SPI transmit clock frequency (Hz) as the first and second arguments of the Open function. When using a callback function, specify it as the fourth argument.

Example: Uses JDI' s LCD (SPI0, 1 MHz, no callback function)

```
R_SMIP_Open(0, 1000000, &g_smip_tbl_lcd_info[SMIP_TYPE_JDI] ,NULL);
```

2.4 Macro and Type Definitions

For the SMIP driver, the macro and types that can be referenced by the user are defined in the r_smip_api.h file.

2.4.1 SMIP LCD Type Definitions

The SMIP LCD type definitions are used for structure table index to be set as the third argument of the Open function.

Table 2-3 List of SMIP LCD Type Definitions

Definition	Setting	Description
SMIP_TYPE_KYOCERA	(0x00)	Uses an LCD manufactured by KYOCERA
SMIP_TYPE_JDI	(0x01)	Uses an LCD manufactured by JDI
SMIP_TYPE_SHARP	(0x02)	Uses an LCD manufactured by SHARP

2.4.2 SMIP Error Code Definitions

These define the SMIP error codes.

Table 2-4 List of SMIP Error Code Definitions

Definition	Setting	Description
SMIP_OK	0	SMIP driver processing completed
SMIP_ERROR	1	SMIP driver processing failed
SMIP_ERROR_BUSY	2	SMIP driver in busy state
SMIP_ERROR_PARAMETER	3	Incorrect parameter
SMIP_ERROR_MODE	4	Incorrect mode setting
SMIP_ERROR_LOCKED	5	AGT resource locked
SMIP_ERROR_TIMEOUT	6	Timeout error
SMIP_ERROR_SYSTEM_SETTING	7	Incorrect system setting
SMIP_ERROR_POWER_SEQUENCE	8	Error in power-on or power-off sequence

2.4.3 SMIP Event Definitions

These define the SMIP events.

Table 2-5 SMIP Event Definitions

Definition	Setting	Description
SMIP_EVENT_SEND_COMPLETE	(1UL << 0)	Data output completed
SMIP_EVENT_SEND_ERROR	(1UL << 1)	Data output failed

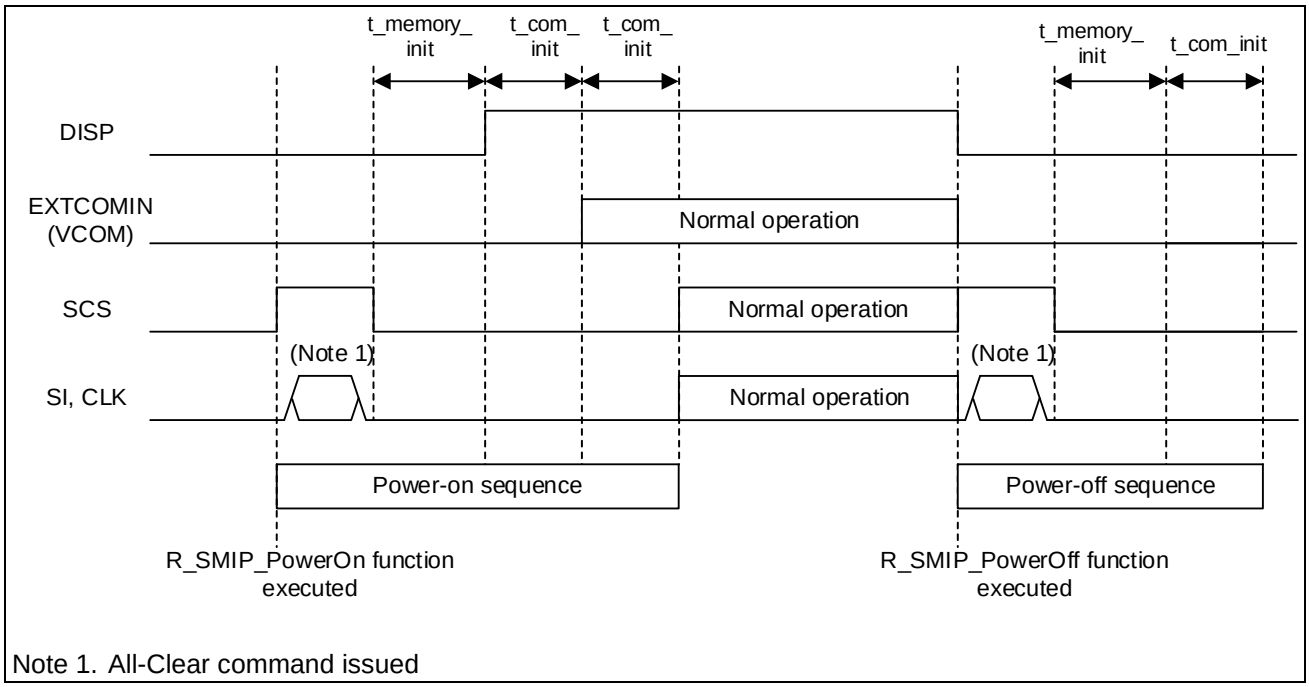
2.5 Structure Definitions

For the SMIP driver, the structures that can be referenced by the user are defined in the r_smip_api.h file.

2.5.1 st_smip_cfg_t Structure

This structure is used for specifying the LCD setting to use by the Open function. The settings for KYOCERA, JDI, and SHARP are defined in r_smip_api.c. To modify the setting, change the configuration in r_smip_cfg.h corresponding to the st_smip_cfg_t structure member. For configuration setting, see section 4.6 Configurations.

Figure 2-2 to Figure 2-4 show the timing settings of the power-on/power-off sequences defined by the st_smip_cfg_t structure, and Table 2-6 lists the st_smip_cfg_t structure members.



Note 1. All-Clear command issued

Figure 2-2 Timing Setting for Power-on/Power-off Sequence for JDI and SHARP

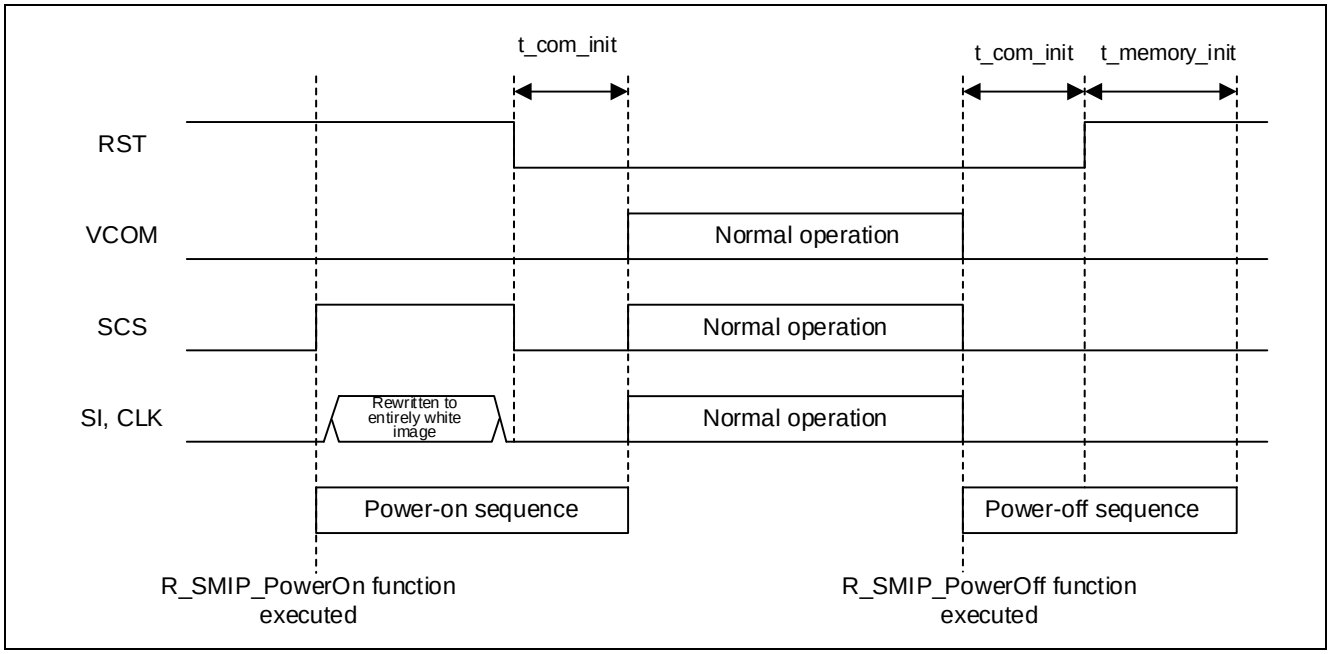
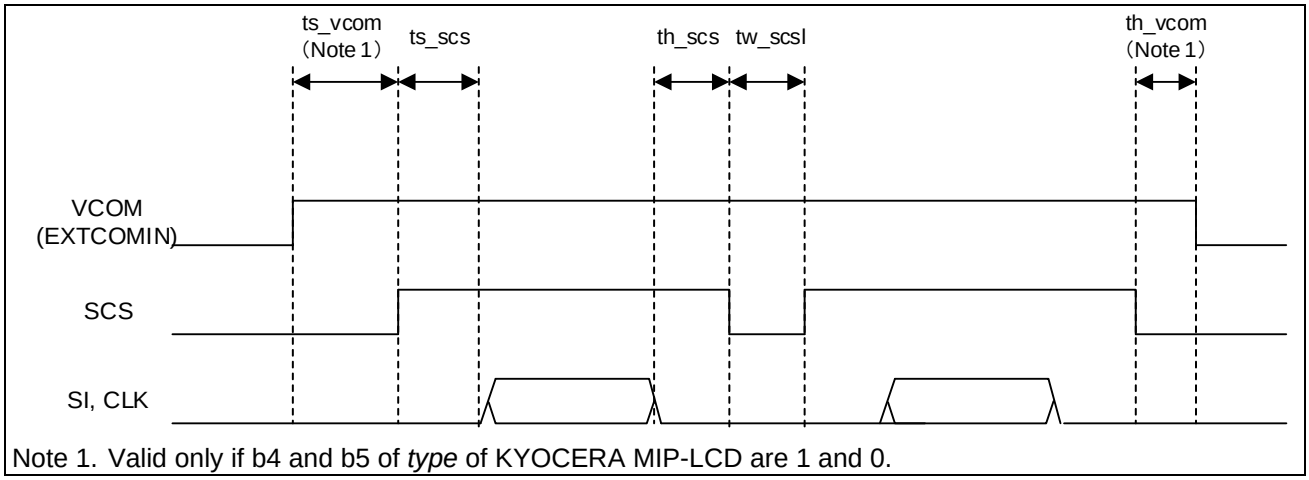


Figure 2-3 Timing Setting for Power-on/Power-off Sequence for KYOCERA



Note 1. Valid only if b4 and b5 of *type* of KYOCERA MIP-LCD are 1 and 0.

Figure 2-4 Setting Transmit Processing Time

Table 2-6 st_smip_cfg_t Structure

Element Name	Type	Description
type	uint8_t	b0: Color mode 0: 3-bit data mode 1: 4-bit data mode b1: Power ON-OFF sequence 0: JDI or SHARP type 1: KYOCERA type b2: AG bit endian 0: LSB 1: MSB b3: Dummy output between multiple lines. 0: No dummy output 1: Dummy output b4: Send timing 0: Transmission upon Send function execution 1: Transmission in synchronization with VCOM output
mode_size	uint8_t	Number of mode bits
mode_dmy_size	uint8_t	Number of mode dummy bits
address_size	uint8_t	Number of address bits
end_dummy_size	uint8_t	Number of dummy bits
first_addr	uint8_t	Initial line address
disp_width	uint16_t	Number of bits in MIP-LCD horizontal direction
disp_line	uint16_t	Number of lines of MIP-LCD
vcom_clk	uint16_t	VCOM frequency
t_memory_init	uint16_t	JDI, SHARP: Memory initialization wait time KYOCERA: RESET input time for termination
t_com_init	uint16_t	JDI, SHARP: VCOM wait time KYOCERA: VCOM wait time input time
ts_scs	uint8_t	SCS setup time
th_scs	uint8_t	SCS hold time
tw_scsl	uint8_t	SCS low width (Note 1)
ts_vcom	uint8_t	VCOM setup time (Note 2)
th_vcom	uint8_t	VCOM hold time (Note 2)
*cmd_tbl	uint8_t	Command table

Note 1. Invalid if SMIP_CFG_SCS_WAIT_EN is 0.
Note 2. Valid only if b4 of *type* are 1.

3. State Transitions

The state transition diagram of the SMIP driver is shown in Figure 3-1.

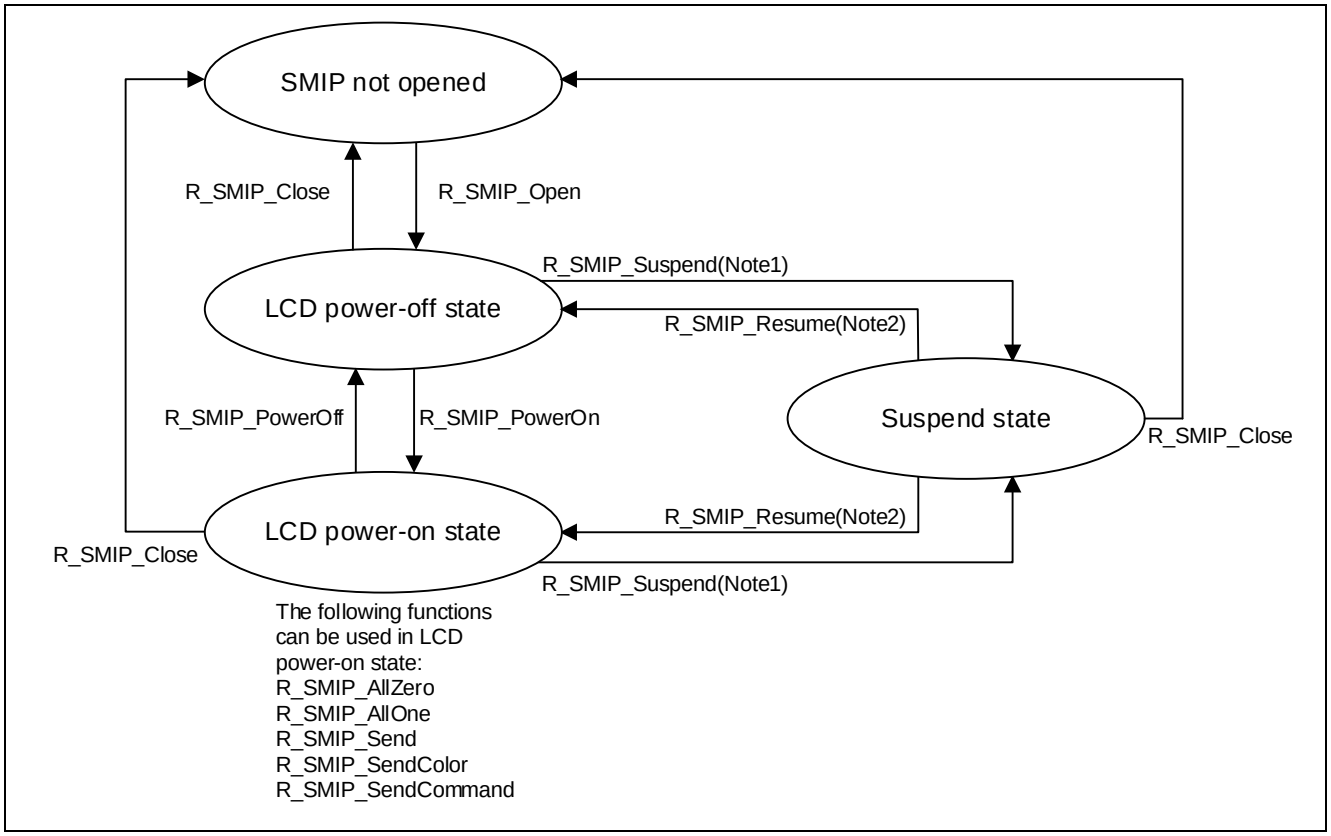


Figure 3-1 State Transitions (Note)

Note 1. R_SMIP_ERROR_BUSY is returned when executed during LCD display processing by SPI.

Note 2. The state transition when the R_SMIP_Resume function is executed in the suspended state is as follows:
[When transitioning from the LCD power-off state to the suspend state]
Returning to the LCD power-off state
[When transitioning from the LCD power-on state to the suspend state]
Returning to the LCD power-on state

Note 3. The R_SMIP_GetVersion function can be called from any state.

Note 4. The R_SMIP_ReconfigSpiSpeed function can be called after executing the Open function (can be executed in both LCD power-on and power-off states). However, R_SMIP_ERROR_BUSY will be returned if executed during LCD display processing by SPI.

4. Descriptions of Driver Operations

The SMIP driver outputs image data to the MIP-LCD by using the SPI driver. This chapter describes the driver operations with each LCD listed in chapter 4.6 Configurations, Overview.

The LCD settings are defined in r_smip_api.c as the st_smip_cfg_t structure table. For the st_smip_cfg_t structure, see section 2.5.1, st_smip_cfg_t Structure. The configurations corresponding to the st_smip_cfg_t structure members are defined for each LCD. For details of the configuration definitions, see section 4.6 Configurations.

4.1 Example of Using JDI's MIP-LCD

4.1.1 Example of Displaying Images on JDI's MIP-LCD

The JDI's LCD setting is defined in g_smip_tbl_lcd_info[SMIP_TYPE_JDI] of r_smip_api.c. For definition of JDI configuration, see section 4.6.14, JDI 's LCD Setting Definitions.

Figure 4-1 shows an example of setting the JDI's LCD.

```
#include "r_smip_api.h"

static void callback(uint32_t event);
extern const unsigned char image_jdi[11970];    /* JDI form image data */
main()
{
    (void)R_SMIP_Open(1, 1000000, &g_smip_tbl_lcd_info[SMIP_TYPE_JDI] ,callback);
                            /* SPI channel to use: 1; JDI's LCD; callback function */
    (void)R_SMIP_PowerOn();    /* Execute SMIP power-on sequence */
    while(1);
    {
        R_SYS_SoftwareDelay(1000, SYSTEM_DELAY_UNITS_MILLISECONDS);
        (void)R_SMIP_SendColor(0, 176, image_jdi);
                            /* Start address: 0; data line size: 176; image data table */
        R_SYS_SoftwareDelay(1000, SYSTEM_DELAY_UNITS_MILLISECONDS);
        (void)R_SMIP_SendCommand(0x20);    /* All-Clear command */
    }
}

/***********************************************************************************************
* callback function
***********************************************************************************************/
static void callback(uint32_t event)
{
    /* Describe the process when all transfers are completed */
    switch(event)
    {
        case SMIP_EVENT_SEND_COMPLETE:
        {
            /* Describe the process when transmission is normally completed */
        }
        break;
        case SMIP_EVENT_SEND_ERROR:
        {
            /* Describe the process when transmission error occurs */
        }
        break;
    }
}
```

Figure 4-1 Example of SMIP LCD Setting and Transmission (JDI)

4.1.2 Image Data for JDI's MIP-LCD

Figure 4-2 shows the address map for the JDI's LCD, Figure 4-3 shows an example of data arrangement when SMIP_CFG_MAKE_LINE_INFO is 0 (the SMIP driver does not add the header or footer during transmission (note 1)), and Figure 4-4 shows an example of data arrangement when SMIP_CFG_MAKE_LINE_INFO is 1 (the SMIP driver adds the header and footer during transmission (note 1)). (Note2)

Note 1. When SMIP_CFG_MAKE_LINE_INFO is set to 0, the automatic header and footer generation function is disabled; the image data including the header and footer must be provided.

Note 2. Here is an example of the arrangement when SMIP_CFG_SPI_BIT_ORDER is 0 (MSB first is selected for SPI bit order). When SMIP_CFG_SPI_BIT_ORDER is 1 (LSB first is selected for the SPI bit order), invert the bit string of the data to be arranged.

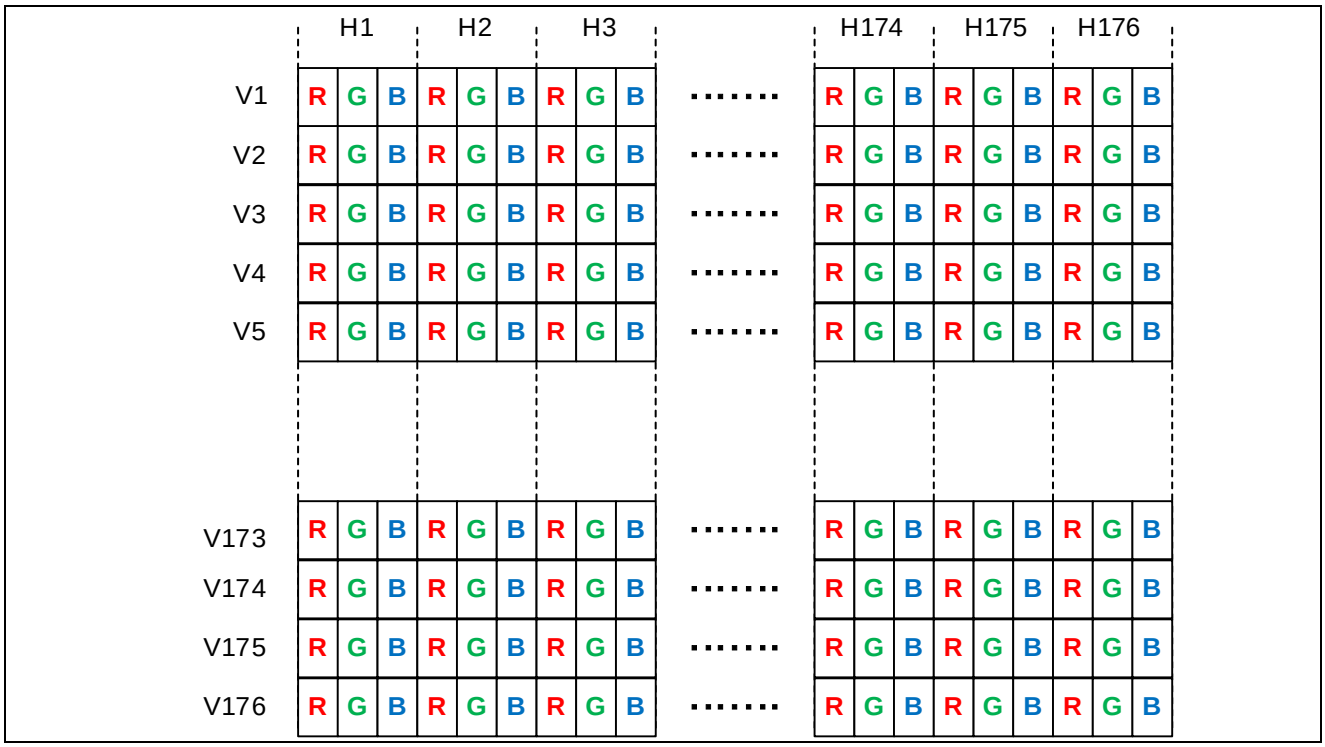


Figure 4-2 Address Map for JDI's MIP-LCD (3-bit color, 176 x 176)

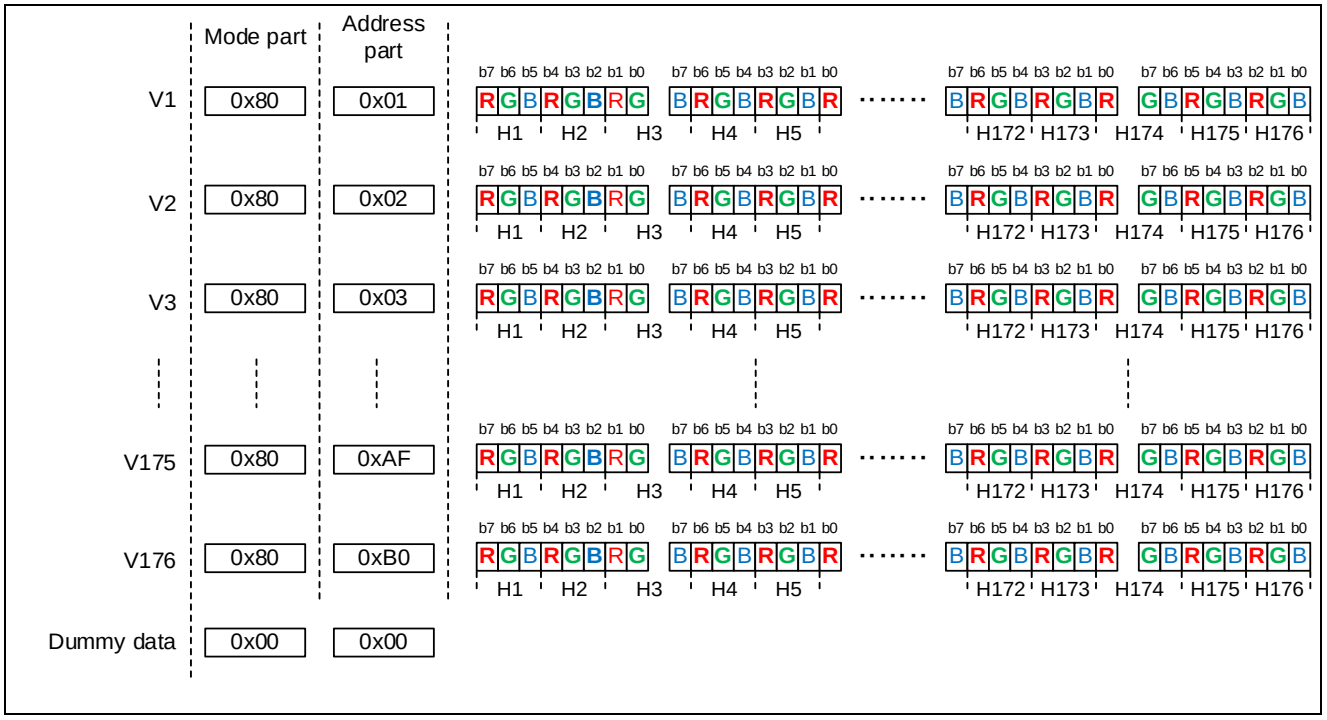


Figure 4-3 Example of Data Arrangement When SMIP_CFG_MAKE_LINE_INFO is 0
(SMIP driver does not add the header or footer during transmission)

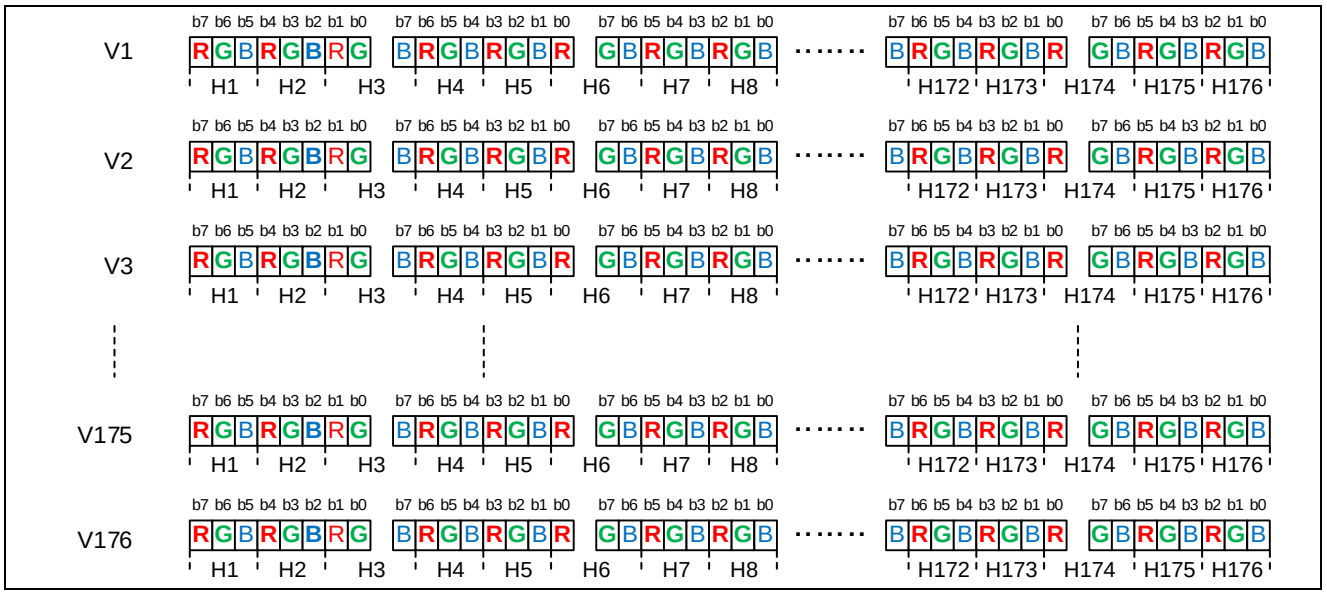


Figure 4-4 Example of Data Arrangement When SMIP_CFG_MAKE_LINE_INFO is 1
(SMIP driver adds the header or footer during transmission)

4.2 Example of Using SHARP's MIP-LCD

4.2.1 Example of Displaying Images on SHARP's MIP-LCD

The SHARP's LCD setting is defined in g_smip_tbl_lcd_info[SMIP_TYPE_SHARP] of r_smip_api.c. For definition of SHARP configuration, see section 4.6.16 SHARP's LCD Setting Definitions.

Figure 4-5 shows an example of setting the SHARP's LCD.

```
#include "r_smip_api.h"

static void callback(uint32_t event);
extern const unsigned char image_sharp[2306];    /* SHARP form image data */
main()
{
    (void)R_SMIP_Open(1, 1000000, &g_smip_tbl_lcd_info[SMIP_TYPE_SHARP] ,callback);
                            /* SPI channel to use: 1; SHARP's LCD; callback function */
    (void)R_SMIP_PowerOn();    /* Execute SMIP power-on sequence */
    (void)R_SMIP_Send(0, 128, image_sharp);
                /* Start address: 0; data line size: 128; image data table */
    while(1);
    {
        R_SYS_SoftwareDelay(1000, SYSTEM_DELAY_UNITS_MILLISECONDS);
        (void)R_SMIP_Send(0, 128, image_sharp);
                                  /* Start address: 0; data line size: 128; image data table */
        R_SYS_SoftwareDelay(1000, SYSTEM_DELAY_UNITS_MILLISECONDS);
        (void)R_SMIP_SendCommand(0x20);    /* All-Clear command */
    }

    while(1);
}

/***********************************************************************************************************
* callback function
***********************************************************************************************************/
static void callback(uint32_t event)
{
    /* Describe the process when all transfers are completed */
    switch(event)
    {
        case SMIP_EVENT_SEND_COMPLETE:
        {
            /* Describe the process when transmission is normally completed */
        }
        break;
        case SMIP_EVENT_SEND_ERROR:
        {
            /* Describe the process when transmission error occurs */
        }
        break;
    }
}
```

Figure 4-5 Example of SMIP LCD Setting and Transmission (SHARP)

4.2.2 Image Data for SHARP's MIP-LCD

Figure 4-6 shows the address map for the SHARP's LCD, Figure 4-7 shows an example of data arrangement when SMIP_CFG_MAKE_LINE_INFO is 0 (the SMIP driver does not add the header or footer during transmission (note 1)), and Figure 4-8 shows an example of data arrangement when SMIP_CFG_MAKE_LINE_INFO is 1 (the SMIP driver adds the header and footer during transmission (note 1)). (Note2)

Note 1. When SMIP_CFG_MAKE_LINE_INFO is set to 0, the automatic header and footer generation function is disabled; the image data including the header and footer must be provided.

Note 2. Here is an example of the arrangement when SMIP_CFG_SPI_BIT_ORDER is 0 (MSB first is selected for SPI bit order). When SMIP_CFG_SPI_BIT_ORDER is 1 (LSB first is selected for the SPI bit order), invert the bit string of the data to be arranged.

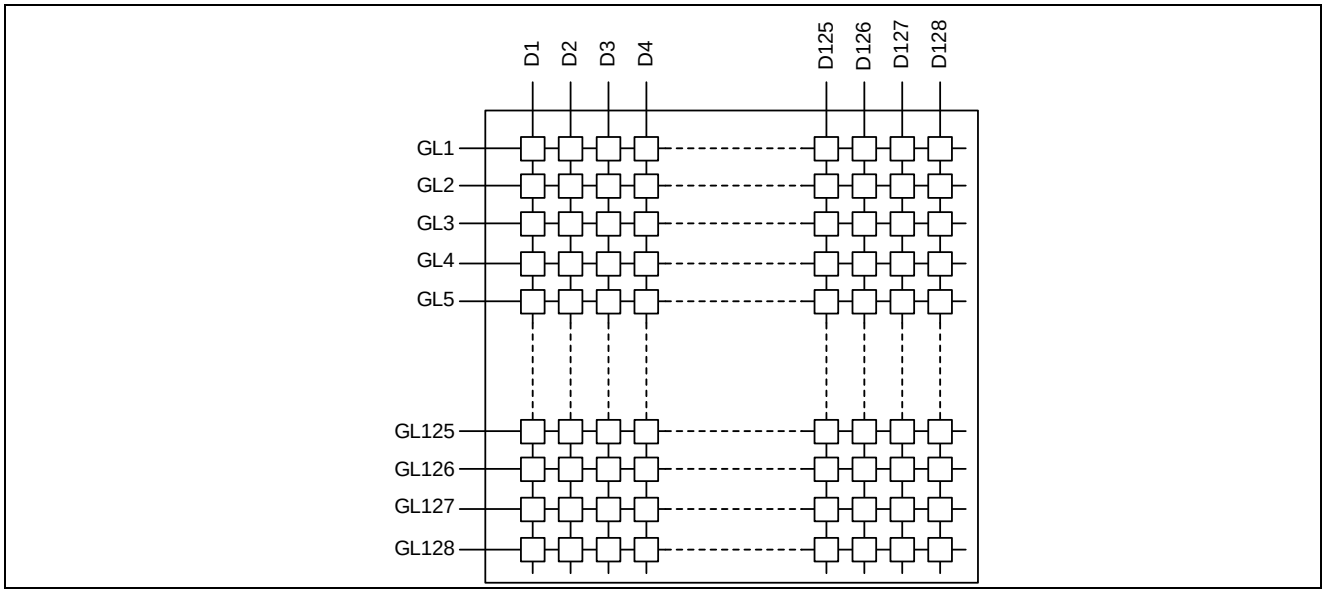


Figure 4-6 Address Map for SHARP's MIP-LCD (monochromatic, 128 x 128)

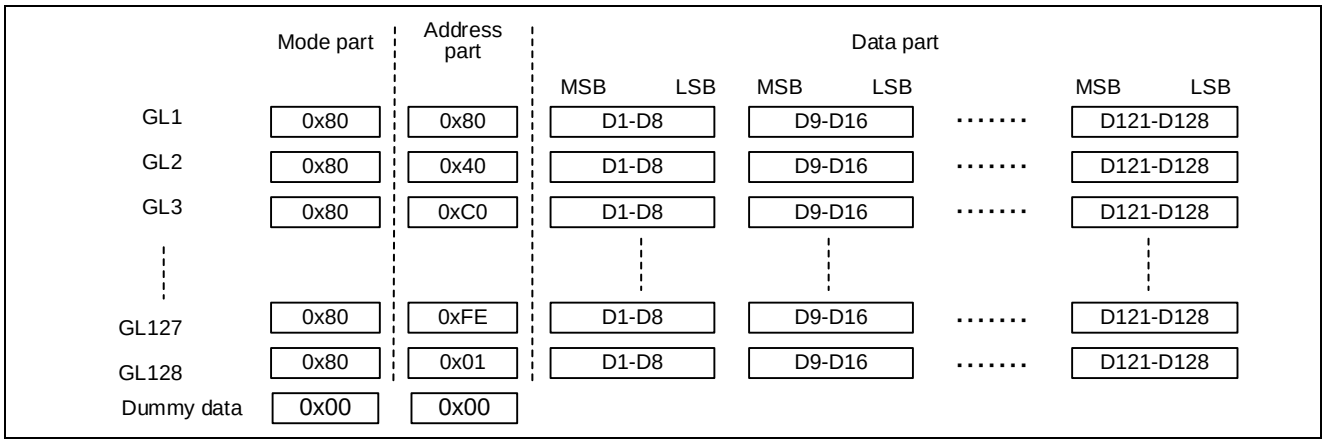


Figure 4-7 Example of Data Arrangement When SMIP_CFG_MAKE_LINE_INFO is 0
(SMIP driver does not add the header or footer during transmission)

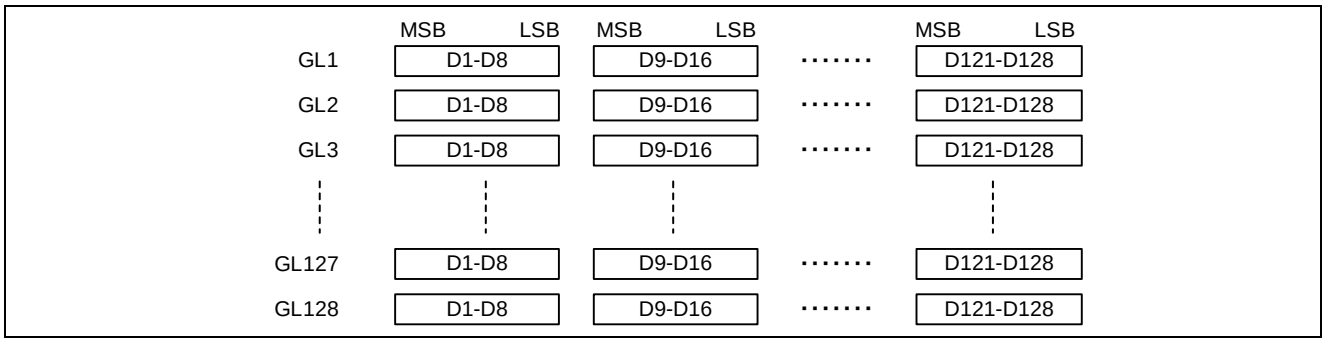


Figure 4-8 Example of Data Arrangement When SMIP_CFG_MAKE_LINE_INFO is 1
(SMIP driver adds the header or footer during transmission)

4.3 Example of Using KYOCERA's MIP-LCD

4.3.1 Example of Displaying Images on KYOCERA’s MIP-LCD

The KYOCERA’s LCD setting is defined in g_smip_tbl_lcd_info[SMIP_TYPE_KYOCERA] of r_smip_api.c. For definition of JDI configuration, see section 4.6.15 KYOCERA's LCD Setting Definitions .

Figure 4-9 shows an example of setting the KYOCERA's LCD.

```
#include "r_smip_api.h"

static void callback(uint32_t event);
extern const unsigned char image_kyocera[9472];    /* KYOCERA form image data */
main()
{
    (void)R_SMIP_Open(1, 1000000, &g_smip_tbl_lcd_info[SMIP_TYPE_KYOCERA] ,callback);
                              /* SPI channel to use: 1; KYOCERA's LCD; callback function */
    (void)R_SMIP_PowerOn();    /* Execute SMIP power-on sequence */
    (void)R_SMIP_Send(0, 256, image_kyocera);
                              /* Start address: 0; data line size: 256; image data table */
    while(1);
}

/***********************************************************************************************
* callback function
***********************************************************************************************/
static void callback(uint32_t event)
{
    /* Describe the process when all transfers are completed */
    switch(event)
    {
        case SMIP_EVENT_SEND_COMPLETE:
        {
            /* Describe the process when transmission is normally completed */
        }
        break;
        case SMIP_EVENT_SEND_ERROR:
        {
            /* Describe the process when transmission error occurs */
        }
        break;
    }
}
```

Figure 4-9 Example of SMIP LCD Setting and Transmission (KYOCERA)

4.3.2 Image Data for KYOCERA's MIP-LCD

Figure 4-10 shows the address map for the KYOCERA's LCD, Figure 4-11 shows an example of data arrangement when SMIP_CFG_MAKE_LINE_INFO is 0 (the SMIP driver does not add the header or footer during transmission (Note 1)), and Figure 4-12 shows an example of data arrangement when SMIP_CFG_MAKE_LINE_INFO is 1 (the SMIP driver adds the header and footer during transmission (Note 1)). (Note2)

Note 1. When SMIP_CFG_MAKE_LINE_INFO is set to 0, the automatic header and footer generation function is disabled; the image data including the header and footer must be provided.

Note 2. Here is an example of the arrangement when SMIP_CFG_SPI_BIT_ORDER is 0 (MSB first is selected for SPI bit order). When SMIP_CFG_SPI_BIT_ORDER is 1 (LSB first is selected for the SPI bit order), invert the bit string of the data to be arranged.

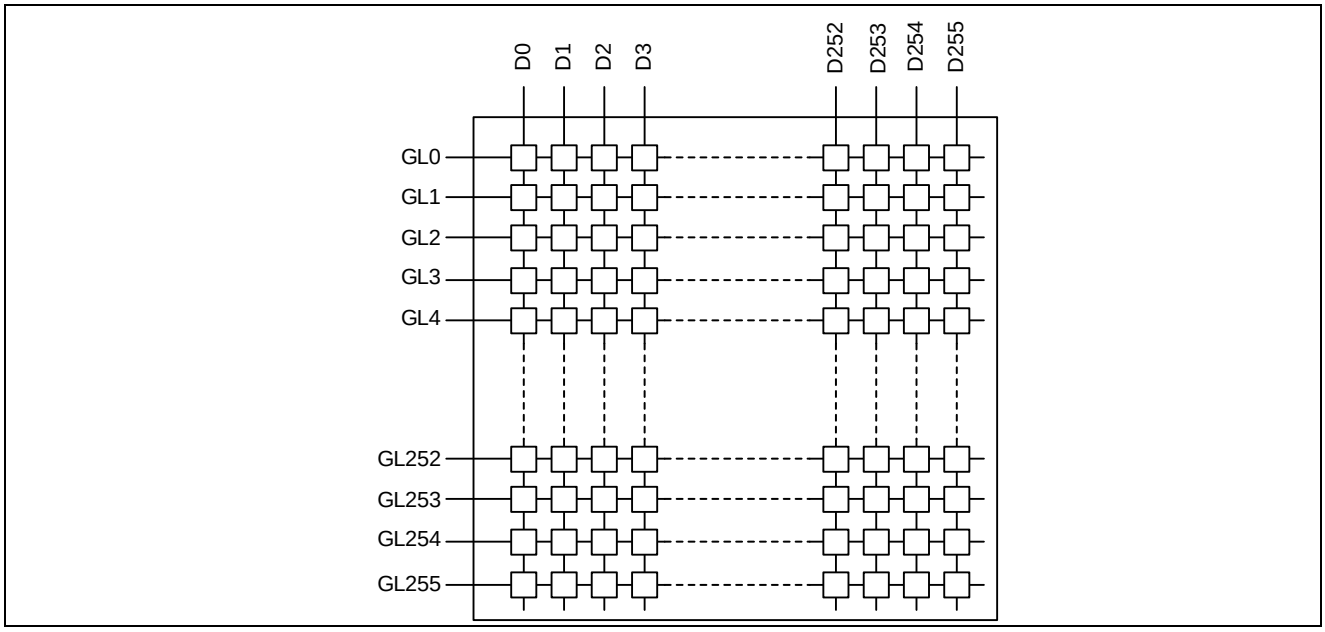


Figure 4-10 Address Map for KYOCERA's MIP-LCD (monochromatic, 256 x 256)

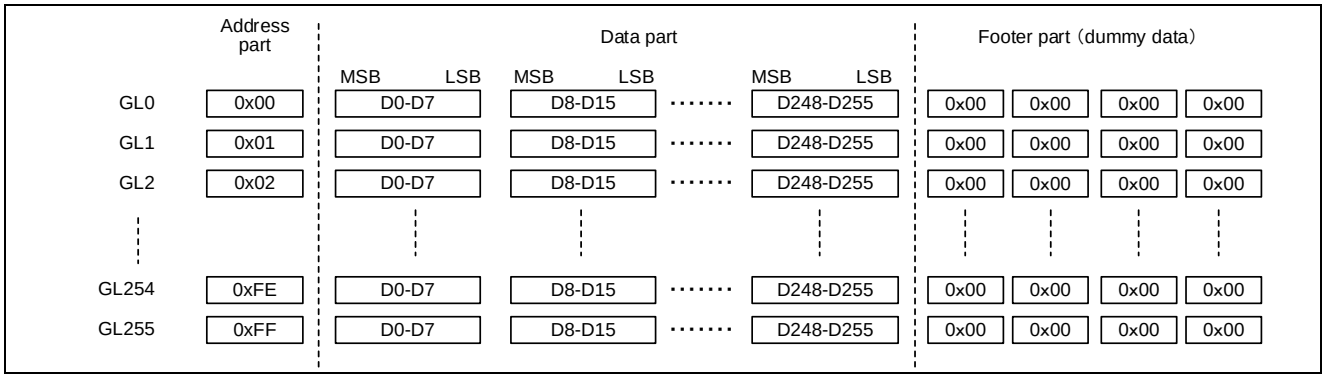


Figure 4-11 Example of Data Arrangement When SMIP_CFG_MAKE_LINE_INFO is 0
(SMIP driver does not add the header or footer during transmission)

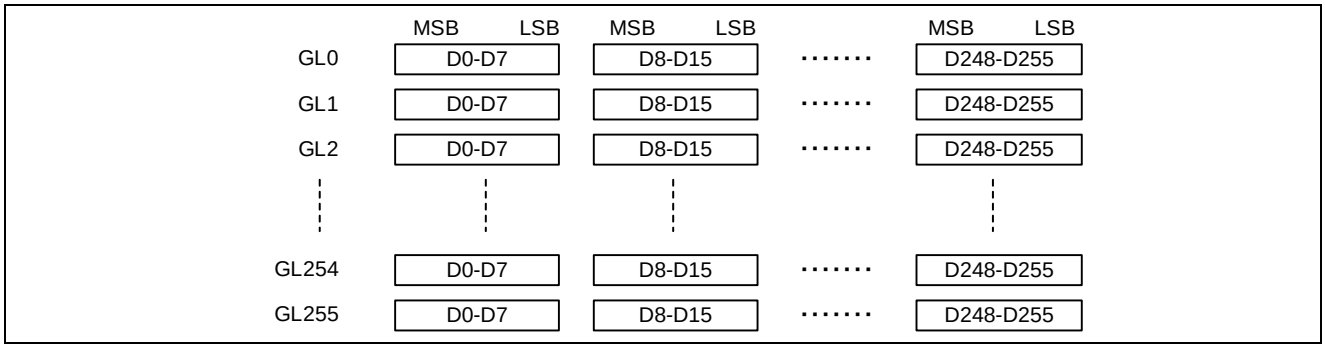


Figure 4-12 Example of Data Arrangement When SMIP_CFG_MAKE_LINE_INFO is 1
(SMIP driver adds the header or footer during transmission)

4.4 Transition to suspend mode

When shutting off the power supply to SPI with the power consumption reduction function, the LCD display is maintained by shifting the SMIP to suspend mode in advance. Use the R_SMIP_Suspend function (Note 1) to enter suspend mode. When the suspend mode is set, the SPI settings and SPI use pins are released (Note 2). Use the R_SMIP_Resume function to reconfigure SPI from suspend mode.

Table 4-1 shows the status of each pin in suspend mode, and Figure 4-13 shows an example of transition to suspend mode using the R_SMIP_Suspend function.

Note 1. R_SMIP_ERROR_BUSY is returned when executed during LCD display processing by SPI.

Note 2. The following functions cannot be used in suspend mode.
Function to output image data (R_SMIP_AllZero, R_SMIP_AllOne, R_SMIP_Send, R_SMIP_SendColor), R_SMIP_SendCommand function, R_SMIP_PowerOn function, and R_SMIP_PowerOff function

Table 4-1 List of pin states in suspend mode

Pin	Status
DISP(RST)	Keep state before transition
EXTCOMIN(VCOM)	VCOM output inversion (Note)
SCS	L output
SI,CLK	Released state

Note Under any of the following conditions, the VCOM output retains the state before transition (inverted output is not performed)

- When the power supply to the ISO1 domain (AGTn function) is cut off with the power consumption reduction function
- When the AGTOn output function is disabled (the macro "SMIP_CFG_AGTO_EN" in r_smip_cfg.h is 0) and the CPU is stopped (in standby mode)

```
#include "r_smip_api.h"

static void callback(uint32_t event);
extern const unsigned char image_sharp[2306];    /* SHARP form image data */
main()
{
    /* Transition to suspend mode can be performed after initialization by Open */
    (void)R_SMIP_Open(1, 1000000, &g_smip_tbl_lcd_info[SMIP_TYPE_SHARP] ,callback);
                            /* SPI channel to use: 1; SHARP's LCD; callback function */
    (void)R_SMIP_PowerOn();    /* Execute SMIP power-on sequence */
    (void)R_SMIP_Send(0, 128, image_sharp);
                /* Start address: 0; data line size: 128; image data table */
    while(1);
    {
        . . .
        /* Transition to suspend mode before shutting off power supply to SPI function */
        while (SMIP_OK != R_SMIP_Suspend())
        {
            /* When LCD display data transmission processing by SPI is in progress,
               SMIP_ERROR_BUSY occurs and transition to suspend mode is not possible */
            R_SYS_SoftwareDelay(1, SYSTEM_DELAY_UNITS_MILLISECONDS);
        }
        /* LCD display is maintained after transition to suspend mode */
        . . .
        /* To update the LCD display after power supply to the SPI function is restored,
            reset the SPI and assign pins to the SPI using the Resume function. */
        (void)R_SMIP_Resume(100000);  /* Specify the SPI communication speed as an argument
                                         (changeable from the value specified in the R_SMIP_Open function)
                                        If the SPI communication speed setting is invalid,
                                        SMIP_ERROR_PARAMETER is returned. */
        . . .
    }
}
```

Figure 4-13 Example of transition to suspend mode using the R_SMIP_Suspend function

4.5 Changing the SPI communication speed

If you want to change the SPI communication speed or if the SPI count source is changed, use the R_SMIP_ReconfigSpiSpeed function to reset the communication speed (Note). An example of resetting the SPI communication speed using the R_SMIP_ReconfigSpiSpeed function is shown in Figure 4-14.

Note R_SMIP_ERROR_BUSY is returned when executed during LCD display processing by SPI communication.

```
#include "r_smip_api.h"

static void callback(uint32_t event);
extern const unsigned char image_sharp[2306];    /* SHARP form image data */
main()
{
    /* SPI communication speed can be changed after initialization by Open */
    (void)R_SMIP_Open(1, 1000000, &g_smip_tbl_lcd_info[SMIP_TYPE_SHARP] ,callback);
    while(1);
    {
        . . .
        /* SPI communication speed changed to 500kHz */
        /* If the SPI communication speed setting is invalid, SMIP_ERROR_PARAMETER is returned. */
        while (SMIP_OK != R_SMIP_ReconfigSpiSpeed(500000))
        {
            /* The SPI communication speed cannot be changed
                while the LCD display data transmission process is in progress.*/
            R_SYS_SoftwareDelay(1, SYSTEM_DELAY_UNITS_MILLISECONDS);
        }
        . . .
    }
}
```

Figure 4-14 Example of SPI communication speed reconfiguration using the R_SMIP_ReconfigSpiSpeed function

4.6 Configurations

For SMIP driver, configuration definitions that can be modified by the user are provided in the r_smip_cfg.h file.

4.6.1 Parameter Checks

Parameter checks in the SMIP are enabled or disabled.

Name: SMIP_CFG_PARAM_CHECKING_ENABLE

Table 4-2 Setting of SMIP_CFG_PARAM_CHECKING_ENABLE

Setting	Description
0	Disables parameter checks Errors relating to validity of arguments appearing in function specifications are not detected.
1 (initial value)	Enables parameter checks Error conditions for parameters appearing in function specifications are detected.

4.6.2 Definition of Setting SCS Low Time (tw_scsl) Wait Process

This sets how to process the SCS low time (tw_scsl) wait.

Name: SMIP_CFG_SCS_WAIT_EN

Table 4-3 Setting of SMIP_CFG_SCS_WAIT_EN

Setting	Description
0	SCS low time (tw_scsl) wait is not processed in the SMIP driver. (Note)
1 (initial value)	SCS low time (tw_scsl) wait is processed in the SMIP driver.

Note When consecutively transmitting data by a callback function, insert the appropriate wait time to satisfy the SCS low width (twSCSL) by the user program.

4.6.3 Definition of Maximum Number of Command Bytes

This sets the maximum number of bytes for transmitting a command.

Name: SMIP_CFG_COMMAND_MAX_BYTE

Table 4-4 Setting of SMIP_CFG_COMMAND_MAX_BYTE

Setting	Description
2 (initial value)	Maximum number of command bytes

4.6.4 Definition of Maximum Buffer Size for Single-Line Transmission

This sets the maximum buffer size for transmitting a single line.

Name: SMIP_CFG_ALL_WRITE_BUFF

Table 4-5 Setting of SMIP_CFG_ALL_WRITE_BUFF

Setting	Description
37 (initial value)	Maximum buffer size for transmitting a single line

4.6.5 EXTMODE Pin Setting Definition

This sets the input level for the EXTMODE pin. Set 1 for KYOCERA's LCD.

Name: SMIP_CFG_EXTMODE

Table 4-6 Setting of SMIP_CFG_EXTMODE

Setting	Description
0	Input level for EXTMODE pin is low. (EXTCOMIN output not provided.) (Note)
1 (initial value)	Input level for EXTMODE pin is high. (EXTCOMIN output not provided.)

Note Use the SendCommand function for COM input.

4.6.6 AGTO Output Setting Definition for VCOM (EXTCOMIN) Output Control

This enables or disables AGTO output for VCOM (EXTCOMIN) control.

Name: SMIP_CFG_AGTO_EN

Table 4-7 Setting of SMIP_CFG_AGTO_EN

Setting	Description
0(initial value)	Uses ports for VCOM (EXTCOMIN) control.
1	Uses AGTOn output for VCOM (EXTCOMIN) control.

4.6.7 VCOM (EXTCOMIN) Output Pin Setting Definitions

These define the pins to be used for VCOM (EXTCOMIN) output.

Table 4-8 List of VCOM Output Pin Setting Definitions

Definition Name	Initial Value	Description
SMIP_CFG_VCOM_PORT	PORT1	Selects PORT1 as the VCOM (EXTCOMIN) pin.
SMIP_CFG_VCOM_PIN	6	Selects PORTi06 as the VCOM (EXTCOMIN) pin. (i indicates a port specified with SMIP_CFG_VCOM_PORT.)

4.6.8 SCS Output Pin Setting Definitions

These define the pins to be used for SCS output.

Table 4-9 List of SCS Setting Definitions

Definition Name	Initial Value	Description
SMIP_CFG_SCS_PORT	PORT1	Selects PORT1 as the SCS pin.
SMIP_CFG_SCS_PIN	3	Selects PORTi03 as the SCS pin. (i indicates a port specified with SMIP_CFG_SCS_PORT.)

4.6.9 RST, DISP Output Pin Setting Definitions

These define the pins to be used for RST output.

Table 4-10 List of RST, DISP Setting Definitions

Definition Name	Initial Value	Description
SMIP_CFG_RST_DISP_PORT	PORT1	Selects PORT1 as the RST pin.
SMIP_CFG_RST_DISP_PIN	5	Selects PORTi05 as the RST pin. (i indicates a port specified with SMIP_CFG_RST_PORT.)

4.6.10 Definition of Transmit Data Information Generation

This enables or disables the automatic header and footer generation function for transmit data information. For the differences between the transmit data information when the automatic header and footer generation function is used and when not used, see chapter 4, Descriptions of Driver Operations.

Name: SMIP_CFG_MAKE_LINE_INFO

Table 4-11 Setting of SMIP_CFG_MAKE_LINE_INFO

Setting	Description
0	Does not add a header or footer during transmission (the SMIP driver does not add the header and footer automatically).
1 (initial value)	Adds a header and a footer during transmission (the SMIP driver adds the header and footer automatically).

4.6.11 VCOM(EXTCOMIN) Timer Setting Definitions

These define the AGT timer settings when using the AGT timer for VCOM (EXTCOMIN) control.

Table 4-12 List of VCOM (EXTCOMIN) Timer Setting Definitions

Definition Name	Initial Value	Description
SMIP_CFG_VCOM_TIMER_CH	0	VCOM (EXTCOMIN) timer AGT channel (0 or 1)
SMIP_CFG_VCOM_TIMER_INT_PRIORITY	3	VCOM (EXTCOMIN) timer interrupt (AGTn_AGTI) priority level Settable range: 0 to 3 (highest: 0, lowest: 3)
SMIP_CFG_VCOM_TIMER_CLOCK	0	AGT timer base clock 0: Subclock (SOCO) 1: Low-speed on-chip oscillator (LOCO)
SMIP_CFG_VCOM_TIMER_LPM	1	AGT timer low power consumption mode setting 0: Normal mode 1: Low power consumption mode

4.6.12 Timeout Value Setting Definition

This specifirsthe timeout time for power-on and power-off sequences.

Name: SMIP_CFG_TIMEOUT

Table 4-13 Setting of SMIP_CFG_TIMEOUT

Setting	Description
2000	Timeout time for power-on and power-off sequences

4.6.13 SPI Communication Bit Order Setting Definition

Specify the bit order used in SPI communication.

Name: SMIP_CFG_SPI_BIT_ORDER

Table 4-14 Setting of SMIP_CFG_SPI_BIT_ORDER

Setting	Description
0(initial value)	MSB
1	LSB

4.6.14 JDI‘s LCD Setting Definitions

When the JDI's LCD is specified by the Open function, the driver operates according to the JDI's LCD setting definitions. JDI's LCD settings are defined in g_smip_tbl_lcd_info[SMIP_TYPE_JDI] of the st_smip_cfg_t structure table. The following table lists the JDI's LCD setting definitions.

Table 4-15 List of JDI‘s LCD Setting Definitions

Definition Name	Initial Value	Description	st_smip_cfg_t Member
SMIP_CFG_JDI_COLOR_MODE	0	Color mode setting 0: 3-bit color mode 1: 4-bit color mode	type:b0
SMIP_CFG_JDI_MODE_SIZE	6	Mode bit size	mode_size
SMIP_CFG_JDI_MODE_DMY_SIZE	0	Mode dummy bit size	mode_dmy_size
SMIP_CFG_JDI_ADDR_SIZE	10	Address bit size	address_size
SMIP_CFG_JDI_DUMMY_SIZE	16	Dummy bit size	end_dummy_size
SMIP_CFG_JDI_DISP_WIDTH	176	LCD width	disp_width
SMIP_CFG_JDI_DISP_LINE	176	LCD line size	disp_line
SMIP_CFG_JDI_VCOM_CLK	120 * 10	VCOM (EXTCOMIN) cycle (x 0.1 Hz)	vcom_clk
SMIP_CFG_JDI_T_MEMORY_INI	1000	Pixel memory initialization time (us)	t_memory_init
SMIP_CFG_JDI_T_COM_INI	30	COM initialization time (us)	t_com_init
SMIP_CFG_JDI_TS_SCS	6	SCS setup time (us)	ts_scs
SMIP_CFG_JDI_TH_SCS	2	SCS hold time (us)	th_scs
SMIP_CFG_JDI_TW_SCSL	6	SCS low time (us)	tw_scsl
SMIP_CFG_JDI_FIRST_ADDR	1	Initial line address	first_addr

4.6.15 KYOCERA's LCD Setting Definitions

When the KYOCERA's LCD is specified by the Open function, the driver operates according to the KYOCERA's LCD setting definitions. KYOCERA's LCD settings are defined in g_smip_tbl_lcd_info[SMIP_TYPE_KYOCERA] of the st_smip_cfg_t structure table. The following table lists the KYOCERA's LCD setting definitions.

Table 4-16 List of KYOCERA‘s LCD Setting Definitions

Definition Name	Initial Value	Description	st_smip_cfg_t Member
SMIP_CFG_KYOCERA_COLOR_MODE	0	Color mode setting 0: 3-bit color mode 1: 4-bit color mode	type:b0
SMIP_CFG_KYOCERA_MODE_SIZE	0	Mode bit size	mode_size
SMIP_CFG_KYOCERA_MODE_DMY_SIZE	0	Mode dummy bit size	mode_dmy_size
SMIP_CFG_KYOCERA_ADDR_SIZE	8	Address bit size	address_size
SMIP_CFG_KYOCERA_DUMMY_SIZE	32	Dummy bit size	end_dummy_size
SMIP_CFG_KYOCERA_DISP_WIDTH	256	LCD width	disp_width
SMIP_CFG_KYOCERA_DISP_LINE	256	LCD line size	disp_line
SMIP_CFG_KYOCERA_VCOM_CLK	0.5 * 10	VCOM cycle (x 0.1 Hz)	vcom_clk
SMIP_CFG_KYOCERA_T_MEMORY_INI	1000	Pixel memory initialization time (us)	t_memory_init
SMIP_CFG_KYOCERA_T_COM_INI	1000	COM initialization time (us)	t_com_init
SMIP_CFG_KYOCERA_TS_SCS	4	SCS setup time (us)	ts_scs
SMIP_CFG_KYOCERA_TH_SCS	4	SCS hold time (us)	th_scs
SMIP_CFG_KYOCERA_TW_SCSL	10	SCS low time (us)	tw_scsl
SMIP_CFG_KYOCERA_TS_VCOM	4	VCOM setup time (us)	ts_vcom
SMIP_CFG_KYOCERA_TH_VCOM	1	VCOM hold time (us)	th_vcom
SMIP_CFG_KYOCERA_FIRST_ADDR	0	Initial line address	first_addr

4.6.16 SHARP's LCD Setting Definitions

When the SHARP's LCD is specified by the Open function, the driver operates according to the SHARP's LCD setting definitions. SHARP's LCD settings are defined in g_smip_tbl_lcd_info[SMIP_TYPE_SHARP] of the st_smip_cfg_t structure table. The following table lists the SHARP's LCD setting definitions.

Table 4-17 List of SHARP's LCD Setting Definitions

Definition Name	Initial Value	Description	st_smip_cfg_t Member
SMIP_CFG_SHARP_COLOR_MODE	0	Color mode setting 0: 3-bit color mode 1: 4-bit color mode	type:b0
SMIP_CFG_SHARP_MODE_SIZE	3	Mode bit size	mode_size
SMIP_CFG_SHARP_MODE_DMY_SIZE	5	Header dummy bit size	mode_dmy_size
SMIP_CFG_SHARP_ADDR_SIZE	8	Address bit size	address_size
SMIP_CFG_SHARP_DUMMY_SIZE	16	Dummy bit size	end_dummy_size
SMIP_CFG_SHARP_DISP_WIDTH	128	LCD width	disp_width
SMIP_CFG_SHARP_DISP_LINE	128	LCD line size	disp_line
SMIP_CFG_SHARP_VCOM_CLK	60*10	VCOM (EXTCOMIN) cycle (x 0.1 Hz)	vcom_clk
SMIP_CFG_SHARP_T_MEMORY_INI	0	Pixel memory initialization time (us)	t_memory_init
SMIP_CFG_SHARP_T_COM_INI	30	COM initialization time (us)	t_com_init
SMIP_CFG_SHARP_TS_SCS	6	SCS setup time (us)	ts_scs
SMIP_CFG_SHARP_TH_SCS	2	SCS hold time (us)	th_scs
SMIP_CFG_SHARP_TW_SCSL	6	SCS low time (us)	tw_scsl
SMIP_CFG_SHARP_FIRST_ADDR	1	Initial line address	first_addr

4.6.17 Function Allocation to RAM

This initializes the settings for executing specific functions of the SMIP driver in RAM.

This configuration definition for setting function allocation to RAM has function-specific definitions.

Name: SMIP_CFG_SECTION_xxx

A function name xxx should be written in all capital letters.

Example: R_SMIP_Open function → SMIP_CFG_R_SMIP_OPEN

Table 4-18 Setting of SMIP_CFG_SECTION_xxx

Setting	Description
SYSTEM_SECTION_CODE	Does not allocate the function to RAM.
SYSTEM_SECTION_RAM_FUNC	Allocates the function to RAM.

Table 4-19 Initial State of Function Allocation to RAM

No.	Function Name	Allocation to RAM
1	R_SMIP_GetVersion	
2	R_SMIP_Open	
3	R_SMIP_Close	
4	R_SMIP_PowerOn	
5	R_SMIP_PowerOff	
6	R_SMIP_AllZero	
7	R_SMIP_AllOne	
8	R_SMIP_Send	
9	R_SMIP_SendColor	
10	R_SMIP_SendCommand	
11	R_SMIP_Suspend	
12	R_SMIP_Resume	
13	R_SMIP_ReconfigSpiSpeed	
14	r_smip_cb_spi_event	✔
15	r_smip_cb_vcom_timer_event	✔

5. Usage Notes

5.1 Registering AGTI Interrupts to NVIC

It is required to set the AGT timer interrupts (AGTn_AGTI) if SMIP_CFG_EXTMODE is 1 or transmission control synchronizes with VCOM output. For details, see section 1.2, Setting AGT Used with SMIP.

5.2 Setting SPI Driver

This driver uses the SPI driver. For setting the SPI, see chapter 1, Overview.

5.3 Restrictions on Function Execution

The functions used to output image data by using this driver (R_SMIP_AllZero, R_SMIP_AllOne, R_SMIP_Send, and R_SMIP_SendColor) and the R_SMIP_SendCommand function can be used only after the R_SMIP_PowerOn function is executed.

5.4 Setting RST, SCS, and VCOM Control Pins

The RST and SCS signals used by this driver are controlled with port outputs. The RST and SCS control pins must not be assigned to peripheral functions. Peripheral function settings for pins can be changed with pin.c. Similarly, the VCOM control pin must not be assigned to a peripheral function when AGTOn is not used for the VCOM signal.

5.5 Operation when Suspend is executed during ReconfigSpiSpeed processing

If the R_SMIP_Suspend function is executed by an interrupt when the SPI communication speed is changed by the R_SMIP_ReconfigSpiSpeed function, incorrect operation may occur. Do not execute the R_SMIP_Suspend function while executing the R_SMIP_ReconfigSpiSpeed function.

6. Reference Documents

User's Manual: Hardware

RE01 1500KB Group User's Manual: Hardware R01UH0796
RE01 256KB Group User's Manual: Hardware R01UH0894
(The latest version can be downloaded from the Renesas Electronics website.)

RE01 Group CMSIS Package Startup Guide

RE01 1500KB, 256KB Group Startup Guide to Development Using CMSIS Package R01AN4660

(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

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct.10.2019	—	First edition issued
1.01	Nov.19.2019	Program	Added the following functions ・R_SMIP_Suspend ・R_SMIP_Resume ・R_SMIP_ReconfigSpiSpeed
		—	Clerical error correction
		6~9,34	Modification to comment out default pin setting of pin.c
		6, 7	Fixed the SCS (RST) signal precautions and Example of Setting Pins
		10	Added driver API ・R_SMIP_Suspend ・R_SMIP_Resume ・R_SMIP_ReconfigSpiSpeed
		15	Add "suspend state"
		24	Added "Transition to suspend mode"
		25	Added "Changing the SPI communication speed"
		33	Added "Operation when Suspend function is executed during R_SMIP_ReconfigSpiSpeed function"
1.03	Mar.4.2020	4, 18, 21, 24, 31	Added SPI communication bit order setting
		21,22,24,25	Error correction of data array
		Program	Added SPI communication bit order setting (SMIP_CFG_SPI_BIT_ORDER) AGT module stop processing of internal function r_smip_tm_lpm_on () is deleted.
1.04	Mar.5.2020	—	Compatible with 256KB group
		Program (256KB)	
1.05	Apr.17.2020	Program (1500KB)	Fixed ToolNews R20TS0574
		Program (256KB)	Fixed the problem that AGT register value could not be rewritten when Kyocera LCD is used.
		Program	The setting of SMIP_CFG_SPI_BIT_ORDER is reflected in bit order of SPI communication when R_SMIP_Resume function is executed.
		—	Clerical error correction

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 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.
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; 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.
6. 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.
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. 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.
8. 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.
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 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.
10. 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.
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.

(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.4.0-1 November 2017)

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/.