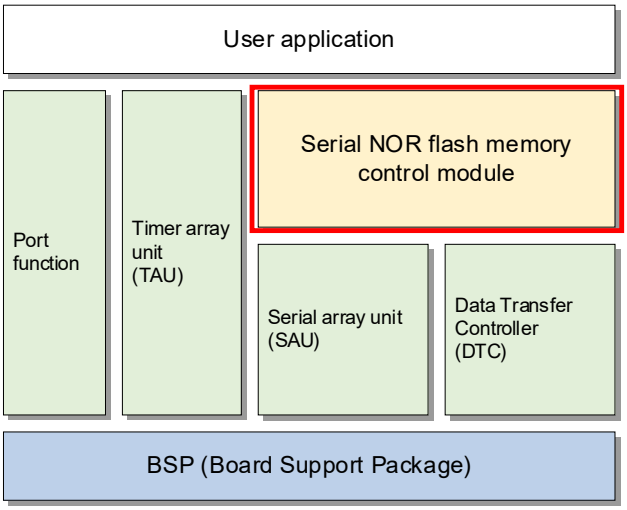
RL78 Family

Serial NOR Flash Memory Control Module Software Integration System

Overview

This application note describes the serial NOR flash memory control module conforming to the Software Integration System (SIS). This module controls serial NOR flash memory by using simple SPI (CSI) mode of the serial array unit (SAU) in RL 78 family microcomputers manufactured by Renesas Electronics. This module is referred to as “serial NOR flash memory control module” hereafter.

The following shows the position of this module.



Target Devices

- RL78/G15, RL78/G16, RL78/G22, RL78/G23, and RL78/G24

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

Target Compilers

- Renesas Electronics RL78 Family C Compiler Package
- LLVM for Renesas RL78
- IAR C/C++ Compiler for Renesas RL78

For details about the confirmed operational aspects of each compiler, see section 7.1 Operation Confirmation Environment.

Related Documents

- RL78 Family Board Support Package Module Software Integration System (R01AN5522)
- Smart Configurator User's Manual: RL78 API Reference (R20UT4852)

Contents

1. Overview

1.1 Serial NOR Flash Memory Control Module

Combined use of this module and Renesas software modules (such as Smart Configurator code generation) which are available free of charge (see Figure 1-1 and Table 1-1) enables control of serial NOR flash memory manufactured by Renesas Electronics and Macronix.

For details about serial NOR flash memory to be controlled, see section 7.2 Controllable Serial NOR Flash Memory Products.

This module is installed in a project as an API. For details about how to install this module, see section 2.11, Adding the SIS Module.

Figure 1-1 shows an application configuration example when serial NOR flash memory is controlled by using the serial NOR flash memory control module.

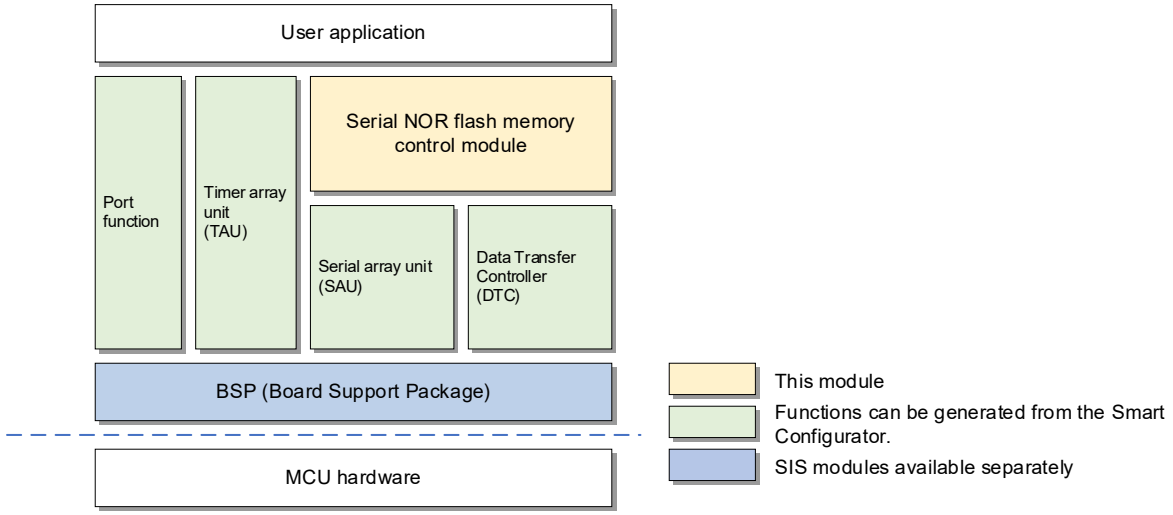


Figure 1-1 Application Configuration Example

Table 1-1 lists the modules used in the application configuration.

Table 1-1 List of Modules Used in the Application Configuration

Name	Component Name	Component Type
Board Support Package Module (BSP)	Board Support Packages	RL78 Software Integration System
Serial array unit (SAU)	SPI (CSI) communication	Code generation
Data Transfer Controller (DTC)	Data Transfer Controller	Code generation
Timer array unit (TAU)	Interval timer	Code generation
Port function	Port	Code generation

(1) Board Support Package Module (BSP)

This component specifies initial settings such as clock setting.

For details about the BSP, refer to the BSP document.

https://www.renesas.com/jp/ja/document/apn/rl78-family-board-support-package-module-using-software-integration-system

(2) Serial array unit (SAU)

This component is used for SPI communications with the serial NOR flash memory.

(3) Data Transfer Controller (DTC)

This component is used to read to or write from CSI registers without using a CPU during SPI communications with the serial NOR flash memory.

(4) Timer Array Unit (TAU)

This component is used to detect a timeout when DTC transfer is used.

(5) Port function

Select a chip select pin and set it to high output. This is the initial setting of the chip select pin before this SIS module is used.

(6) Pin settings

When SPI communication is selected for the Smart Configurator, the pin of the serial array unit (SAU) is also set.

To change the pin to be used for SPI communication, you can change the setting on the Pins page of the Smart Configurator.

For details, refer to the “RL78 Smart Configurator User’s Guide” for your development environment.

RL78 Smart Configurator | Renesas

1.2 API Overview

Table 1-2 shows the API functions included in this module.

Table 1-2 API Function List

Function	Description
R_NOR_FLASH_Open()	Initialization processing of this control software
R_NOR_FLASH_Close()	End processing of this control software
R_NOR_FLASH_ReadStatus()	Status Register 1 read processing
R_NOR_FLASH_ReadStatus2()	Status Register 2 read processing
R_NOR_FLASH_ReadStatus3()	Status Register 3 read processing
R_NOR_FLASH_WriteStatus()	Status Register 1 write processing
R_NOR_FLASH_WriteStatus2()	Status Register 2 write processing
R_NOR_FLASH_WriteStatus3()	Status Register 3 write processing
R_NOR_FLASH_SetWriteProtect()	Write protect set processing
R_NOR_FLASH_WriteDisable()	Write-protect command processing
R_NOR_FLASH_ReadData()	Data read processing
R_NOR_FLASH_WriteDataPage()	Data write (one-page write) processing
R_NOR_FLASH_Erase()	Erase processing
R_NOR_FLASH_CheckBusy()	Busy check processing
R_NOR_FLASH_ReadId()	ID read processing
R_NOR_FLASH_GetMemoryInfo()	Memory size acquisition processing
R_NOR_FLASH_ReadConfiguration()	Configuration Register read processing
R_NOR_FLASH_WriteConfiguration()	Configuration Register write processing
R_NOR_FLASH_Set4byteAddressMode()	4-byte address mode set processing
R_NOR_FLASH_ReadSecurity()	Security Register read processing
R_NOR_FLASH_ReadDataSecurityPage()	Data security page read processing
R_NOR_FLASH_WriteDataSecurityPage()	Data security page write processing
R_NOR_FLASH_GetVersion()	Version information acquisition processing for this control software
R_NOR_FLASH_Interval()*1	Interval timer counter processing
R_NOR_FLASH_SendendNotification()	Transmission completion callback processing for SPI communication

Note: 1. When DTC transfer is used, this function must be called at 1-ms intervals by using a hardware timer or software timer to detect a timeout.

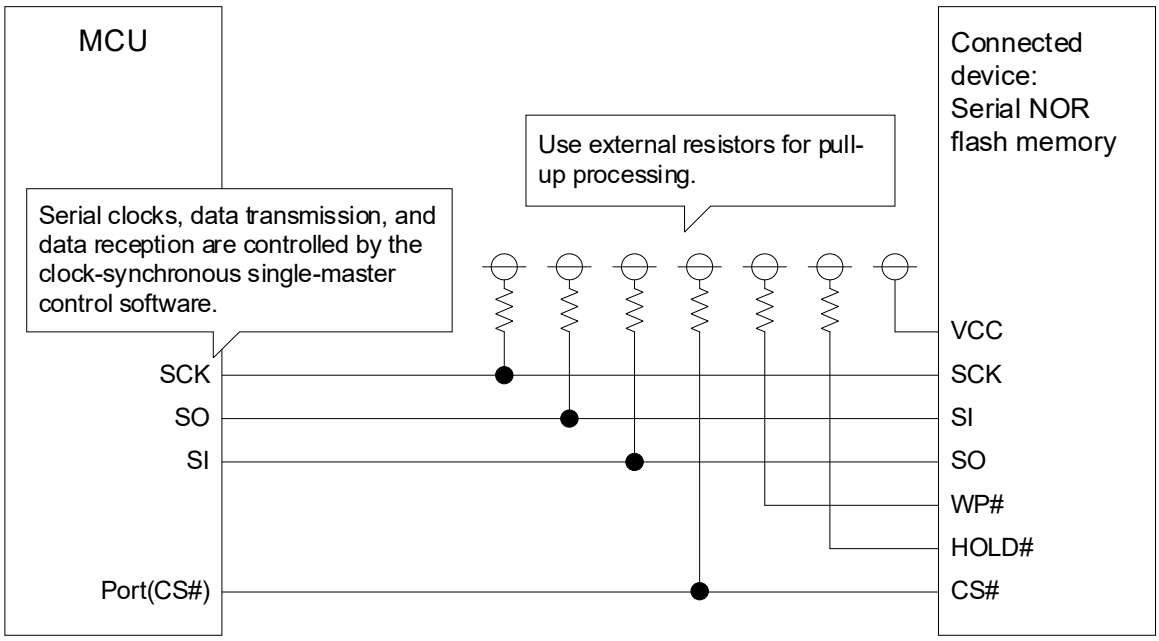
1.3 Hardware Settings

1.3.1 Hardware configuration example

Figure 1-2 shows the pin connection diagram. See the pins and functions in Table 1-3, and then allocate the pins to the appropriate pins of the MCU.

To achieve high-speed operation, consider adding damping resistors or capacitors to improve the circuit matching of the various signal lines.

This module only supports Single-SPI control.



- The pin names in the figure are those of serial NOR flash memory manufactured by Renesas Electronics. The pin names of the serial NOR flash memory differ depending on the product.
- In the example, WP# and HOLD# are not used. When using WP# and HOLD#, check the specifications of the connected device.

Figure 1-2 Pin Connection Diagram (Single-SPI Configuration Example)

Table 1-3 Used Pins and Functions

Pin Name	I/O	Description
SCK	Output	Clock output
SO	Output	Master data output
SI	Input	Master data input
Port (Port(CS#) in Figure 1-2)	Output	Chip select (CS#) output

1.4 Software Description

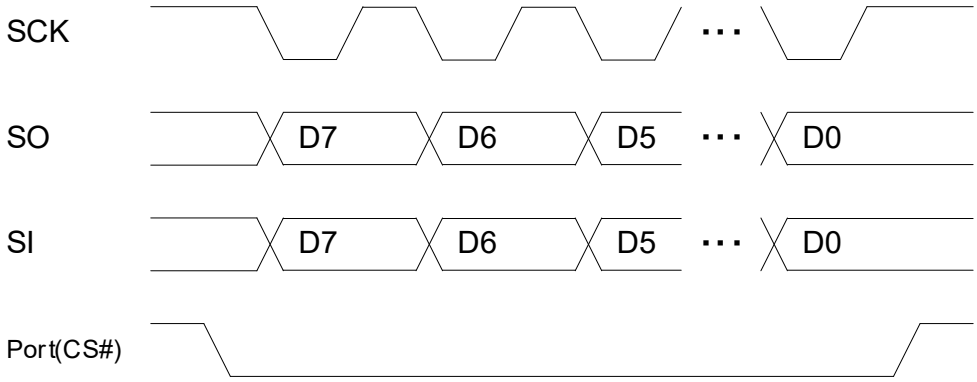
1.4.1 Operation overview

Clock-synchronous single-master control using internal clock is implemented by using the clock-synchronous serial communication feature of the microcomputer.

Confirm the available serial clock frequency by referring to the “User’s Manual: Hardware” of the microcomputer and the data sheet of the slave device.

This module only supports Single-SPI control.

Control is provided in SPI mode 3 (CPOL=1, CPHA=1) shown in Figure 1-3.



- For transmission from the microcomputer to the slave device: Transmit data output starts at the falling edge of the transfer clock.
  Transmit data is confirmed at the rising edge of the transfer clock.
- For reception from the slave device to the microcomputer: Receive data input is achieved at the rising edge of the transfer clock.
- MSB-first mode transfer
- Level of the CLK pin when no transfer is performed: H

Figure 1-3 Timing of a Controllable Slave Device

1.4.2 Chip select pin control for the serial NOR flash memory

The chip select pin of the serial NOR flash memory is connected to a port of the microcomputer and controlled by general port output of the microcomputer.

This module controls the following duration by using software wait to provide the wait time for chip select setup of the serial NOR flash memory: From the falling edge of the chip select (MCU Port(CS#)) signal of the serial NOR flash memory to the falling edge of the clock (MCU SCK) signal of the serial NOR flash memory.

This module controls the following duration by using software wait to provide the wait time for chip select hold of the serial NOR flash memory: From the rising edge of the clock (MCU SCK) signal of the serial NOR flash memory to the rising edge of the chip select (MCU Port(CS#)) signal of the serial NOR flash memory.

For this module, approximately 1 us is set for the wait time for chip select setup and for chip select hold.

2. API information

Operation of this module has been confirmed under the following conditions.

2.1 Hardware Requirements

Your microcomputer must support the following feature:

- SPI (CSI)

2.2 Software Requirements

This module depends on the following package.

- r_bsp (Rev.1.62 or later)

This module also depends on the following code generation modules:

- Serial array unit (SAU)
- Data Transfer Controller (DTC)
- Timer array unit (TAU)
- Port function

2.3 Supported Tool Chains

Operation of this module has been confirmed with the tool chains shown in section 7.1 Operation Confirmation Environment.

2.4 Interrupt Vector to Use

None

2.5 Header Files

All API calls and the interface definitions to use are described in r_nor_flash_rl78_if.h.

Configuration options for each build are selected in r_nor_flash_rl78_config.h.

2.6 Integer Types

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

2.7 Settings at Compilation

The configuration option settings for this module are specified in r_nor_flash_rl78_config.h.

The configuration options can be set in the software component configuration screen by using the Smart Configurator. The set values are automatically reflected to r_nor_flash_rl78_config.h when the module is added. The following table describes the option names and settings.

Configuration options in r_nor_flash_rl78_config.h	
NOR_FLASH_CFG_PARAM_CHECKING_ENABLE Note: The default value is BSP_CFG_PARAM_CHECKING_ENABLE.	1: Parameter check processing is included in the code during build. 0: Parameter check processing is omitted in the code during build. If BSP_CFG_PARAM_CHECKING_ENABLE is specified for this option, the system default setting is used.
NOR_FLASH_CFG_WEL_CHK Note: The default value is 1 (Enabled).	Select whether to check the WEL bit after the Write Enable command is issued. (1: Enabled, 0: Disabled)
NOR_FLASH_CFG_DEVICE Note: The default value is NOR_FLASH_TYPE_AT25SF.	Select the serial NOR flash memory to be controlled. NOR_FLASH_TYPE_AT25SF: Renesas Electronics AT25SF NOR_FLASH_TYPE_AT25QF: Renesas Electronics AT25QF NOR_FLASH_TYPE_MX25L: Macronix MX25L NOR_FLASH_TYPE_MX66L: Macronix MX66L NOR_FLASH_TYPE_MX25R: Macronix MX25R
NOR_FLASH_CFG_SIZE Note: The default value is NOR_FLASH_SIZE_64M: 64 M-bit (8 M bytes).	Select the capacity of the serial NOR flash memory to be controlled. NOR_FLASH_SIZE_4M: 4 M-bit (512 K bytes) NOR_FLASH_SIZE_8M: 8 M-bit (1 M bytes) NOR_FLASH_SIZE_16M: 16 M-bit (2 M bytes) NOR_FLASH_SIZE_32M: 32 M-bit (4 M bytes) NOR_FLASH_SIZE_64M: 64 M-bit (8 M bytes) NOR_FLASH_SIZE_128M: 128 M-bit (16 M bytes) NOR_FLASH_SIZE_256M: 256 M-bit (32 M bytes) NOR_FLASH_SIZE_1G: 1 G-bit (128 M bytes)
NOR_FLASH_CFG_CS_PORTNO Note: The default value is “0”.	Specify the port number to which the CS# pin is allocated. The port numbers must be configured with 0 to 15.
NOR_FLASH_CFG_CS_BITNO Note: The default value is “6”.	Specify the bit number to which the CS# pin is allocated. The bit numbers must be configured with 0 to 7.
NOR_FLASH_CFG_MODE_TRNS Note: The default value is NOR_FLASH_TRNS_CPU, which specifies CPU transfer (software transfer).	NOR_FLASH_TRNS_CPU: CPU transfer (software transfer) NOR_FLASH_TRNS_DTC: DTC transfer
NOR_FLASH_CFG_DRVR_CH Note: The default value is NOR_FLASH_DRVR_CH3: CSI11.	Specify the channel number used for SPI communication. NOR_FLASH_DRVR_CH0: CSI00 NOR_FLASH_DRVR_CH1: CSI01 NOR_FLASH_DRVR_CH2: CSI10 NOR_FLASH_DRVR_CH3: CSI11 NOR_FLASH_DRVR_CH4: CSI20 NOR_FLASH_DRVR_CH5: CSI21 NOR_FLASH_DRVR_CH6: CSI30 NOR_FLASH_DRVR_CH7: CSI31
NOR_FLASH_CFG_DTCD_NO Note: The default value is “0”.	When using the DTC, specify the DTC control data number used for SPI communication. Specify the DTC control data number in the range from 0 to 22.

2.8 Code Size

The following table shows the ROM size, RAM size, and maximum available stack size for this module.

The sizes of ROM (code and constants) and RAM (global data) are determined by the configuration option during build described in section 2.7 Settings at Compilation.

The values in the table below have been confirmed under the following conditions.

Module revision: r_nor_flash rev.1.00

Compiler versions: Renesas Electronics RL78 Family C Compiler Package V1.13.00
(Default settings of the integrated development environment)
LLVM for Renesas RL78 10.0.0.202312
(Default settings of the integrated development environment)
IAR C/C++ Compiler for Renesas RL78 version 5.10.3
(Default settings of the integrated development environment)

Configuration options: Default settings

ROM, RAM, and Stack Code Sizes

Device	Category	Memory Used							
		Renesas Compiler				LLVM Compiler		IAR Compiler	
		Optimization level: -Olite		Optimization level: -Odefault		Optimization level (-Og)		Low level	
		Parameter check		Parameter check		Parameter check		Parameter check	
		Provided	None	Provided	None	Provided	None	Provided	None
RL78/ G23	ROM	2883	2608	2725	2511	3208	2926	2803	2531
	RAM	22		22		22		22	
	Maximum available stack size	150		138		158		116	

2.9 Parameters

This section describes the structures used as parameters of API functions. These structures are described in r_nor_flash_rl78_if.h along with the prototype declarations of the API functions.

(1) st_nor_flash_info_t structure definition

```
/* Flash memory information */
typedef struct
{
    uint32_t        addr;           /* Start address specified by a command */
    uint16_t        cnt;            /* Number of bytes to be read or written */
    uint16_t        data_cnt;       /* Counter used by this module */
    uint8_t       * p_data;         /* Data storage destination pointer */
} st_nor_flash_info_t;              /* 10 bytes */
```

(2) st_nor_flash_mem_info_t structure definition

```
/* Flash memory size information */
typedef struct
{
    uint32_t        mem_size;       /* Maximum memory size */
    uint32_t        wpag_size;      /* Write page size */
} st_nor_flash_mem_info_t;          /* 8 bytes */
```

(3) st_nor_flash_erase_info_t structure definition

```
/* Flash memory erase information */
typedef struct
{
    uint32_t        addr;           /* Start address specified by a command */
    e_nor_flash_erase_mode_t  mode; /* Erase mode */
} st_nor_flash_erase_info_t;        /* 6 bytes */
```

(4) st_nor_flash_reg_info_t structure definition

```
/* Flash memory register information */
typedef struct
{
    uint8_t         status;         /* Status Register */
    uint8_t         config1;        /* Configuration Register-1 */
    uint8_t         config2;        /* Configuration Register-2*/
    uint8_t         rsv[1];
} st_nor_flash_reg_info_t;          /* 4 bytes */
```

2.10 Return Values and Error Codes

This section shows the return values of API functions. This enumerated type is described in r_nor_flash_rl78_if.h along with the prototype declarations of the API functions.

If an error occurs during processing of an API function of this module, the API function returns an error code in the return value.

Table 2-1 shows the Error Codes. Note that values not included in the table are reserved for future expansion.

Table 2-1 Error Codes

Macro Definition	Value	Meaning
NOR_FLASH_SUCCESS_BUSY	1	Normal end: Serial NOR flash memory is in the busy state.
NOR_FLASH_SUCCESS	0	Normal end
NOR_FLASH_ERR_PARAM	-1	Parameter error
NOR_FLASH_ERR_HARD	-2	Hardware error
NOR_FLASH_ERR_NOT_OPEN	-3	Error: Module not open
NOR_FLASH_ERR_ALREADY_OPEN	-4	Error: Module already open
NOR_FLASH_ERR_WEL_CHK	-5	Write Enable check error
NOR_FLASH_ERR_NON_SUPPORTED_API	-6	Unsupported API

2.11 Adding the SIS Module

This module must be added for each project used.

(1) Adding the SIS module by using the Smart Configurator in e^2 studio
Use the Smart Configurator in e^2 studio to automatically add the SIS module to the user project. For details, refer to the application note “RL78 Smart Configurator User's Guide: e^2 studio (R20AN0579)”.

(2) Adding the SIS module by using the Smart Configurator in IAREW
Use the Smart Configurator Standalone version in IAREW to automatically add the SIS module to the user project. For details, refer to the application note “RL78 Smart Configurator User's Guide: IAREW (R20AN0581)”.

(3) Adding the SIS module by using the Smart Configurator in CS+
Use the Smart Configurator Standalone version in CS+ to automatically add the SIS module to the user project. For details, refer to the application note “RL78 Smart Configurator User's Guide: CS+ (R20AN0580)”.

2.12 “for”, “while” and “do while” Statements

This module uses “for”, “while” and “do while” statements (loop processing) for processing such as waiting for information to be reflected to registers. Such loop processing contains comments with “WAIT_LOOP” as a keyword. This allows users to search for relevant processing with “WAIT_LOOP” when adding Fail Safe processing to loop processing.

The following shows description examples.

```
Example of “for” statement:
/* WAIT_LOOP */
for (w_count = 0U; w_count <= BSP_CFG_FIHWAITTIME; w_count++)
{
    BSP_NOP();
}

Example of “do while” statement:
/* WAIT_LOOP */
do
{
    tmp_stab_wait  = OSTC;
    tmp_stab_wait &= tmp_stab_set;
}
while (tmp_stab_wait != tmp_stab_set);
```

3. API Functions

Prototype declarations of API functions are contained in r_nor_flash_rl78_if.h.

Notes on using API functions

1. When using this module, you must first call the R_NOR_FLASH_Open function.
2. Processes of all API functions are blocking processing. This also applies when DTC transfer is used.
3. Make sure that processing of the current API function terminates before calling the next API function.
4. After performing a write or erasure for the serial NOR flash memory, use the R_NOR_FLASH_CheckBusy function to confirm that the write or erasure is completed.
5. Reentrant functions are supported.

3.1 R_NOR_FLASH_Open()

This is the first function to be called when the serial NOR flash memory control software is used.

Format

```
e_nor_flash_status_t R_NOR_FLASH_Open(
void
)
```

Parameters

None

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_ALREADY_OPEN	This module is already open.

Description

Initializes the chip select pin. After this function is executed, the pin enters the port H output state.

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;

ret = R_NOR_FLASH_Open();
```

Special Notes

None

3.2 R_NOR_FLASH_Close()

This function is used to terminate the serial NOR flash memory control software being used.

Format

```
e_nor_flash_status_t R_NOR_FLASH_Close(
void
)
```

Parameters

None

Return Values

NOR_FLASH_SUCCESS	Normal end

Description

Initializes the chip select pin. After this function is executed, the pin enters the input port state.

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;

ret = R_NOR_FLASH_Close();
```

Special Notes

None

3.3 R_NOR_FLASH_ReadStatus()

This function is used to read Status Register 1 of the serial NOR flash memory.

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadStatus(
uint8_t * p_status
)
```

Parameters

`* p_status`

Status Register 1 storage buffer (Set 1 byte for the read buffer.)

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.

Description

Reads Status Register 1 of the serial NOR flash memory, and then stores the read data in p_status.

For details about Status Register 1, refer to the data sheet of the serial NOR flash memory used.

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint8_t stat = 0;

ret = R_NOR_FLASH_ReadStatus(&stat);
```

Special Notes

None

3.4 R_NOR_FLASH_ReadStatus2()

This function is used to read Status Register 2 of the serial NOR flash memory. This API function is dedicated to the AT25QF/AT25SF family serial NOR flash memory manufactured by Renesas Electronics.

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadStatus2(
uint8_t * p_status
)
```

Parameters

`* p_status`

Status Register 2 storage buffer (Set 1 byte for the read buffer.)

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.
NOR_FLASH_ERR_NON_SUPPORTED_API	Unsupported API

Description

Reads Status Register 2 of the serial NOR flash memory, and then stores the read data in p_status.

For details about Status Register 2, refer to the data sheet of the serial NOR flash memory used.

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint8_t stat = 0;

ret = R_NOR_FLASH_ReadStatus2(&stat);
```

Special Notes

None

3.5 R_NOR_FLASH_ReadStatus3()

This function is used to read Status Register 3 of the serial NOR flash memory. This API function is dedicated to the AT25QF/AT25SF family serial NOR flash memory manufactured by Renesas Electronics.

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadStatus3(
uint8_t * p_status
)
```

Parameters

`* p_status`

Status Register 3 storage buffer (Set 1 byte for the read buffer.)

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.
NOR_FLASH_ERR_NON_SUPPORTED_API	Unsupported API

Description

Reads Status Register 3 of the serial NOR flash memory, and then stores the read data in p_status.

For details about Status Register 3, refer to the data sheet of the serial NOR flash memory used.

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint8_t stat = 0;

ret = R_NOR_FLASH_ReadStatus3(&stat);
```

Special Notes

None

3.6 R_NOR_FLASH_WriteStatus()

This function is used to write to Status Register 1 of the serial NOR flash memory. This API function is dedicated to the AT25QF/AT25SF family serial NOR flash memory manufactured by Renesas Electronics.

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteStatus(
uint8_t * p_reg
)
```

Parameters

`* p_reg`

Status Register 1 setting data buffer (Set 1 byte for the setting data.)

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.
NOR_FLASH_ERR_WEL_CHK	Write Enable check error
NOR_FLASH_ERR_NON_SUPPORTED_API	Unsupported API

Description

The value set in p_reg is written to Status Register 1. However, depending on the serial NOR flash memory used, a function might be allocated or the value might be a reserved bit.

For details about Status Register 1, refer to the data sheet of the serial NOR flash memory used.

Before calling this user API function, read the value of Status Register 1, and then only change the bit values that must be changed. After processing terminates, read Status Register 1 and confirm that the correct value is written.

When this user API function terminates normally, the serial NOR flash memory transitions to the write state. You must use R_NOR_FLASH_CheckBusy() to confirm that the write is completed.

R_NOR_FLASH_CheckBusy() can be called at any timing specified by the user. Therefore, other processing of the user application can be performed while a write is in progress.

Example

```
#define FLASH_WR_BUSY_WAIT (uint32_t)(40) /* 40 * 1ms = 40ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint32_t loop_cnt = 0;
uint8_t gStat;
uint8_t Reg;

ret = R_NOR_FLASH_ReadStatus(&gStat]);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

Reg = (gStat | 0x10);
ret = R_NOR_FLASH_WriteStatus(&Reg);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

loop_cnt = FLASH_WR_BUSY_WAIT;
mode = NOR_FLASH_MODE_REG_WRITE_BUSY;
do
{
    /* FLASH is busy.
    User application can perform other processing while flash is busy. */

    ret = R_NOR_FLASH_CheckBusy(mode);
    if (NOR_FLASH_SUCCESS_BUSY != ret)
    {
        /* FLASH is ready or error. */
        break;
    }
    loop_cnt--;
    wait_timer(0, 1); /* 1ms */
}
while (0 != loop_cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
    /* Error */
}
```

Special Notes

None

3.7 R_NOR_FLASH_WriteStatus2()

This function is used to write to Status Register 2 of the serial NOR flash memory. This API function is dedicated to the AT25QF/AT25SF family serial NOR flash memory manufactured by Renesas Electronics.

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteStatus2(
uint8_t * p_reg
)
```

Parameters

`* p_reg`

Status Register 2 setting data buffer (Set 1 byte for the setting data.)

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.
NOR_FLASH_ERR_WEL_CHK	Write Enable check error
NOR_FLASH_ERR_NON_SUPPORTED_API	Unsupported API

Description

The value set in p_reg is written to Status Register 2. However, depending on the serial NOR flash memory used, a function might be allocated or the value might be a reserved bit.

For details about Status Register 2, refer to the data sheet of the serial NOR flash memory used.

Before calling this user API function, read the value of Status Register 2, and then only change the bit values that must be changed. After processing terminates, read Status Register 2 and confirm that the correct value is written.

When this user API function terminates normally, the serial NOR flash memory transitions to the write state. You must use R_NOR_FLASH_CheckBusy() to confirm that the write is completed.

R_NOR_FLASH_CheckBusy() can be called at any timing specified by the user. Therefore, other processing of the user application can be performed while a write is in progress.

Example

```
#define FLASH_WR_BUSY_WAIT (uint32_t)(40) /* 40 * 1ms = 40ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint32_t loop_cnt = 0;
uint8_t gStat;
uint8_t Reg;

ret = R_NOR_FLASH_ReadStatus2(&gStat]);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

Reg = (gStat | 0x10);
ret = R_NOR_FLASH_WriteStatus2(&Reg);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

loop_cnt = FLASH_WR_BUSY_WAIT;
mode = NOR_FLASH_MODE_REG_WRITE_BUSY;
do
{
    /* FLASH is busy.
    User application can perform other processing while flash is busy. */

    ret = R_NOR_FLASH_CheckBusy(mode);
    if (NOR_FLASH_SUCCESS_BUSY != ret)
    {
        /* FLASH is ready or error. */
        break;
    }
    loop_cnt--;
    wait_timer(0, 1); /* 1ms */
}
while (0 != loop_cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
/* Error */
}
```

Special Notes

None

3.8 R_NOR_FLASH_WriteStatus3()

This function is used to write to Status Register 3 of the serial NOR flash memory. This API function is dedicated to the AT25QF/AT25SF family serial NOR flash memory manufactured by Renesas Electronics.

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteStatus3(
uint8_t * p_reg
)
```

Parameters

`* p_reg`

Status Register 3 setting data buffer (Set 1 byte for the setting data.)

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.
NOR_FLASH_ERR_WEL_CHK	Write Enable check error
NOR_FLASH_ERR_NON_SUPPORTED_API	Unsupported API

Description

The value set in p_reg is written to Status Register 3. However, depending on the serial NOR flash memory used, a function might be allocated or the value might be a reserved bit.

For details about Status Register 3, refer to the data sheet of the serial NOR flash memory used.

Before calling this user API function, read the value of Status Register 3, and then only change the bit values that must be changed. After processing terminates, read Status Register 3 and confirm that the correct value is written.

When this user API function terminates normally, the serial NOR flash memory transitions to the write state. You must use R_NOR_FLASH_CheckBusy() to confirm that the write is completed.

R_NOR_FLASH_CheckBusy() can be called at any timing specified by the user. Therefore, other processing of the user application can be performed while a write is in progress.

Example

```
#define FLASH_WR_BUSY_WAIT (uint32_t)(40) /* 40 * 1ms = 40ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint32_t loop_cnt = 0;
uint8_t gStat;
uint8_t Reg;

ret = R_NOR_FLASH_ReadStatus3(&gStat]);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

Reg = (gStat | 0x10);
ret = R_NOR_FLASH_WriteStatus3(&Reg);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

loop_cnt = FLASH_WR_BUSY_WAIT;
mode = NOR_FLASH_MODE_REG_WRITE_BUSY;
do
{
    /* FLASH is busy.
    User application can perform other processing while flash is busy. */

    ret = R_NOR_FLASH_CheckBusy(mode);
    if (NOR_FLASH_SUCCESS_BUSY != ret)
    {
        /* FLASH is ready or error. */
        break;
    }
    loop_cnt--;
    wait_timer(0, 1); /* 1ms */
}
while (0 != loop_cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
    /* Error */
}
```

Special Notes

None

3.9 R_NOR_FLASH_SetWriteProtect()

This function is used to set the write protect for the serial NOR flash memory.

Format

```
e_nor_flash_status_t R_NOR_FLASH_SetWriteProtect(
uint8_t wpsts
)
```

Parameters

`wpsts`

Write protect setting data

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.
NOR_FLASH_ERR_WEL_CHK	Write Enable check error

Description

Sets the write protect.

For the serial NOR flash memory manufactured by Macronix International Co., Ltd, set 0 to the SRWD bit.

For the serial NOR flash memory manufactured by Renesas Electronics, set 0 to the SRP0 bit.

Specify the write protect setting data (wpsts) as shown below.

For details about the relationship between the protect area and protect bits, refer to the data sheet of the serial NOR flash memory used.

<Renesas Electronics AT25QF family serial NOR flash memory>

Specify the block sizes (large and small), top, and bottom during write processing of Status Register 1.

Specify the complement setting during write processing of Status Register 2.

wpsts	BP2	BP1	BP0
0x00	0	0	0
0x01	0	0	1
0x02	0	1	0
0x03	0	1	1
0x04	1	0	0
0x05	1	0	1
0x06	1	1	0
0x07	1	1	1

<MX25L/MX66L/MX25R family serial NOR flash memory manufactured by Macronix International Co., Ltd>

Specify the Top/Bottom setting during Configuration Register write processing.

wpsts	BP3	BP2	BP1	BP0
0x00	0	0	0	0
0x01	0	0	0	1
0x02	0	0	1	0
0x03	0	0	1	1
0x04	0	1	0	0
0x05	0	1	0	1
0x06	0	1	1	0
0x07	0	1	1	1
0x08	1	0	0	0
0x09	1	0	0	1
0x0a	1	0	1	0
0x0b	1	0	1	1
0x0c	1	1	0	0
0x0d	1	1	0	1
0x0e	1	1	1	0
0x0f	1	1	1	1

When this user API function terminates normally, the serial NOR flash memory transitions to the write state. You must use R_NOR_FLASH_CheckBusy() to confirm that the write is completed.

R_NOR_FLASH_CheckBusy() can be called at any timing specified by the user. Therefore, other processing of the user application can be performed while a write is in progress.

For details, refer to Figure 3-1.

Example

```
#define FLASH_WR_BUSY_WAIT (uint32_t)(40) /* 40 * 1ms = 40ms */

e_nor_flash_status_t    ret = NOR_FLASH_SUCCESS;
uint32_t                loop_cnt = 0;
e_nor_flash_check_busy_t      mode;

ret = R_NOR_FLASH_SetWriteProtect(0);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

loop_cnt = FLASH_WR_BUSY_WAIT;
mode = NOR_FLASH_MODE_REG_WRITE_BUSY;
do
{
    /* FLASH is busy.
        User application can perform other processing while flash is busy. */
    ret = R_NOR_FLASH_CheckBusy(mode);
    if (NOR_FLASH_SUCCESS_BUSY != ret)
    {
        /* FLASH is ready or error. */
        break;
    }
    loop_cnt--;
    wait_timer(0, 1); /* 1ms */
}
while (0 != loop_cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
    /* Error */
}
```

Special Notes

None

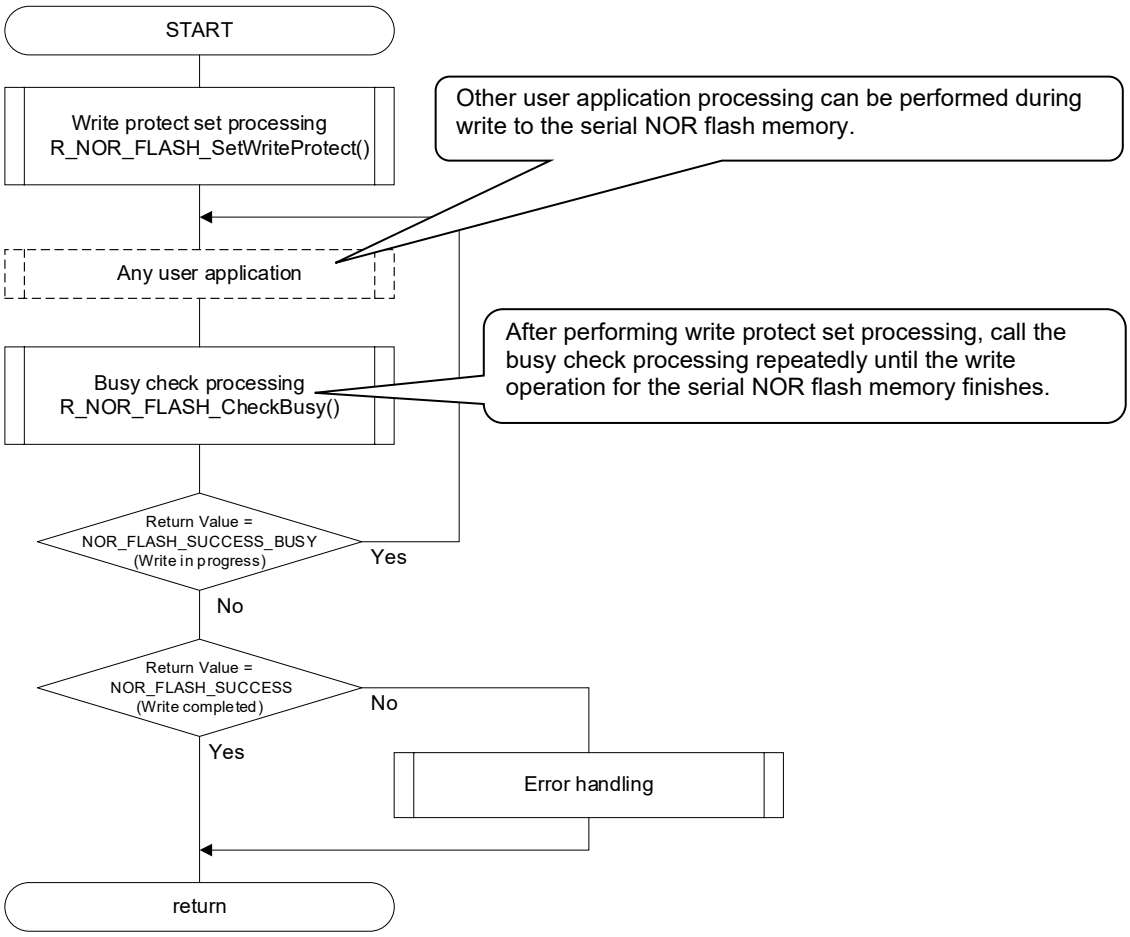


Figure 3-1 R_NOR_FLASH_SetWriteProtect() Processing Example

3.10 R_NOR_FLASH_WriteDisable()

This function is used to disable a write to the serial NOR flash memory.

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteDisable(
void
)
```

Parameters

None

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.

Description

Sends the Write Disable command to clear the WEL bit of Status Registers of the serial NOR flash memory.

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;

ret = R_NOR_FLASH_WriteDisable();
```

Special Notes

None

3.11 R_NOR_FLASH_ReadData()

This function is used to read data from the serial NOR flash memory.

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadData(
st_nor_flash_info_t * p_nor_flash_info
)
```

Parameters

`* p_nor_flash_info`

Serial NOR flash memory information structure

`addr`

Specify the read start address of the memory.

`cnt`

Specify the number of read bytes. The specifiable range is from 1 to 65,535.

If you specify 0, an error will be returned.

`data_cnt`

Number of read bytes (User setting is prohibited because this parameter is used by this API function.)

`*p_data`

Specify the address of the read data storage buffer.

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.

Description

Reads the specified number of bytes of data from the specified address in the serial NOR flash memory, and then stores the read data in p_data.

The last address of the serial NOR flash memory is the serial NOR Flash memory capacity minus 1.

A read exceeding the last address cannot be specified. In this case, terminate processing after the last address is read. Then, specify the address again and call the user API function.

Make sure that the sum of the number of read bytes (specified in cnt) and the address (specified in addr) does not exceed the last address.

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
st_nor_flash_info_t Flash_Info_R;
uint8_t buf2[128];

Flash_Info_R.addr = 0;
Flash_Info_R.cnt = 32;
Flash_Info_R.p_data = &buf2[0];
ret = R_NOR_FLASH_ReadData(&Flash_Info_R);
```

Special Notes

None

3.12 R_NOR_FLASH_WriteDataPage()

This function is used to write data to the serial NOR flash memory in units of one page (up to 256 bytes).

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteDataPage(
st_nor_flash_info_t * p_nor_flash_info
)
```

Parameters

`* p_nor_flash_info`

Serial NOR flash memory information structure

`addr`

Specify the write start address of the memory.

`cnt`

Specify the number of write bytes. The specifiable range is from 1 to 65,535.

If you specify 0, an error will be returned.

`data_cnt`

Number of write bytes (User setting is prohibited because this parameter is used by this API function.)

`*p_data`

Specify the address of the write data storage buffer.

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.
NOR_FLASH_ERR_WEL_CHK	Write Enable check error

Description

Writes the specified number of bytes of data (up to the size of one page (256 bytes)) from p_data to the serial NOR flash memory, starting from the specified address of the memory.

If the number of bytes exceeding one page is specified, the number of remaining bytes and next address information remain in the information structure p_nor_flash_info of the serial NOR flash memory after completion of one-page write processing. It is possible to write data for the remaining number of bytes by setting that p_nor_flash_info in this user API function without change.

The maximum value that can be specified for the number of write bytes (cnt) is the capacity of the serial NOR flash memory.

Make sure that the sum of the number of write bytes (specified in cnt) and the address (specified in addr) does not exceed the last address of the serial NOR flash memory.

Example

```
#define FLASH_PP_BUSY_WAIT (uint32_t)(3) /* 3 * 1ms = 3ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
st_nor_flash_info_t Flash_Info_W;
uint8_t buf1[512];
uint32_t loop_cnt = 0;

Flash_Info_W.addr = 0;
Flash_Info_W.cnt = 512;
Flash_Info_W.p_data = &buf1[0];

do
{
    ret = R_NOR_FLASH_WriteDataPage(&Flash_Info_W);
    if (NOR_FLASH_SUCCESS > ret)
    {
        /* Error */
    }

    loop_cnt = FLASH_PP_BUSY_WAIT;
    mode = NOR_FLASH_MODE_PROG_BUSY;
    do
    {
        /* FLASH is busy.
        User application can perform other processing while flash is busy. */
        ret = R_NOR_FLASH_CheckBusy(mode);
        if (NOR_FLASH_SUCCESS_BUSY != ret)
        {
            /* FLASH is ready or error. */
            break;
        }
        loop_cnt--;
        wait_timer(0, 1); /* 1ms */
    }
    while (0 != loop_cnt);
}
while (0 != Flash_Info_W.cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
/* Error */
}
```

Special Notes

Writing to the serial NOR flash memory is possible only when the write protect is canceled.

When this user API function terminates normally, the serial NOR flash memory transitions to the write state. You must use R_NOR_FLASH_CheckBusy() to confirm that the write is completed.

R_NOR_FLASH_CheckBusy() can be called at any timing specified by the user. Therefore, other processing of the user application can be performed while a write is in progress.

For details, refer to Figure 3-2.

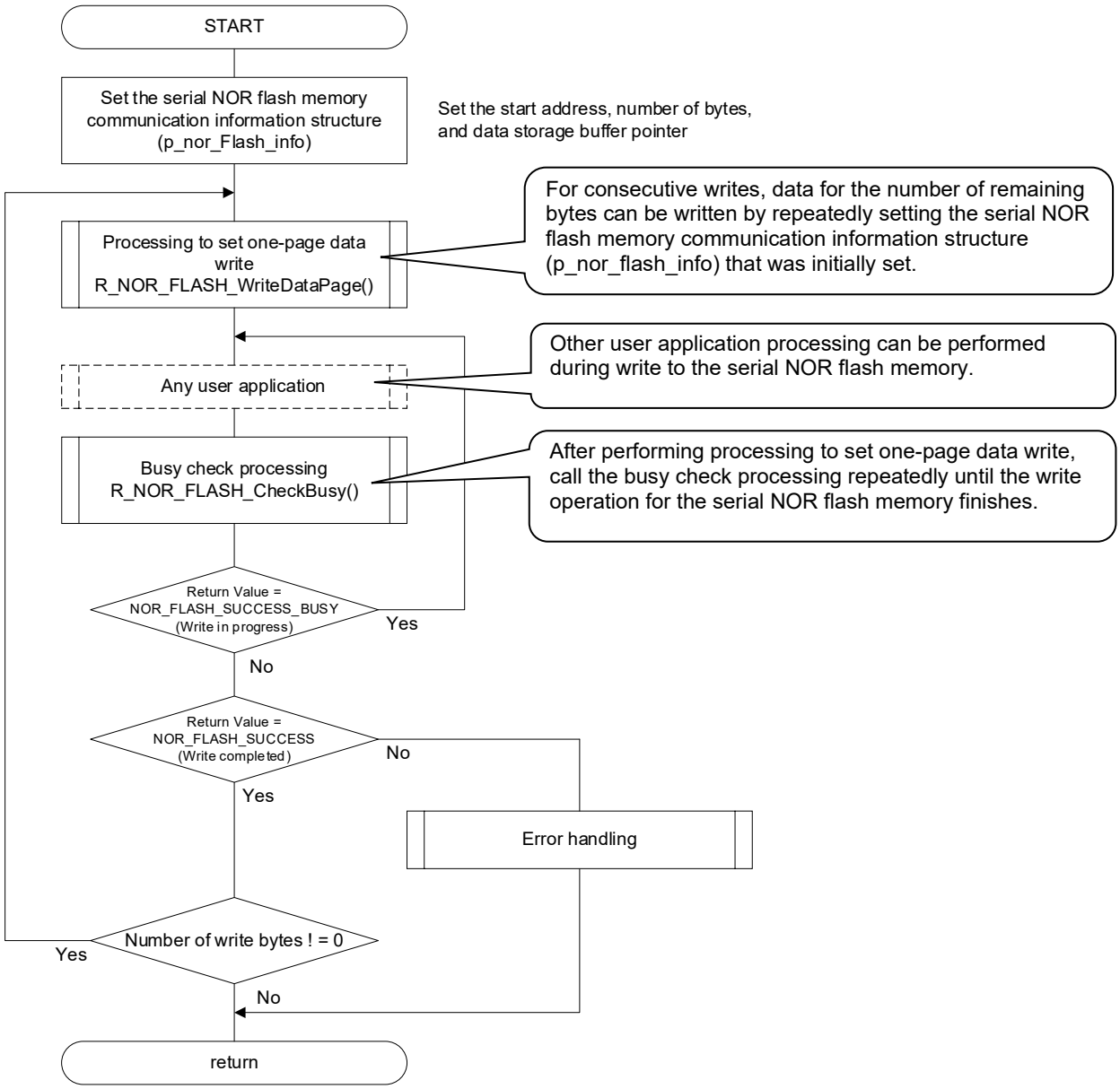


Figure 3-2 R_NOR_FLASH_SetWriteProtect() Processing Example

3.13 R_NOR_FLASH_Erase()

This function is used to erase the specified memory for the serial NOR flash memory. The target to be erased can be specified by the mode parameter.

Format

```
e_nor_flash_status_t R_NOR_FLASH_Erase(
st_nor_flash_erase_info_t * p_nor_flash_erase_info
)
```

Parameters

`* p_nor_flash_erase_info`

Serial NOR flash memory Erase information structure

`addr`

Specify the write start address of the memory.

`mode`

Erasure mode setting

Select one of the following:

<AT25QF/AT25SF family serial NOR flash memory manufactured by Renesas Electronics>

NOR_FLASH_MODE_C_ERASE	/* Erase all data on the chip (Chip Erase) */
NOR_FLASH_MODE_BK4_ERASE	/* Erase all data in the block (Block Erase: 4 KB) */
NOR_FLASH_MODE_B32K_ERASE	/* Erase all data in the block (Block Erase: 32 KB) */
NOR_FLASH_MODE_B64K_ERASE	/* Erase all data in the block (Block Erase: 64 KB) */
NOR_FLASH_MODE_SCUR_ERASE	/* Erase the Security Register pages */

<MX25L/MX66L/MX25R family serial NOR flash memory manufactured by Macronix International Co., Ltd>

NOR_FLASH_MODE_C_ERASE	/* Erase all data on the chip (Chip Erase) */
NOR_FLASH_MODE_S_ERASE	/* Erase all data in the sector (Sector Erase: 4 KB) */
NOR_FLASH_MODE_B32K_ERASE	/* Erase all data in the block (Block Erase: 32 KB) */
NOR_FLASH_MODE_B64K_ERASE	/* Erase all data in the block (Block Erase: 64 KB) */

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.
NOR_FLASH_ERR_WEL_CHK	Write Enable check error

Description

For Sector Erase, specify the start address of the sector for addr.

For Block Erase, specify the start address of the block for addr.

For security erase, specify the start address of the Security Register page in addr.

For Chip Erase, specify 0x00000000 for addr.

Erasing data in the serial NOR flash memory is possible only when the write protect is canceled.

When this user API function terminates normally, the serial NOR flash memory transitions to the erasure state. You must use R_NOR_FLASH_CheckBusy() to confirm that the erasure is completed.

R_NOR_FLASH_CheckBusy() can be called at any timing specified by the user. Therefore, other processing of the user application can be performed while erasure is in progress.

For details, refer to Figure 3-3.

Example

```
#define FLASH_SE_BUSY_WAIT (uint32_t)(200) /* 200 * 1ms = 200ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
st_nor_flash_erase_info_t Flash_Info_E;
uint32_t loop_cnt = 0;

Flash_Info_E.addr = 0;
Flash_Info_E.mode = NOR_FLASH_MODE_S_ERASE;

ret = R_NOR_FLASH_Erase(&Flash_Info_E);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

loop_cnt = FLASH_SE_BUSY_WAIT;
mode = NOR_FLASH_MODE_ERASE_BUSY;
do
{
    /* FLASH is busy.
    User application can perform other processing while flash is busy. */

    ret = R_NOR_FLASH_CheckBusy(mode);
    if (NOR_FLASH_SUCCESS_BUSY != ret)
    {
        /* FLASH is ready or error. */
        break;
    }
    loop_cnt--;
    wait_timer(0, 1); /* 1ms */
}
while (0 != loop_cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
    /* Error */
}
```

Special Notes

None

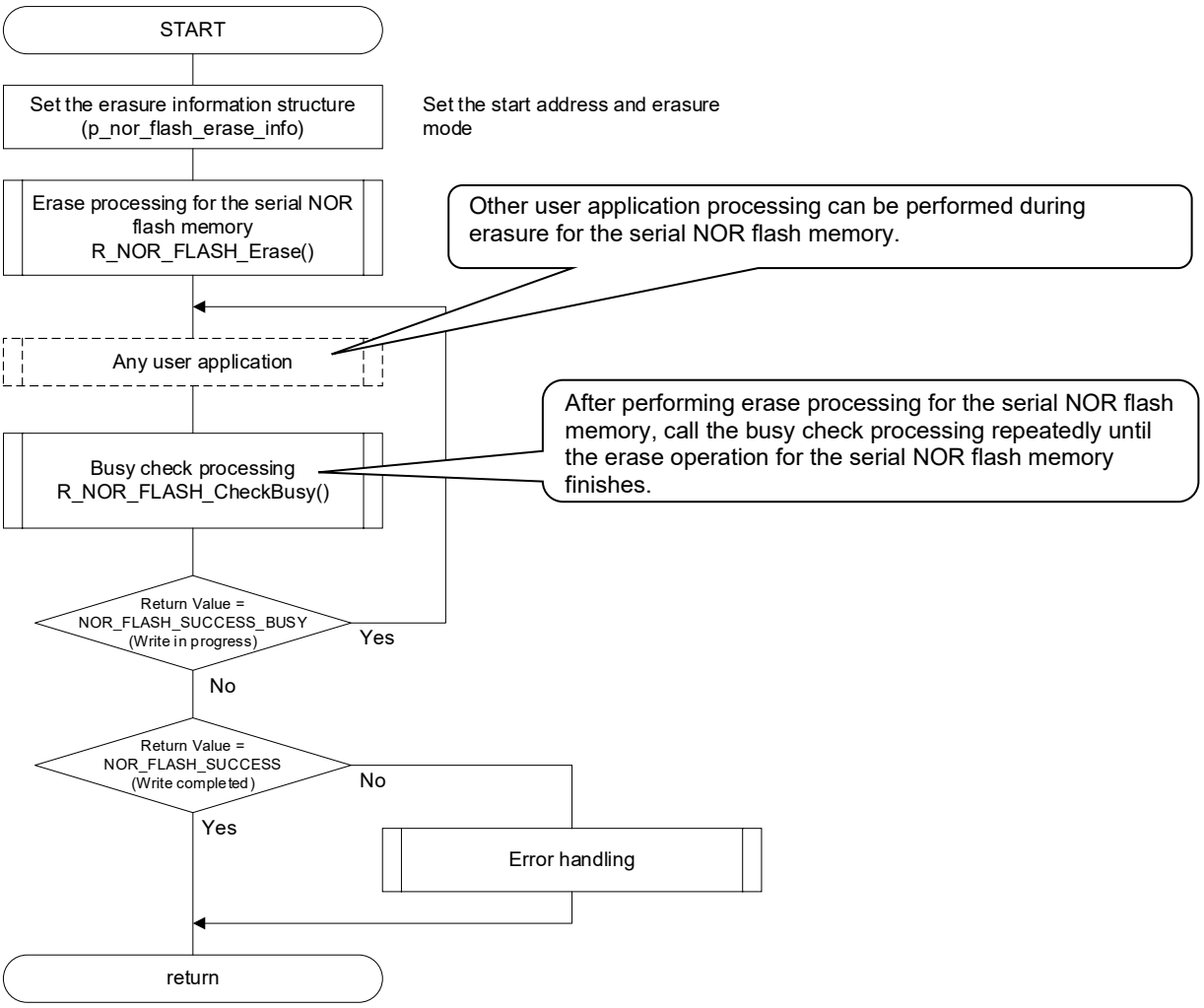


Figure 3-3 R_NOR_FLASH_Erase() Processing Example

3.14 R_NOR_FLASH_CheckBusy()

This function is used to confirm that a write to or erasure for the serial NOR flash memory is completed.

Format

```
e_nor_flash_status_t R_NOR_FLASH_CheckBusy(
e_nor_flash_check_busy_t mode
)
```

Parameters

`mode`

Completion confirmation setting

<AT25QF/AT25SF family serial NOR flash memory manufactured by Renesas Electronics>

NOR_FLASH_MODE_REG_WRITE_BUSY /* Confirm that register write completed */

<MX25L/MX66L/MX25R family serial NOR flash memory manufactured by Macronix International Co., Ltd>

Select one of the following:

NOR_FLASH_MODE_REG_WRITE_BUSY /* Confirm that register write completed */

NOR_FLASH_MODE_PROG_BUSY /* Confirm that data write completed */

NOR_FLASH_MODE_ERASE_BUSY /* Confirm that erasure completed */

Return Values

NOR_FLASH_SUCCESS	Normal end and write completed
NOR_FLASH_SUCCESS_BUSY	Normal end and write in progress
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.

Description

Determines whether the write or erase operation has finished.

Example

Refer to Figure 3-1 or Figure 3-2.

Special Notes

R_NOR_FLASH_CheckBusy() can be called at any timing specified by the user. Therefore, other processing of the user application can be performed while a write is in progress.

3.15 R_NOR_FLASH_ReadId()

This function is used to read ID information of the serial NOR flash memory.

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadId(
uint8_t * p_data
)
```

Parameters

`* p_data`

ID information storage buffer

The manufacture ID and device ID will be read. Set 3 bytes for the read buffer.

(1) Manufacturer ID (1 byte)

(2) Device ID (2 byes)

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.

Description

Stores ID information of the serial NOR flash memory in p_data.

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint8_t gID[4];

ret = R_NOR_FLASH_ReadId(&gID[0]);
```

Special Notes

None

3.16 R_NOR_FLASH_GetMemoryInfo()

This function returns the size information of the serial NOR flash memory as a return value.

Format

```
e_nor_flash_status_t R_NOR_FLASH_GetMemoryInfo(
st_nor_flash_mem_info_t * p_nor_flash_mem_info
)
```

Parameters

`* p_nor_flash_mem_info`

Serial NOR flash memory size information structure

`mem_size`

Maximum memory size

`wpag_size`

Page size

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error

Description

Acquires size information of the serial NOR flash memory.

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
st_nor_flash_mem_info_t Flash_MemInfo;

ret = R_NOR_FLASH_GetMemoryInfo(&Flash_MemInfo);
```

Special Notes

This function returns the size information of the specified serial NOR flash memory as a return value. This function returns fixed values defined by this API function without performing communication with the serial NOR flash memory.

3.17 R_NOR_FLASH_ReadConfiguration()

This function is used to read the Configuration Register of the serial NOR flash memory. This API function is dedicated to the MX25L/MX66L/MX25R family serial NOR flash memory manufactured by Macronix International Co., Ltd.

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadConfiguration(
uint8_t * p_config
)
```

Parameters

`* p_config`

Configuration Register storage buffer. The size differs depending on the serial NOR flash memory used. See the following and reserve the read buffer.

<MX25L/MX66L family serial NOR flash memory manufactured by Macronix International Co., Ltd>

Read the Configuration Register. Set 1 byte for the read buffer.

<MX25R family serial NOR flash memory manufactured by Macronix International Co., Ltd>

Read Configuration Register-1 and Configuration Register-2. Set 2 bytes for the read buffer.

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.
NOR_FLASH_ERR_NON_SUPPORTED_API	Unsupported API

Description

Reads the Configuration Register of the serial NOR flash memory, and then stores the read data in p_config.

For details about Configuration Registers, refer to the data sheet of the serial NOR flash memory used.

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint8_t gConfig[2];

ret = R_NOR_FLASH_ReadConfiguration(&gConfig[0]);
```

Special Notes

None

3.18 R_NOR_FLASH_WriteConfiguration()

This function is used to write to Configuration Registers of the serial NOR flash memory. This API function is dedicated to the MX25L/MX66L/MX25R family serial NOR flash memory manufactured by Macronix International Co., Ltd.

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteConfiguration(
st_nor_flash_reg_info_t * p_reg
)
```

Parameters

`* p_reg`

Register information structure

`status`

Status Register (User setting is prohibited because this parameter is used by this API function.)

`config1`

Configuration Register setting data

`config2`

Configuration Register-2 setting data

Note that the configuration of the structure differs depending on the serial NOR flash memory used. See the following and then specify the values. For details about set values, refer to “Description”.

<MX25L/MX66L family serial NOR flash memory manufactured by Macronix International Co., Ltd>

The value set in p_reg->config1 is written to the Configuration Register.

Setting of p_reg->config2 is invalid.

<MX25R family serial NOR flash memory manufactured by Macronix International Co., Ltd>

The value set in p_reg->config1 is written to Configuration Register-1.

The value set in p_reg->config2 is written to Configuration Register-2.

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.
NOR_FLASH_ERR_WEL_CHK	Write Enable check error
NOR_FLASH_ERR_NON_SUPPORTED_API	Unsupported API

Description

Writes the values set in p_reg->config1 and p_reg->config2 to Configuration Registers.

For details about Configuration Registers, refer to the data sheet of the serial NOR flash memory used.

When calling this user API function, read the values of Configuration Registers in advance. Then, only change the bit values you want to rewrite, and then set the new values in p_reg->config1 and p_reg->config2.

After processing terminates, read the Configuration Register to confirm the write values.

Setting of 4-byte bits is ignored because they are read-only bits.

When this user API function terminates normally, the serial NOR flash memory transitions to the write state. You must use R_NOR_FLASH_CheckBusy() to confirm that the write is completed.

R_NOR_FLASH_CheckBusy() can be called at any timing specified by the user. Therefore, other processing of the user application can be performed while a write is in progress.

For details, refer to Figure 3-4.

Example

```
#define FLASH_WR_BUSY_WAIT (uint32_t)(40) /* 40 * 1ms = 40ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint32_t loop_cnt = 0;
st_nor_flash_reg_info_t Reg;
uint8_t gConfig[2];

ret = R_NOR_FLASH_ReadConfiguration(&gConfig[0]);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

Reg.config1 = (gConfig[0] | 0x10); /* Set Preamble bit Enable */
ret = R_NOR_FLASH_WriteConfiguration(&Reg);
if (NOR_FLASH_SUCCESS > ret)
{
    /* Error */
}

loop_cnt = FLASH_WR_BUSY_WAIT;
mode = NOR_FLASH_MODE_REG_WRITE_BUSY;
do
{
    /* FLASH is busy.
    User application can perform other processing while flash is busy. */

    ret = R_NOR_FLASH_CheckBusy(mode);
    if (NOR_FLASH_SUCCESS_BUSY != ret)
    {
    /* FLASH is ready or error. */
        break;
    }
    loop_cnt--;
    wait_timer(0, 1); /* 1ms */
}
while (0 != loop_cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
    /* Error */
}
```

Special Notes

None

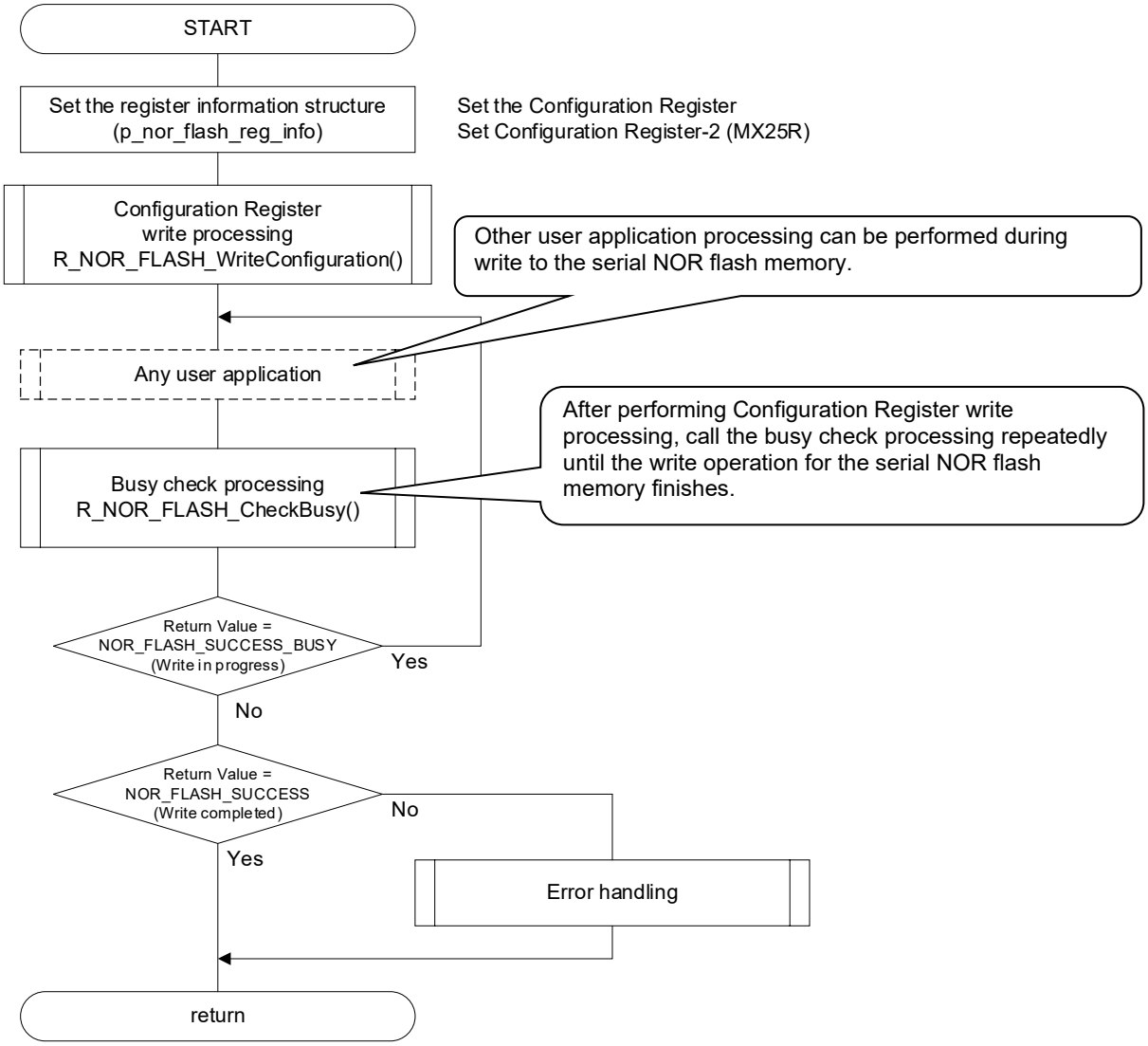


Figure 3-4 R_NOR_FLASH_WriteConfiguration() Processing Example

3.19 R_NOR_FLASH_Set4byteAddressMode()

This function is used to set 4-byte address mode for the serial NOR flash memory. This API function is dedicated to the MX25L/MX66L/MX25R family serial NOR flash memory manufactured by Macronix International Co., Ltd.

Format

```
e_nor_flash_status_t R_NOR_FLASH_Set4byteAddressMode(
void
)
```

Parameters

None

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.
NOR_FLASH_ERR_NON_SUPPORTED_API	Unsupported API

Description

<MX25L/MX66L/MX25R family serial NOR flash memory manufactured by Macronix International Co., Ltd>

Issues the EN4B (0xb7) command to set 1 to the 4BYTE bit of the Configuration Register.

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;

ret = R_NOR_FLASH_Set4byteAddressMode();
```

Special Notes

None

3.20 R_NOR_FLASH_ReadSecurity()

This function is used to read the Security Register of the serial NOR flash memory. This API function is dedicated to the MX25L/MX66L/MX25R family serial NOR flash memory manufactured by Macronix International Co., Ltd.

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadSecurity(
uint8_t * p_scur
)
```

Parameters

`* p_scur`

Security Register storage buffer (Set 1 byte for the read buffer.)

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.

Description

Reads the Security Register, and then stores the read data in p_scur.

For details about the Security Register, refer to the data sheet of the serial NOR flash memory used.

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
uint8_t scur = 0;

ret = R_NOR_FLASH_ReadSecurity(&scur);
```

Special Notes

None

3.21 R_NOR_FLASH_ReadDataSecurityPage()

This function is used to read data from the Security Register of the serial NOR flash memory. This API function is dedicated to the AT25QF/AT25SF family serial NOR flash memory manufactured by Renesas Electronics.

Format

```
e_nor_flash_status_t R_NOR_FLASH_ReadDataSecurityPage(
st_nor_flash_info_t * p_nor_flash_info
)
```

Parameters

`* p_nor_flash_info`

Serial NOR flash memory information structure

`addr`

Specify the read start address of the memory.

`cnt`

Specify the number of read bytes. The specifiable range is from 1 to 256.

If you specify 0, an error will be returned.

`data_cnt`

Number of read bytes (Setting is prohibited because this parameter is used by this control software.)

`*p_data`

Specify the address of the read data storage buffer.

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.
NOR_FLASH_ERR_NON_SUPPORTED_API	Unsupported API

Description

Reads the specified number of bytes of data from the specified address in the Security Register, and then stores the read data in p_data.

The last address of the Security Register is the page size minus 1.

A read exceeding the last address cannot be specified. In this case, terminate processing after the last address is read. Then, specify the address again and call the user API function.

Make sure that the sum of the number of read bytes (specified in cnt) and the address (specified in addr) does not exceed the last address.

Example

```
e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
st_nor_flash_info_t Flash_Info_R;
uint8_t buf2[128];

Flash_Info_R.addr = 0x1000;/* Security Register 1 Address */
Flash_Info_R.cnt = 32;
Flash_Info_R.p_data = &buf2[0];
ret = R_NOR_FLASH_ReadDataSecurityPage(&Flash_Info_R);
```

Special Notes

None

3.22 R_NOR_FLASH_WriteDataSecurityPage()

This function is used to write data in units of one page to Security Register pages of the serial NOR flash memory. This API function is dedicated to the AT25QF/AT25SF family serial NOR flash memory manufactured by Renesas Electronics.

Format

```
e_nor_flash_status_t R_NOR_FLASH_WriteDataSecurityPage(
st_nor_flash_info_t * p_nor_flash_info
)
```

Parameters

`* p_nor_flash_info`

Serial NOR flash memory information structure

`addr`

Specify the write start address of the memory.

`cnt`

Specify the number of write bytes. The specifiable range is from 1 to 65,535.

If you specify 0, an error will be returned.

`data_cnt`

Number of write bytes (Setting is prohibited because this parameter is used by this control software.)

`*p_data`

Specify the address of the write data storage buffer.

Return Values

NOR_FLASH_SUCCESS	Normal end
NOR_FLASH_ERR_PARAM	Parameter error
NOR_FLASH_ERR_HARD	Hardware error
NOR_FLASH_ERR_NOT_OPEN	R_NOR_FLASH_Open function of this module is not executed.
NOR_FLASH_ERR_WEL_CHK	Write Enable check error
NOR_FLASH_ERR_NON_SUPPORTED_API	Unsupported API

Description

Writes the specified number of bytes of data (up to one page as the maximum size) from p_data to the Security Register page, starting from the specified address.

When writing a large volume of data, communication is divided into page units. This prevents other processing from being disabled while communication is in progress.

Writing to the Security Register page is possible only when the page is not locked.

The maximum value that can be specified for the number of write bytes (cnt) is the page size of the Security Register.

Make sure that the sum of the number of write bytes (specified in cnt) and the address (specified in addr) does not exceed the last address of the Security Register.

Example

```
#define FLASH_PP_BUSY_WAIT (uint32_t)(3) /* 3 * 1ms = 3ms */

e_nor_flash_status_t ret = NOR_FLASH_SUCCESS;
st_nor_flash_info_t Flash_Info_W;
uint8_t buf1[128];
uint32_t loop_cnt = 0;

Flash_Info_W.addr = 0;
Flash_Info_W.cnt = 128;
Flash_Info_W.p_data = &buf1[0];

do
{
    ret = R_NOR_FLASH_WriteDataSecurityPage(&Flash_Info_W);
    if (NOR_FLASH_SUCCESS > ret)
    {
        /* Error */
    }

    loop_cnt = FLASH_PP_BUSY_WAIT;
    mode = NOR_FLASH_MODE_PROG_BUSY;
    do
    {
        /* FLASH is busy.
        User application can perform other processing while flash is busy. */

        ret = R_NOR_FLASH_CheckBusy(mode);
        if (NOR_FLASH_SUCCESS_BUSY != ret)
        {
            /* FLASH is ready or error. */
            break;
        }
        loop_cnt--;
        wait_timer(0, 1); /* 1ms */
    }
    while (0 != loop_cnt);
}
while (0 != Flash_Info_W.cnt);

if ((0 == loop_cnt) || (NOR_FLASH_SUCCESS > ret))
{
    /* Error */
}
```

Special Notes

None

3.23 R_NOR_FLASH_GetVersion()

This function is used to acquire the version information of the serial NOR flash memory control software.

Format

```
uint32_t R_NOR_FLASH_GetVersion(
void
)
```

Parameters

None

Return Values

Upper 2 bytes	Major version (decimal)
Lower 2 bytes	Minor version (decimal)

Description

Returns version information.

Example

```
uint32_t version;
version = R_NOR_FLASH_GetVersion();
```

Special Notes

None

3.24 R_NOR_FLASH_Interval()

This function is used to detect a timeout when, for example, data transmission does not terminate during DTC transfer.

When using the DTC, call this function every 1 ms by using a timer.

Format

```
void R_NOR_FLASH_Interval(
void
)
```

Parameters

None

Return Values

None

Description

Increments the internal timer counter during wait for the completion of DTC transfer.

Example

```
static void __near r_Config_TAU0_1_higher8bits_interrupt(void)
{
    /* Start user code for r_Config_TAU0_1_higher8bits_interrupt. Do not edit comment generated here
*/
    R_NOR_FLASH_Interval();
    /* End user code. Do not edit comment generated here */
}
```

Special Notes

Call this function every 1 ms by using a timer.

The preceding example shows that this function is called by interrupt processing that is generated every 1 ms.

Unlike other API functions, this function is not called from the main function.

3.25 R_NOR_FLASH_SendendNotification()

This function performs transmission completion notification processing for SPI communication. Call this function from the transmission completion callback function for SPI (CSI) communication.

Format

```
void R_NOR_FLASH_SendendNotification(
void
)
```

Parameters

None

Return Values

None

Description

Clears the communication-in-progress flag for SPI communication.

Example

```
/* Start user code for include. Do not edit comment generated here */
#include "r_nor_flash_rl78_if.h"
/* End user code. Do not edit comment generated here */


void r_Config_CSI11_callback_sendend(void)
{
    /* Start user code for r_Config_CSI11_callback_sendend. Do not edit comment generated here */
    R_NOR_FLASH_SendendNotification();
    /* End user code. Do not edit comment generated here */
}
```

Special Notes

Call this function from the transmission completion callback function for SPI (CSI) communication used by this module.

The preceding example shows that this function is called by the generated transmission completion callback function for SPI (CSI) communication.

Unlike other API functions, this function is not called from the main function.

4. Pin Settings

This module uses the CS pin.

The CS pin changes according to the Open and Close functions as follows.

Table 4-1 Pin Settings for Controlling Serial NOR Flash Memory

Function Name	Chip Select Pin*1
Before R_NOR_FLASH_Open()	Depends on the user setting
After R_NOR_FLASH_Open()	H output state
After R_NOR_FLASH_Close()	Input state

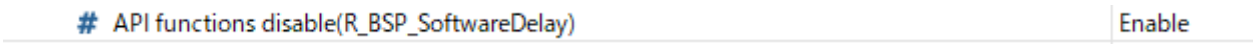
Note: 1. Use external resistors to perform pull-up processing for the chip select pin.

5. Various Settings after Adding This Module

After adding this module to your project, you must add and configure components and specify configuration options according to the environment used.

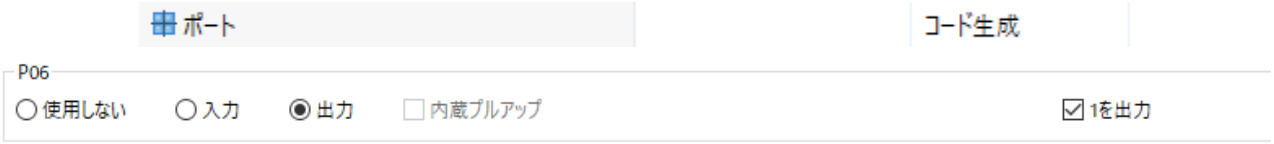
(1) Enabling the function to be used for wait

Specify Enable for R_BSP_SoftwareDelay used for wait by this module.



(2) Port used for the CS# pin

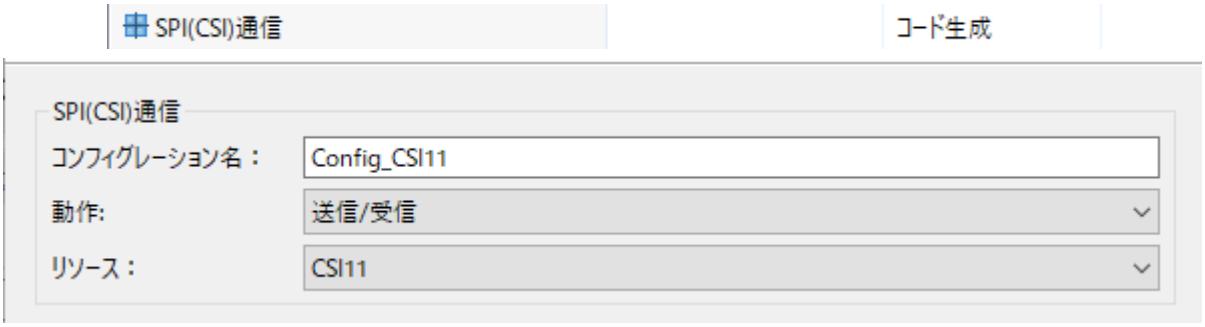
Add a code generation port and specify H output for the port used by this module to control the CS# pin.



Select the port used by this module to control the CS# pin in **CS Port Number** and **CS Bit Number**.

(3) Channel used for SPI communication

In the SPI (CSI) communication setting for code generation, add the channel used for SPI communication.



Call the following callback function of this module within the transmission completion callback of the generated SPI (CSI) communication code.

void R_NOR_FLASH_SendendNotification(void);

Example

```
/* Start user code for include. Do not edit comment generated here */
#include "r_nor_flash_rl78_if.h"
/* End user code. Do not edit comment generated here */


void r_Config_CSI11_callback_sendend(void)
{
    /* Start user code for r_Config_CSI11_callback_sendend. Do not edit comment generated here */
    R_NOR_FLASH_SendendNotification();
    /* End user code. Do not edit comment generated here */
}
```

Select the channel used by this module for SPI communication in **SPI(CSI) channel number**.

(4) When using DTC transfer

Add the Data Transfer Controller for code generation, and then specify the settings according to the channel used for SPI communication.

Select **DTC transfer** for **Data transfer mode**.

# CS Port Number	PORT0
# CS Bit Number	BIT6
# Data transfer mode	DTC transfer
# SPI(CSI) channel number	Channel 3
# DTC Control Data Number	0

Enter the control data number in **DTC Control Data Number**.

# CS Port Number	PORT0
# CS Bit Number	BIT6
# Data transfer mode	DTC transfer
# SPI(CSI) channel number	Channel 3
# DTC Control Data Number	0

6. Demo Projects

6.1 Overview

This application note includes demo projects for describing how to use the serial NOR flash memory control module.

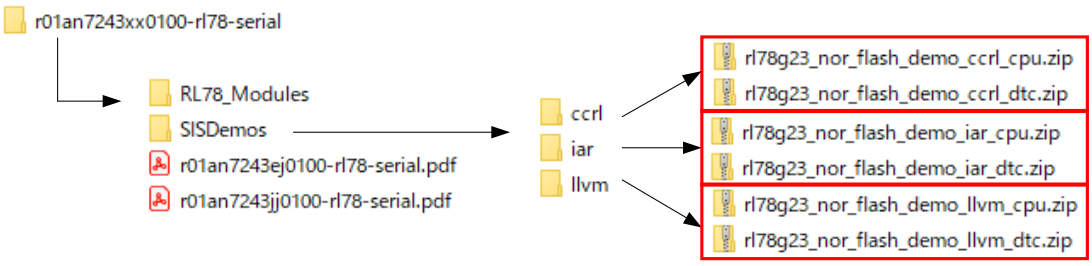


Table 6-1 List of Demo Projects

Compiler	Demo Project Name	Transfer Method
Renesas Compiler [CC-RL]	rl78g23_nor_flash_demo_ccrl_cpu	CPU transfer
	rl78g23_nor_flash_demo_ccrl_dtc	DTC transfer
LLVM	rl78g23_nor_flash_demo_llvm_cpu	CPU transfer
	rl78g23_nor_flash_demo_llvm_dtc	DTC transfer
IAR	rl78g23_nor_flash_demo_iar_cpu	CPU transfer
	rl78g23_nor_flash_demo_iar_dtc	DTC transfer

The demo projects perform the following processing in sequence:

- OPEN processing of this module and a busy check of the serial NOR flash memory
- Clearing write protect
- Erasure, write, and read processing of the serial NOR flash memory
- Register control for the serial NOR flash memory
- Acquisition of serial NOR flash memory information
- Acquisition of version information of this module
- CLOSE processing of this module

6.2 Operation Confirmation Environment

Table 6-2 shows the conditions, including hardware and settings, required for operating the demo projects.

Figure 6-1 shows the hardware configuration including the evaluation board and serial NOR flash memory.

Figure 6-2 shows the wire connection between the RL78/G23 pins and the serial NOR flash memory pins.

Table 6-2 Operating Conditions of Demo Projects

Item	Description
MCU	R7F100GLGxFB (RL78/G23)
Operating frequency	32 MHz
Integrated development environment	Renesas Electronics e^2 studio 2024-01 IAR Embedded Workbench for Renesas RL78 5.10.3
C compiler	Renesas Electronics RL78 Family C Compiler Package V1.13.00 LLVM for Renesas RL78 10.0.0.202312 IAR C/C++ Compiler for Renesas RL78 version 5.10.3
Demo projects	Version 1.00
Evaluation board	RL78/G23-64p Fast Prototyping Board (Part Number: RTK7RLG230CLG000BJ)
Operating voltage	3.3 V (supplied via USB)
Emulator	COM Port debug
Serial NOR flash memory	AT25QF641B

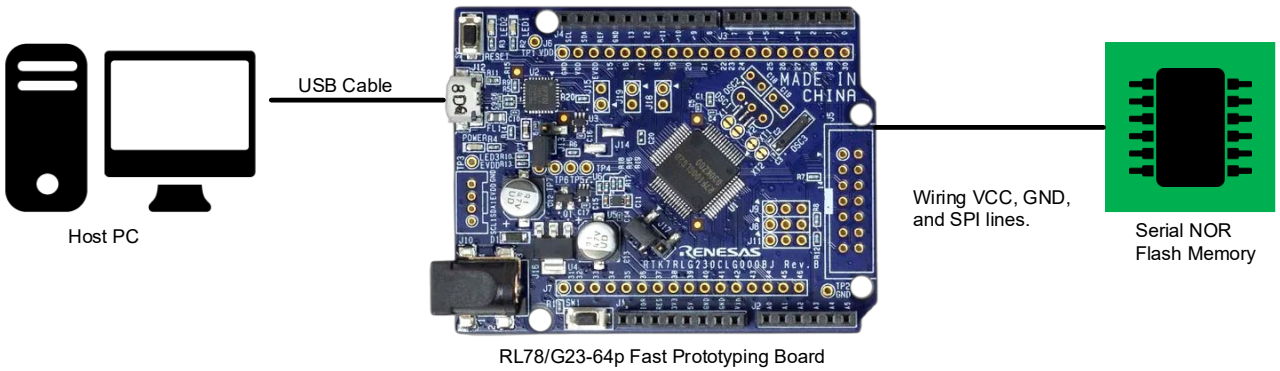


Figure 6-1 Operating Environment of Demo Projects

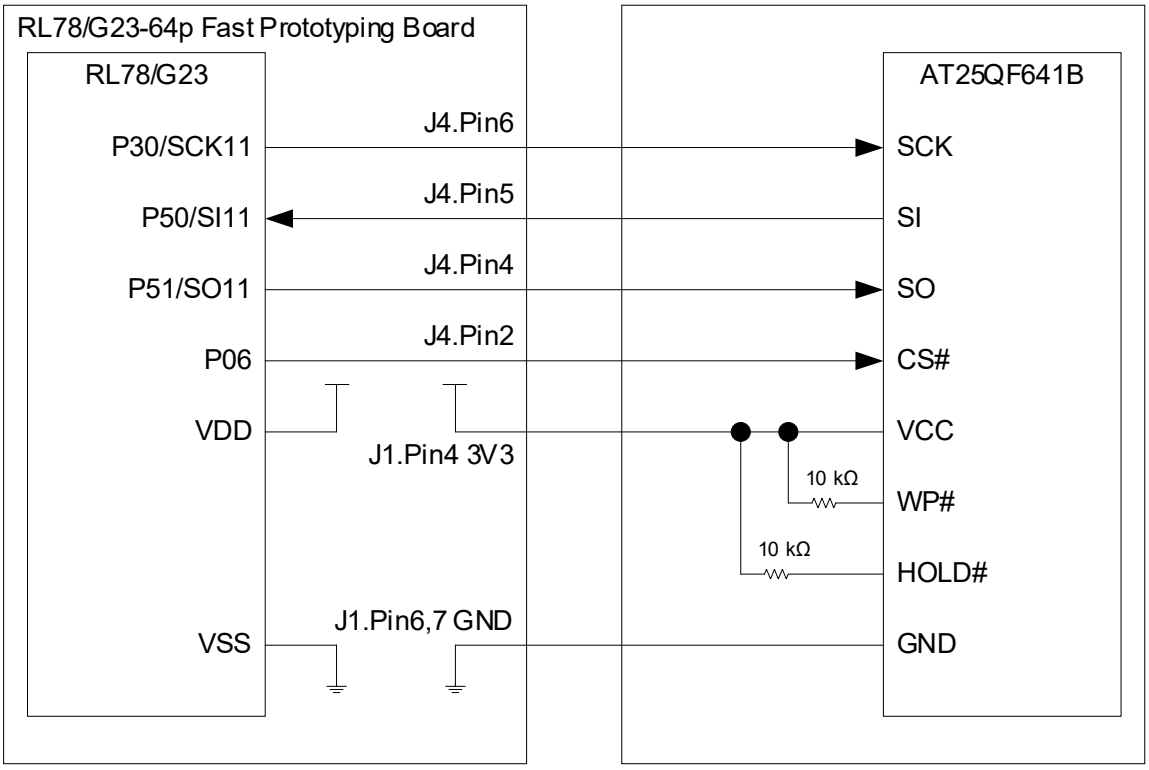


Figure 6-2 Connection Wiring Diagram for the RL78/G23-64p Fast Prototyping Board and Serial NOR Flash Memory

6.3 File Configuration

Table 6-3 shows the source file of the demo projects.

Table 6-3 Demo Project File List

File Name	Description
rl78g23_nor_flash_demo.c	C source file containing the main function of demo projects.

6.4 Functions

Demo projects consist of the main function and internal functions. These functions are described in rl78g23_nor_flash_demo.c.

Table 6-4 Function List

Function Name	Description
main()	Main function of demo projects. • This function calls the following seven functions.
demo_nor_flash_open()	Performs OPEN processing of this module and a busy check of the serial NOR flash memory.
demo_nor_flash_set_write_protect()	Clears the write protect.
demo_nor_flash_erase_write_read()	Erases, writes, and reads data of the serial NOR flash memory, and check the results.
demo_nor_flash_status()	Controls registers of the serial NOR flash memory.
demo_nor_flash_getmemoryinfo	Acquires information about the serial NOR flash memory.
demo_nor_flash_getversion	Acquires version information of this module.
demo_nor_flash_close	Performs a busy check of the serial NOR flash memory and CLOSE processing of this module.

(1) demo_nor_flash_open()

This function runs the R_NOR_FLASH_Open function of the serial NOR flash memory control module, and then performs a busy check of the serial NOR flash memory.

Format

```
void demo_nor_flash_open(
void
)
```

Parameters

None

Return Values

None

Description

- Runs the R_NOR_FLASH_Open function.
- Performs a busy check.

Special Notes

None

(2) demo_nor_flash_set_write_protect()

This function clears the write protect for the serial NOR flash memory.

Format

```
void demo_nor_flash_set_write_protect(
void
)
```

Parameters

None

Return Values

None

Description

- Runs R_NOR_FLASH_SetWriteProtect(0).
- Runs R_NOR_FLASH_ReadStatus to check the Status Register.

Special Notes

None

(3) demo_nor_flash_erase_write_read()

This function erases, writes, and reads data of the serial NOR flash memory, and checks the results.

Format

```
void demo_nor_flash_erase_write_read(
void
)
```

Parameters

None

Return Values

None

Description

- Performs Block Erase (4 KB).
- Writes 256 bytes to the serial NOR flash memory.
- Reads 256 bytes from the serial NOR flash memory.
- Check the results.
- Performs erasure, write of 1 to 511 bytes, and read, and checks the results sequentially.
- Performs Chip Erase.
- Performs Block Erase (32 KB).
- Performs Block Erase (64 KB).

Special Notes

None

(4) demo_nor_flash_status()

This function controls registers of the serial NOR flash memory.

Format

```
void demo_nor_flash_status(
void
)
```

Parameters

None

Return Values

None

Description

- Sets the SEC bit of Status Register 1, and then checks the result.
- Clears the SEC bit of Status Register 1, and then checks the result.
- Sets the CMP bit of Status Register 2, and then checks the result.
- Clears the CMP bit of Status Register 2, and then checks the result.
- Runs R_NOR_FLASH_ReadId.
- Runs R_NOR_FLASH_WriteDisable.

Special Notes

None

(5) demo_nor_flash_getmemoryinfo()

This function acquires information about the serial NOR flash memory.

Format

```
void demo_nor_flash_getmemoryinfo(
void
)
```

Parameters

None

Return Values

None

Description

- Runs R_NOR_FLASH_GetMemoryInfo.

Special Notes

None

(6) demo_nor_flash_getversion()

This function acquires version information of the serial NOR flash memory control module.

Format

```
void demo_nor_flash_getversion(
void
)
```

Parameters

None

Return Values

None

Description

- Runs R_NOR_FLASH_GetVersion.

Special Notes

None

(7) demo_nor_flash_close()

This function closes the serial NOR Flash memory control module.

Format

```
void demo_nor_flash_close(
void
)
```

Parameters

None

Return Values

None

Description

- Runs R_NOR_FLASH_Close.

Special Notes

None

(8) busy_check()

This function performs a busy check of the serial NOR flash memory.

Format

```
e_nor_flash_status_t busy_check(
e_nor_flash_check_busy_t mode,
uint32_t loop_cnt
)
```

Parameters

`mode`

Mode given to R_NOR_FLASH_CheckBusy

`loop_cnt`

Time (ms) during which a busy check is performed

Return Values

None

Description

- Runs R_NOR_FLASH_CheckBusy.
- Terminates processing when a busy check is performed for the specified period of time or when the Serial NOR Flash memory is no longer busy.

Special Notes

None

6.5 Importing a Project into e² studio

Demo projects are provided in e² studio project format. This section describes how to import a project into e² studio. After importing the project, confirm the build and debugger settings.

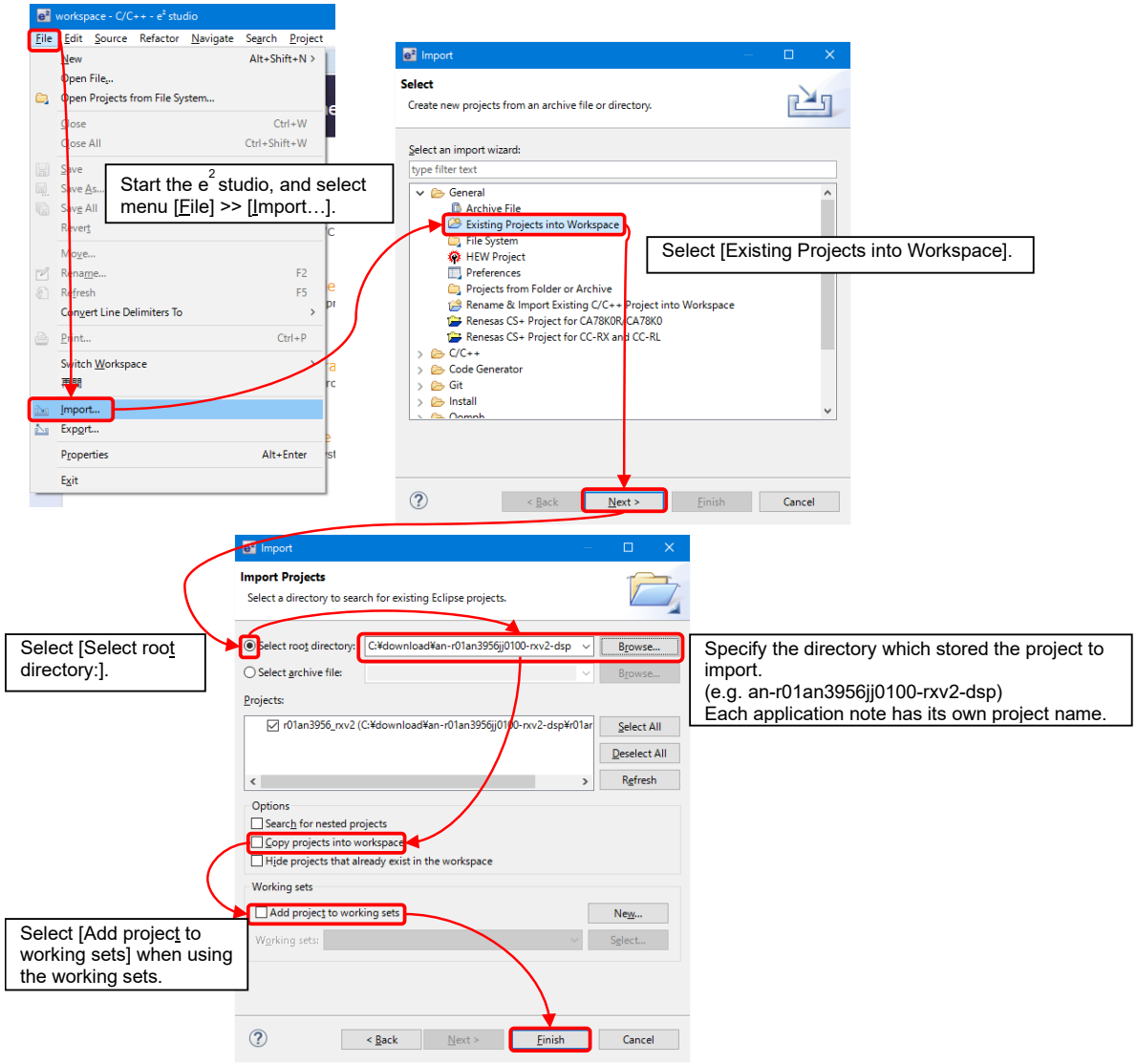


Figure 6-3 Importing a Project into e² studio

7. Appendix

7.1 Operation Confirmation Environment

The following shows the operation confirmation environment for this module.

Table 7-1 Operation Confirmation Environment (Rev. 1.00)

Item	Description
Integrated development environment	Renesas Electronics e^2 studio V2024-01 IAR Embedded Workbench for Renesas RL78 5.10.3
C compiler	Renesas Electronics RL78 Family C Compiler Package V1.13.00 Compiler option: Default settings of the integrated development environment
	LLVM for Renesas RL78 10.0.0.202312 Compiler option: Default settings of the integrated development environment
	IAR C/C++ Compiler for Renesas RL78 version 5.10.3 Compiler option: Default settings of the integrated development environment
Module version	Ver.1.00
Board used	RL78/G23-64p Fast Prototyping Board (Part Number: RTK7RLG230CLG000BJ)

7.2 Controllable Serial NOR Flash Memory Products

The following shows the serial NOR flash memory products that can be controlled by this module.

Table 7-2 Controllable Serial NOR Flash Memory Products

Series	Product Code	Memory
AT25SF	AT25SF041B	4 M-bit
	AT25SF081B	8 M-bit
	AT25SF161B	16 M-bit
	AT25SF321B	32 M-bit
	AT25SF641B	64 M-bit
AT25QF	AT25QF641B	64 M-bit
MX25L	MX25L3233F	32 M-bit
	MX25L6433F	64 M-bit
	MX25L12833F	128 M-bit
	MX25L25645G	256 M-bit
	MX25L51245	512 M-bit
MX66L	MX66L1G45	1 G-bit
MX25R	MX25R6435F	64 M-bit

8. Reference Documents

User’s Manual: Hardware
(Download the latest version from the Renesas Electronics website.)

Technical Updates/Technical News
(Download the latest information from the Renesas Electronics website.)

User's Manual: Development Environment
RL78 Family CC-RL Compiler User's Manual (R20UT3123)
(Download the latest version from the Renesas Electronics website.)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar. 22, 2024	—	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/.