

1. Specifications

The SPI Serial Flash memory control program controls the Micron Technology, Inc. M45PE series SPI Serial Flash memory devices using a Renesas Electronics' MCU.

A clock synchronous single-master control program that is specific to the individual MCU is separately required.

Table 1 summarizes the peripheral devices to be used and their uses. Figure 1 illustrates a sample configuration.

The major functions of the SPI Serial Flash memory control program are summarized below.

- The SPI Serial Flash memory control program is a block-type device driver that assumes a Renesas Electronics MCU as the master device and a Micron Technology, Inc. M45PE series SPI Serial Flash memory as a slave device.
- It controls the devices in the SPI mode using the MCU's internal serial communication function (clock synchronous mode). It can use no more than one user-configured serial I/O channel.
- The SPI Serial Flash memory control program can control a maximum of two SPI Serial Flash memory devices of the same type.
- The communication rate is user-programmable (fixed speed).
- Both big endian and little endian modes are supported (dependent on the MCU in use)

Table 1 Peripheral Devices Used and their Uses

Peripheral Device	Use
MCU internal serial communication function (Clock synchronous mode)	Communication with SPI slave devices using the serial communication function (clock synchronous mode) 1 channel (required)
Port	For SPI slave device select control signals. As many ports as there are SPI slave devices in use are necessary (required).

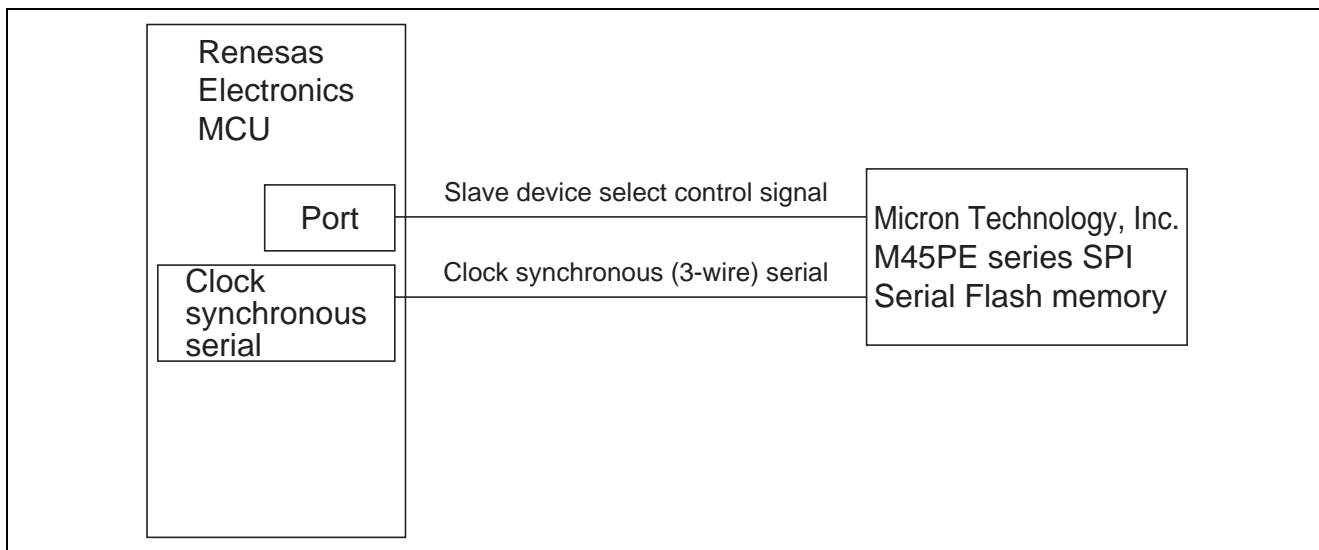


Figure 1 Sample Configuration

2. Conditions for Checking the Operation of the SPI Serial Flash memory Control Software

The sample code described in this application note has been confirmed to run normally under the operating conditions given below.

1. RX610

Table 2 Operating Conditions

Item	Description
Memory used for evaluation	Micron Technology, Inc. M45PE series SPI Serial Flash memory
Microcomputer used for evaluation	RX610 group (program ROM 2 MB/RAM 128 KB)
Operating frequency (microcomputer)	ICLK: 100 MHz, PCLK: 50 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics High-performance embedded Workshop Version 4.07.00.007
C compiler	Renesas Electronics RX family C/C++ compiler package (Toolchain 1.0.0.0) Compiler options: The default settings for the integrated development environment are used.
Endian	Big endian/Little endian
Version of the sample code	Ver.2.00
Software used for evaluation	Clock synchronous single-master control software using the SCI for the RX610, Ver.2.00
Evaluation board used	Renesas Starter Kit for the RX610

2. 78K0R/Kx3-L

Table 3 Operating Conditions

Item	Description
Memory used for evaluation	Micron Technology, Inc. M45PE series SPI Serial Flash memory
Microcomputer used for evaluation	78K0R/KE3-L (program ROM 64KB/RAM 3KB)
Operating frequency (microcomputer)	Main System Clock: 20MHz CPU/peripheral Hardware Clock: 20MHz Serial Clock: 2.5MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Project Manager (PM+ Ver.6.31)
C compiler	Renesas Electronics 78K0R C compiler (CC78K0R Ver.2.13) 78K0R Assembler package (RA78K0R Ver.1.33) Compiler options: The default settings ("-a. -zp") for the integrated development environment are used.
Emulator	Minicube2
Integrated Debugger	Renesas Electronics Integrated Debugger ID78K0R-QB Ver.3.61
Version of the sample code	Ver.2.01
Software used for evaluation	Clock synchronous single-master control software using the SAU CSI mode for the 78K0R/Kx3-L, Ver.2.00
Evaluation board used	78K0R/KE3-L Target Board (QB-78K0RKE3L-TB)

3. Related Application Notes

The applications notes that are related to this application note are listed below. Reference should also be made to those application notes.

- Clock Synchronous Single Master Control Software Using the RX610,621,62N Group SCI (R01AN0534EJ)
- Clock Synchronous Single Master Control Software Using the RX621,62N Group RSPI (R01AN0323EJ)
- Clock Synchronous Single Master Control Software Using the 78K0R/Kx3-L SAU CSI mode (R01AN0708EJ)

4. Hardware Description

4.1 List of Pins

Table 4 lists the MCU pins that are used and their uses.

Table 4 List of Pins Used

Pin Name	I/O	Description
CLK * ¹	Output	Clock output
DataOut * ¹	Output	Master data output
DataIn * ¹	Input	Master data input
Port(CS#) * ¹	Output	Storage device select output

Note: *¹ In this application note, the pin names CLK, DataIn, DataOut, and Port (CS#) are used in accordance with the pin names used in the sample code.

4.2 Reference Circuit

Figure 2 shows a sample wiring configuration.

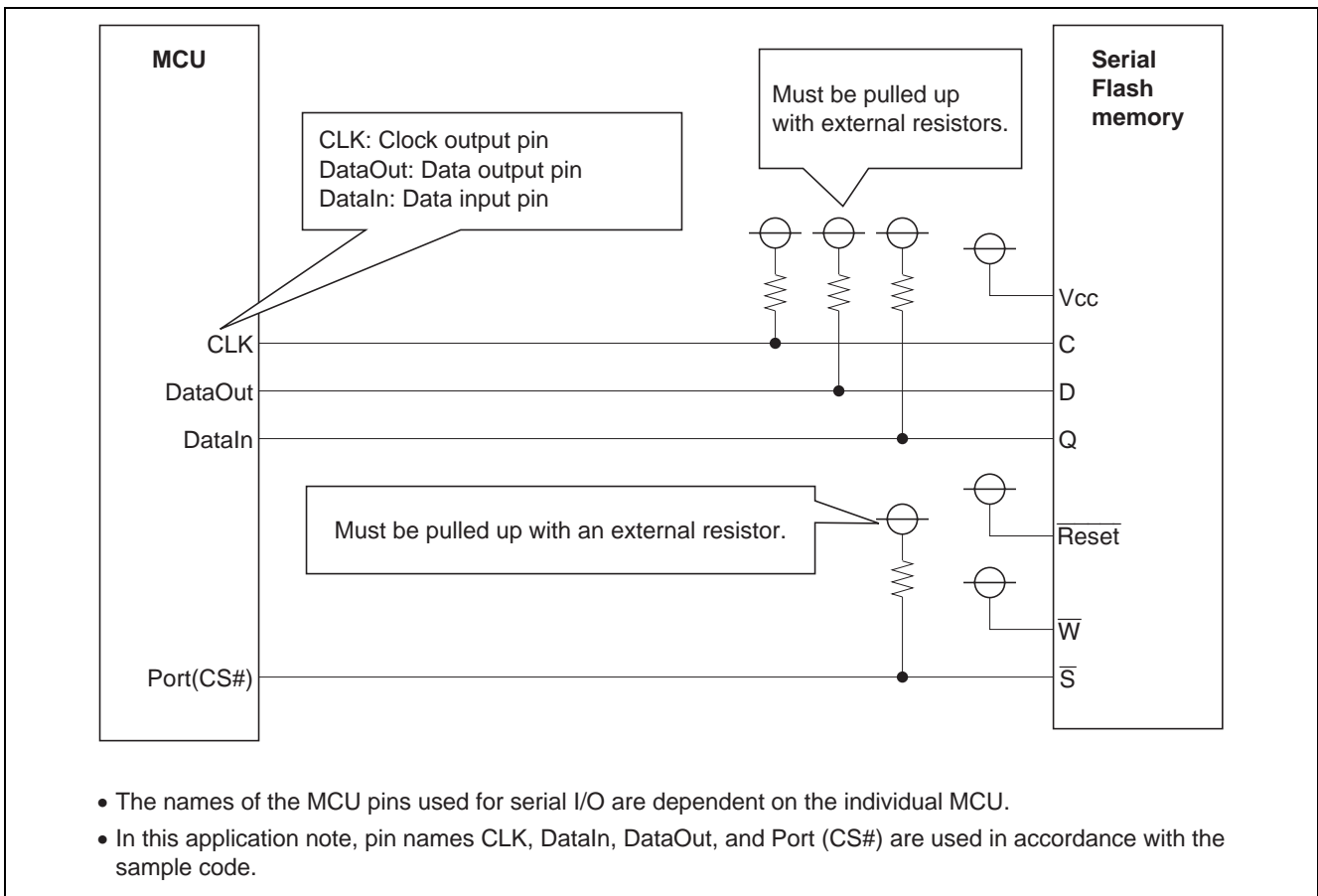


Figure 2 Sample Wiring Diagram for an MCU and an SPI Slave Device

5. Software Description

5.1 Operation Outline

The MCU's clock synchronous (3-wire) serial communication function is used to realize the control of Serial Flash memory devices.

The sample code explained in this section provides the following control functions:

- Connects the S pin of the SPI slave device to the Port pin of the MCU and controls it as an MCU's general port output (controlled by this sample code).
- Controls the input/output of the data in the clock synchronous mode (using an internal clock). (This sample code uses the MCU-specific clock synchronous single-master control software.)

In this sample code, the byte offset value of the data on the device is made equal to the byte offset value in the source or destination memory as illustrated in the figure below.

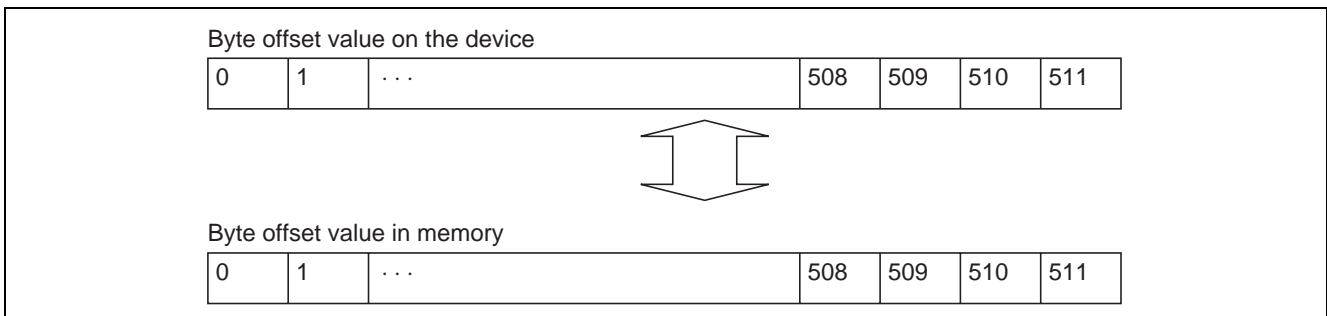


Figure 3 Storage Format of the Transferred Data

5.1.1 Clock Synchronous Mode Timing

The SPI mode 3 (CPOL=1, CPHA=1) timing shown in Figure 4 is used to control the Serial Flash memory.

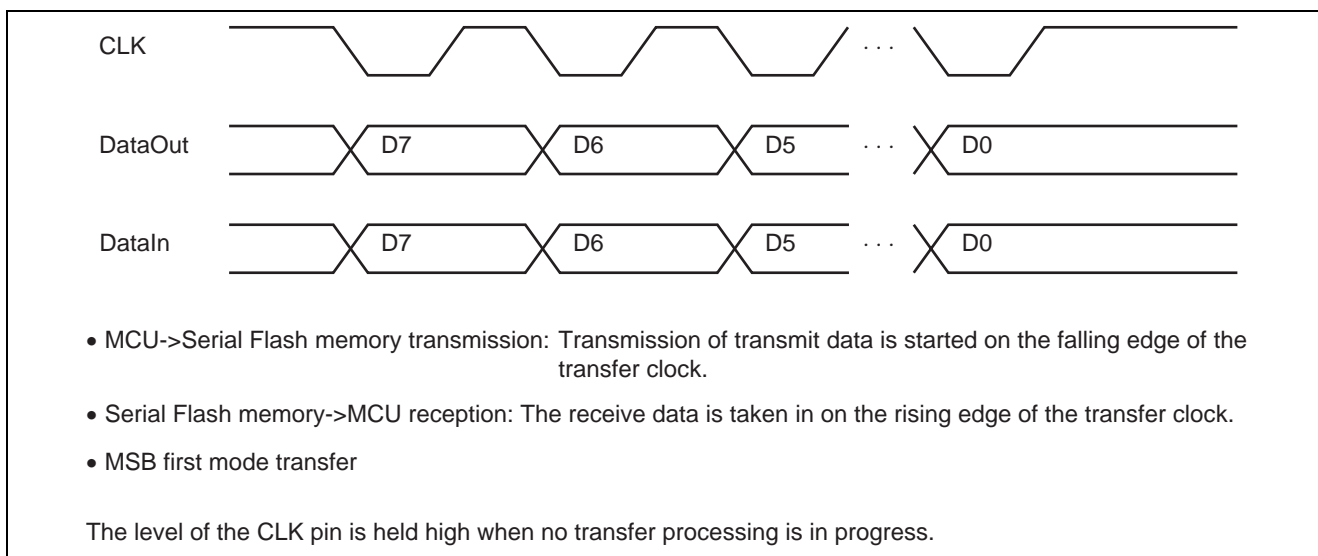


Figure 4 Clock Synchronous Mode Timing Setup

For available serial clock frequencies, see the datasheets for the individual MCUs and SPI devices.

5.1.2 Serial Flash memory S Pin Control

The S pin of the Serial Flash memory is connected to the Port pin of the MCU and controlled as an MCU general port output.

The interval between the falling edge of the S (MCU's Port(CS#)) signal of the Serial Flash memory and the falling edge of the C (MCU's CLK) signal of the Serial Flash memory is controlled with software wait processing to account for the Serial Flash memory S setup time.

The interval between the rising edge of the C (MCU's CLK) signal of the Serial Flash memory and the rising edge of the S (MCU's Port (CS#)) signal of the Serial Flash memory is controlled with software wait processing to account for the Serial Flash memory S hold time.

Check the datasheet for the Serial Flash memory in use and set up the software wait times that are appropriate to your system.

5.1.3 Serial Flash memory Instruction Code

The instruction codes listed in the table below are available for controlling the Serial Flash memory. These codes are used to carry out command control of the Serial Flash memory.

Table 5 Instruction Set

Instruction	Description	Instruction Format
WREN	Write Enable	0000 0110
WRDI	Write Disable	0000 0100
RDID	Read Identification	1001 1111
RDSR	Read Status-Register	0000 0101
READ	Read Data Bytes	0000 0011
FAST READ	Read Data Bytes at Higher Speed	0000 1011
PW	Page Write	0000 1010
PP	Page Program	0000 0010
PE	Page Erase	1101 1011
SE	Sector Erase	1101 1000
DP	Deep Power-down	1011 1001
RES	Release from Deep Power-down	1010 1011

5.1.4 Serial Flash memory Status Register

The status register of the Serial Flash memory can be read and written using dedicated instructions.

See the Serial Flash memory’s datasheet for details on the individual status register bits.

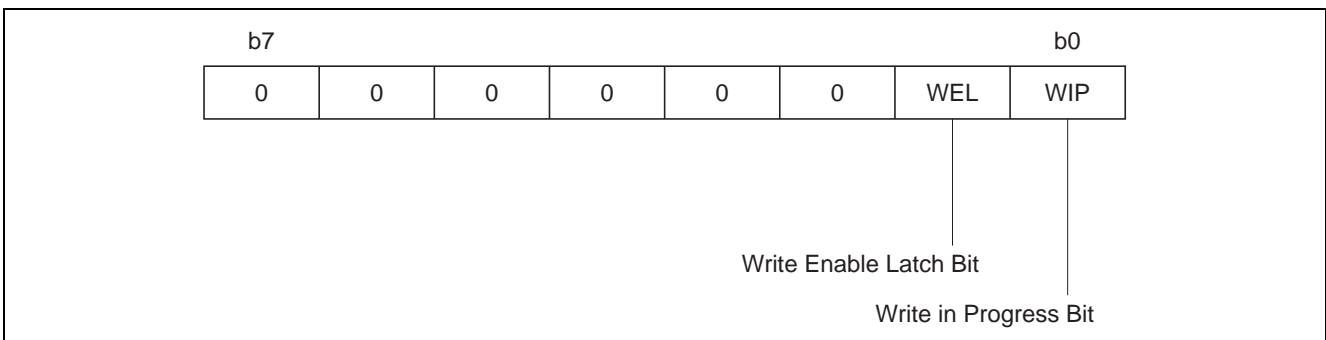


Figure 5 Status Register Configuration

5.2 Software Control Outline

5.2.1 Software Configuration

The sample code ranks in the higher-level layer of the SPI Serial Flash memory control software system (flash memory control software shown in Figure 6).

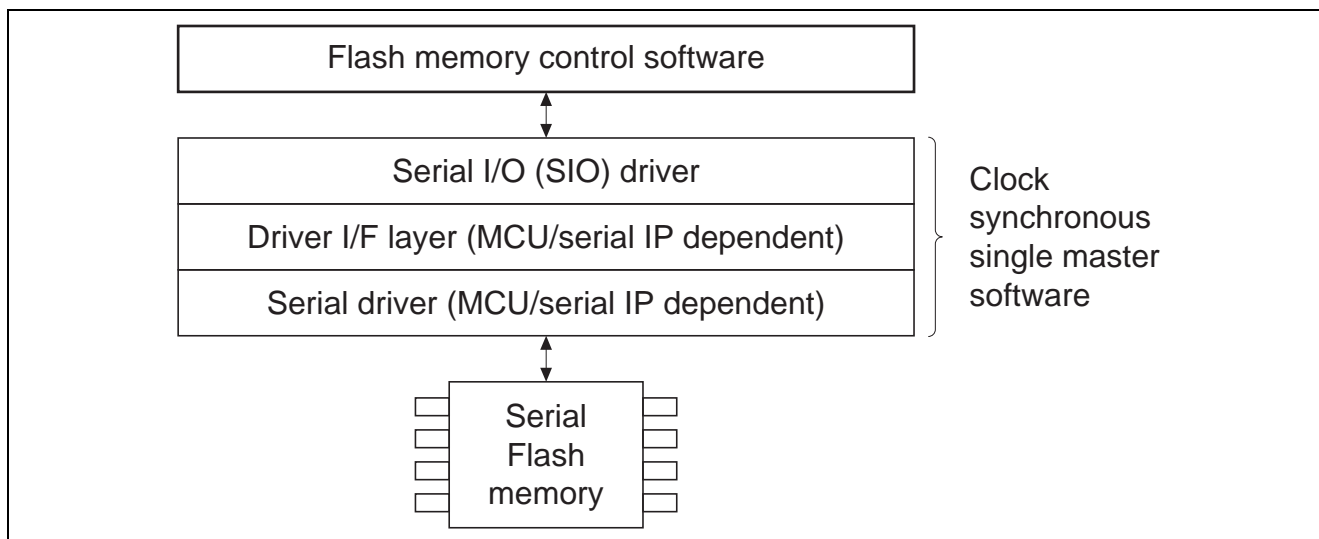


Figure 6 Software Configuration

The general control procedure is given below.

- (1) Set the Port (CS#) signal low.
- (2) Software wait.
- (3) Send/receive command/data using the clock synchronous single-master software.
- (4) Software wait.
- (5) Set the Port (CS#) signal high.

This sample code is made up of the following eight basic routines:

- Chip Select pin initialization
Set the Port (CS#) pin high.
- Chip Select pin control
Set the Port (CS#) pin high/low.
- Software wait
Adjust timing using software wait.
- Serial enabling
Set the DataIn pin for port input, set the DataOut and CLK pins high, Enable serial I/O and set the bit rate.
- Command transmission
Send command to the Serial Flash memory.
- Data transmission
Send data to the Serial Flash memory.
- Data reception
Receive response/data from the Serial Flash memory.
- Serial disabling
Disable serial I/O, set the DataIn pin for port input, set the DataOut and CLK pins high.

5.2.2 Chip Select Pin Initialization (R_SPI_FLASH_Init_Port())

This routine sets the Chip Select signal of the slave device high (deactivates the device before operation). The basic control procedure is as follows, though the actual control procedure varies from MCU to MCU:

- (1) Set the port output value to "High" output (to generate a high output when the port configuration is switched to "output").
- (2) Set the port for "output."
- (3) Set the port output value to "High" output.

5.2.3 Chip Select Pin Control (FLASH_SET_CS())

This routine sets the Chip Select signal of the slave device high or low.

5.2.4 Software Wait (mtl_wait_lp())

This routine controls wait processing using a software loop.

5.2.5 Serial Enabling (R_SIO_Enable())

This routine enables the serial interface using the following procedure:

- (1) Sets the DataIn pin to be used for serial I/O for port input and set the DataOut and CLK pins high.
- (2) Enables the serial I/O function and switches the DataIn pin for data input, the DataOut pin for data output, and the CLK pin for clock output.
- (3) Sets the baud rate (bit rate) to be used for serial I/O.

For details on R_SIO_Enable(), refer to the individual clock synchronous single-master software application note.

5.2.6 Command Transmission (R_SPI_FLASH_Send_Cmd())

This routine sends an instruction code (command) to the Serial Flash memory.

R_SIO_Rx_Data () is used to receive the response to the command.

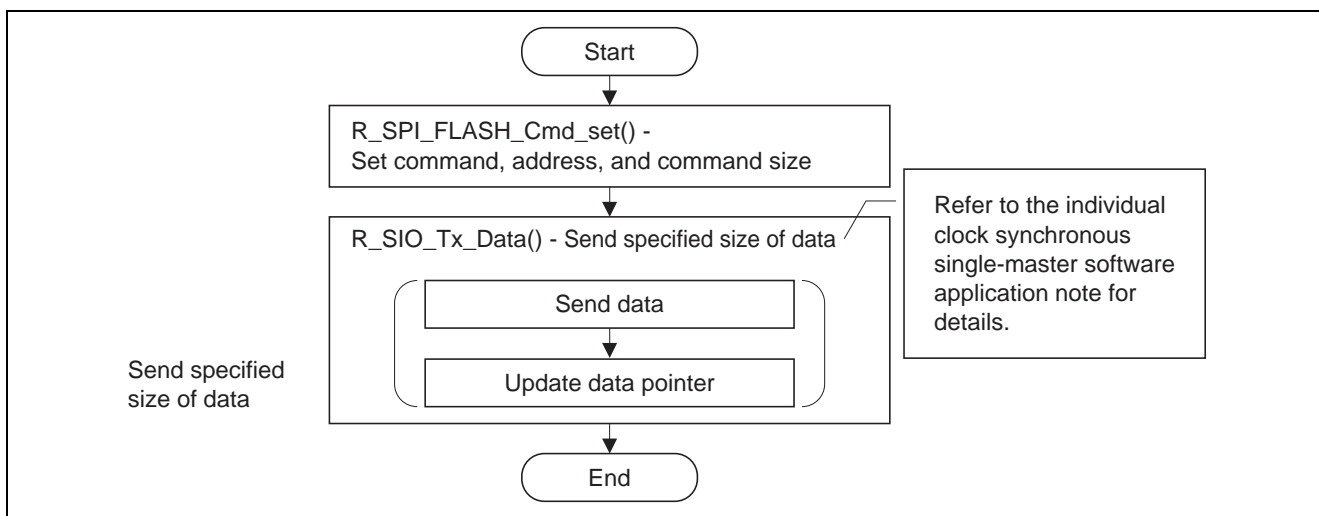


Figure 7 Outline of Command Transmission Processing (R_SPI_FLASH_Send_Cmd())

5.2.7 Data Transmission (R_SIO_Tx_Data ())

This routine sends data using the serial I/O function. It sends an instruction code, address information, write data or the value of the status register.

Refer to the individual clock synchronous single-master software application note for details.

5.2.8 Data Reception (R_SIO_Rx_Data ())

This routine receives data using the serial I/O function. It receives either read data or the value of the status register.

Refer to the individual clock synchronous single-master software application note for details.

5.2.9 Serial Disabling (R_SIO_Disable ())

This routine switches the pin to be used for serial I/O to a port pin and sets the DataIn pin for port input and sets the DataOut and CLK pins high.

Refer to the individual clock synchronous single-master software application note for details.

5.3 Sizes of Required Memory

The sizes of the required memory areas are given below.

1. RX610

See chapter 2, Conditions for Checking the Operation of the SPI Serial Flash memory Control Software, for the environment.

Table 6 Sizes of Required Memory

Memory Used	Size	Remarks
ROM	1010 bytes (little endian) 1074 bytes (little endian)	R_SPI_FLASH_m45pe_usr.c R_SPI_FLASH_m45pe_io.c
RAM	0 bytes (little endian) 5 bytes (little endian)	R_SPI_FLASH_m45pe_usr.c R_SPI_FLASH_m45pe_io.c
Maximum user stack size	152 bytes	
Maximum interrupt stack size	—	Not interrupt used

Note: The sizes of required memory areas vary with the version and compiler options of the C compiler. These sizes do not include the size of the memory that is used by the clock synchronous single-master software. The above-mentioned memory sizes vary with the type of MCU to be used. The above-mentioned memory sizes vary with the endian mode adopted. The above-mentioned Maximum user stack size is included the stack size of clock synchronous single-master control software which lies in the lower-level layer of the software.

2. 78K0R/Kx3-L

See chapter 2, Conditions for Checking the Operation of the SPI Serial Flash memory Control Software, for the environment.

Table 7 Sizes of Required Memory

Memory Used	Size	Remarks
ROM	1709 bytes 1536 bytes	R_SPI_FLASH_m45pe_usr.c R_SPI_FLASH_m45pe_io.c
RAM	0 bytes 6 bytes	R_SPI_FLASH_m45pe_usr.c R_SPI_FLASH_m45pe_io.c
Maximum user stack size	164 bytes	
Maximum interrupt stack size	—	Not interrupt used

Note: The sizes of required memory areas vary with the version and compiler options of the C compiler. These sizes do not include the size of the memory that is used by the clock synchronous single-master software. The above-mentioned memory sizes vary with the type of MCU to be used. The above-mentioned Maximum user stack size is included the stack size of clock synchronous single-master control software which lies in the lower-level layer of the software.

5.4 File Configuration

Table 8 lists the files that are used for the sample code. The table excludes the files that are automatically generated by the integrated development environment.

Table 8 File Configuration

\an_r01an0567ej_mcu	<DIR>	Folder for the sample code
r01an0567ej0101_mcu.pdf		Application note
\r01an0567ej_src	<DIR>	Folder for storing the programs
\r_spi_flash_m45pe	<DIR>	Folder for the Serial Flash memory control software
R_SPI_FLASH_m45pe.h		Header file
R_SPI_FLASH_m45pe_io.c		I/O module
R_SPI_FLASH_m45pe_io.h		I/O module common definitions
R_SPI_FLASH_m45pe_usr.c		User I/F module
R_SPI_FLASH_sfr.h.78k0r		Common register definitions
R_SPI_FLASH_sfr.h.rx610		Common register definitions
\sample	<DIR>	Folder for storing the test program
testmain.c		Sample program for testing

Note: An MCU-specific clock synchronous single-master control program is separately required.

5.5 List of Constants

5.5.1 Return Values

Table 9 lists the return values that are returned by the sample code.

Table 9 Return Values

Constant Name	Value	Description
FLASH_OK	(error_t)(0)	Successful Operation
FLASH_ERR_PARAM	(error_t)(-1)	Parameter Error
FLASH_ERR_HARD	(error_t)(-2)	Hardware Error
FLASH_ERR_OTHER	(error_t)(-7)	Other Error

5.5.2 Command Definitions

Table 10 lists the command definitions that are used in the sample code.

Table 10 Command Definitions

Constant Name	Value	Description
FLASH_CMD_WREN	(uint8_t)0x06	Write Enable
FLASH_CMD_WRDI	(uint8_t)0x04	Write Disable
FLASH_CMD_RDSR	(uint8_t)0x05	Read Status Register
FLASH_CMD_READ	(uint8_t)0x0b	Read for Memory Array at Higher Speed
FLASH_CMD_READ	(uint8_t)0x03	Read for Memory Array
FLASH_CMD_WRITE	(uint8_t)0x0a	Page Write for Memory Array
FLASH_CMD_WRITE	(uint8_t)0x02	Page Program for Memory Array
FLASH_CMD_SE	(uint8_t)0xd8	S_Erase for Memory Array
FLASH_CMD_PE	(uint8_t)0xdb	P_Erase for Memory Array
FLASH_CMD_DP	(uint8_t)0xb9	Deep Power Down
FLASH_CMD_RES	(uint8_t)0xab	Release Deep Power Down
FLASH_CMD_RDID	(uint8_t)0x9f	Read Identification

5.5.3 Miscellaneous Definitions

Table 11 lists miscellaneous definitions that are used in the sample code.

Table 11 Miscellaneous Definitions

Constant Name	Value	Description
FLASH_LOG_ERR	1	Log Type: Error
FLASH_TRUE	(uint8_t)0x01	Flag "ON"
FLASH_FALSE	(uint8_t)0x00	Flag "OFF"
FLASH_WP_NONE	(uint8_t)0x00	Write-Protection Status : None setting
FLASH_P_ERASE	(uint8_t)0x00	Erasure Type: Page Erase
FLASH_S_ERASE	(uint8_t)0x01	Erasure Type: Sector Erase
FLASH_MEM_SIZE	(uint32_t)131072	Memory size (in bytes) The value shown to the left is for 1 Mbit memory.
FLASH_PAGE_ADDR	(uint32_t) 0xfffff00	Page address mask value on erasing a page The value shown to the left is for 1 Mbit memory.
FLASH_SECT_ADDR	(uint32_t)0xffff0000	Sector address mask value on erasing a sector The value shown to the left is for 1 Mbit memory.
FLASH_WPAG_SIZE	(uint32_t)256	Page size (in bytes) The value shown to the left is for 1 Mbit memory.
FLASH_ADDR_SIZE	(uint8_t)3	Address size (in bytes) The value shown to the left is for 1 Mbit memory.
FLASH_CMD_SIZE	(uint8_t)1	Command size (in bytes)
FLASH_STSREG_SIZE	(uint16_t)1	Status register size (in bytes)
FLASH_IDDATA_SIZE	(uint16_t)3	ID Data size (in bytes)
FLASH_SHORT_SIZE	(uint32_t)0x0000fff0	Sets the maximum transfer size for lower-level functions (set to 0xFFFF or lower.)
FLASH_HI	(uint8_t)0x01	Port "H"
FLASH_LOW	(uint8_t)0x00	Port "L"
FLASH_OUT	(uint8_t)0x01	Port Output Setting
FLASH_IN	(uint8_t)0x00	Port Input Setting
FLASH_WBUSY_WAIT	(uint16_t)18000	Write Busy Waiting Time 18000 × 1 μs = 18 ms
FLASH_T_WBUSY_WAIT	(uint16_t)MTL_T_1US	Write Busy Completion Polling Time
FLASH_T_EBUSY_WAIT	(uint16_t)MTL_T_1MS	Erase Busy Completion Polling Time
FLASH_T_DP_WAIT	(uint16_t)MTL_T_4US	Deep Standby Completion Waiting Time
FLASH_T_RES_WAIT	(uint16_t)MTL_T_30US	Release Deep Standby Completion Waiting Time
FLASH_T_CS_HOLD	(uint16_t)MTL_T_1US	CS Stability Waiting Time
FLASH_T_R_ACCESS	(uint16_t)MTL_T_1US	Reading Start Waiting Time
FLASH_REG_SRWD	(uint8_t)0x80	Status Register Write Disable
FLASH_REG_BP1	(uint8_t)0x08	Not Used
FLASH_REG_BP0	(uint8_t)0x04	Not Used
FLASH_REG_WEL	(uint8_t)0x02	Write Enable Latch Bit
FLASH_REG_WIP	(uint8_t)0x01	Write In Progress Bit
FLASH_BYTE_READ	2	Number of bytes used to identify the execution of a 1-byte read
FLASH_BYTE_WRITE	2	Number of bytes used to identify the execution of a 1-byte write

5.8 List of Functions

Table 13 lists the functions that are used in the sample code.

Table 13 List of Functions

Function Name	Outline
R_SPI_FLASH_Init_Driver	Initialize driver.
R_SPI_FLASH_Read_Status	Read status.
R_SPI_FLASH_Read_Data	Read data.
R_SPI_FLASH_Write_Data	Write data.
R_SPI_FLASH_PageErase	Erases the page.
R_SPI_FLASH_SectorErase	Erases the sector.
R_SPI_FLASH_DeepPDown	Deep Power-down Processing
R_SPI_FLASH_ReleaseDeepPDown	Deep Power-down Release Processing
R_SPI_FLASH_ReadID	Read ID.
R_SPI_FLASH_Init_Port	Initialize port.
R_SPI_FLASH_Send_Cmd	Send command.
R_SPI_FLASH_Write_En	Enable writes.
R_SPI_FLASH_Write_Di	Disable writes.
R_SPI_FLASH_Read_StsReg	Read status register.
R_SPI_FLASH_Wait_Busy	Busy wait.
R_SPI_FLASH_Write_Page	Write.
R_SPI_FLASH_Read_Memory	Read memory.
R_SPI_FLASH_Erase	Erase.
R_SPI_FLASH_DPD	Deep Power-down Command Processing
R_SPI_FLASH_ReleaseDPD	Deep Power-down Release Command Processing
R_SPI_FLASH_ID	Read JEDEC ID.
R_SPI_FLASH_Cmd_set	Set command.

When using an MCU that incorporates cache memory, allocate the read/write buffer to a non-cache area.

The address of the buffer for storing read or write data is dependent on the individual MCU-specific clock synchronous single-master control software in the lower-level layer; there are cases in which it is necessary to specify a 4-byte boundary. Refer to the individual MCU-specific clock synchronous single-master control software application note.

5.9 Function Details

5.9.1 Driver Initialization

R_SPI_FLASH_Init_Driver

Synopsis	Initializes driver.
Headers	R_SPI_FLASH_m45pe.h
Declaration	error_t R_SPI_FLASH_Init_Driver(void)
Explanation	<ul style="list-style-type: none"> • Calls the R_SPI_FLASH_Init_Port() function to initialize the CS# pin. • Calls the serial I/O driver's R_SIO_Init_Driver() function to initialize the I/O port. • This function must be called only once at system start time.
Arguments	void
Return value	<ul style="list-style-type: none"> • A description of the results of initialization is returned. FLASH_OK ; Successful operation FLASH_ERR_OTHER ; Other error Returns the return value of R_SIO_Init_Driver().
Remarks	None

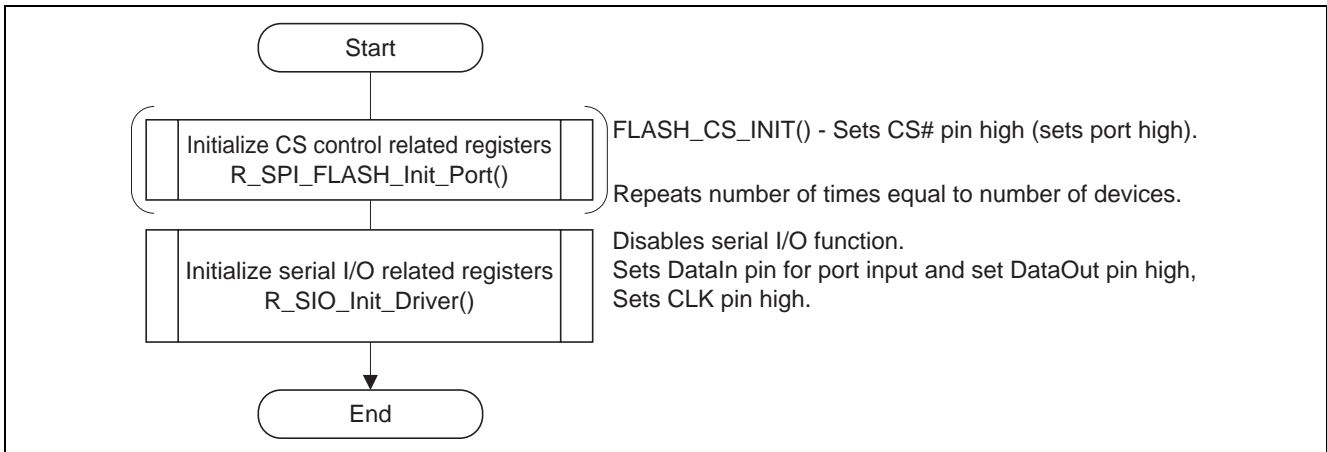


Figure 8 Driver Initialization Processing Outline

5.9.2 Read Status Processing

R_SPI_FLASH_Read_Status

Synopsis	Reads status register.		
Headers	R_SPI_FLASH_m45pe.h		
Declaration	error_t R_SPI_FLASH_Read_Status(uint8_t DevNo, uint8_t FAR* pStatus)		
Explanation	<ul style="list-style-type: none"> • Reads the contents of the status register into pStatus. Set up a 1-byte read buffer. • The read status buffer (pStatus) is loaded with the following information: <ul style="list-style-type: none"> Bit 7 to 2: Reserved (All "0") Bit 1: WEL 1: Internal Write Enable Latch is set 0: Internal Write Enable Latch is reset Bit 0: WIP 1: Program or Erase cycle is in progress 0: No Program or Erase cycle is in progress 		
Arguments	uint8_t	DevNo	; Device number
	uint8_t FAR*	pStatus	; Buffer for storing read status
Return value	<ul style="list-style-type: none"> • Returns the result of reading the status register data. FLASH_OK ; Successful operation FLASH_ERR_PARAM ; Parameter error FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error 		
Remarks	None		

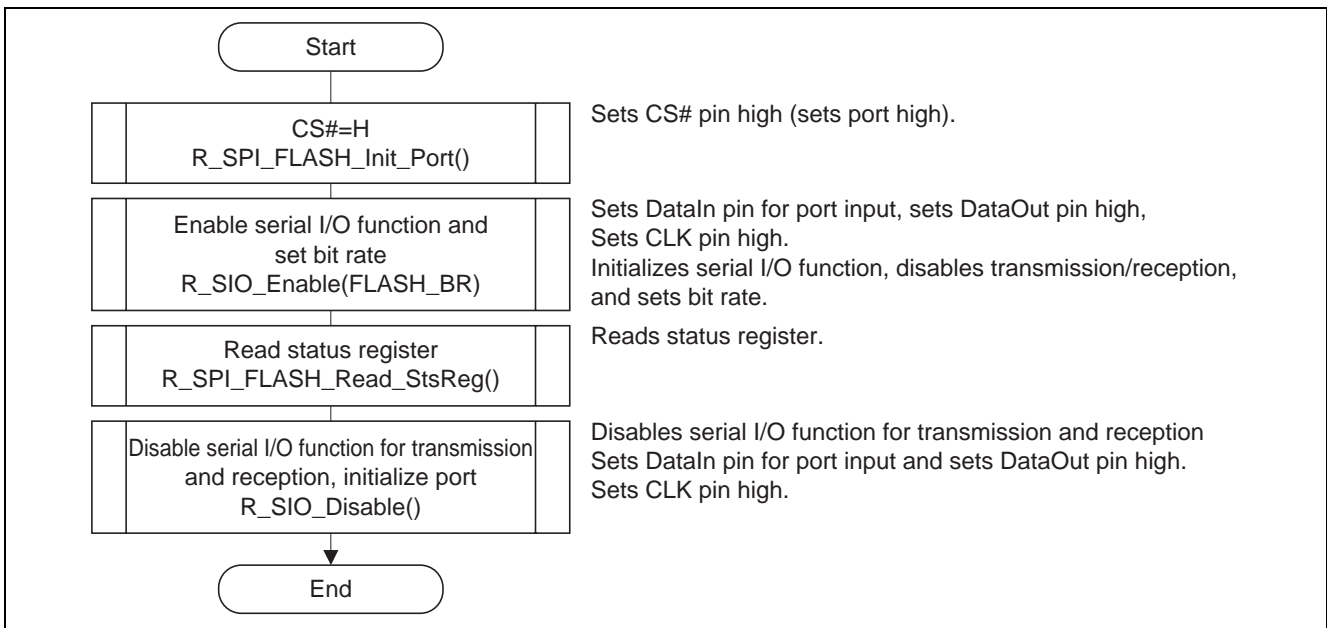


Figure 9 Status Register Read Processing Outline

5.9.3 Data Read Processing

R_SPI_FLASH_Read_Data

Synopsis	Reads data.																
Headers	R_SPI_FLASH_m45pe.h																
Declaration	error_t R_SPI_FLASH_Read_Data(uint8_t DevNo, uint32_t RAddr, uint32_t RCnt, uint8_t FAR* pData)																
Explanation	<ul style="list-style-type: none"> Reads the specified number of bytes from flash memory, 1 byte at a time, starting at the specified address and stores the bytes in pData. 																
Arguments	<table border="0"> <tr> <td>uint8_t</td> <td>DevNo</td> <td>;</td> <td>Device number</td> </tr> <tr> <td>uint32_t</td> <td>RAddr</td> <td>;</td> <td>Read start address</td> </tr> <tr> <td>uint32_t</td> <td>RCnt</td> <td>;</td> <td>Number of read bytes</td> </tr> <tr> <td>uint8_t FAR*</td> <td>pData</td> <td>;</td> <td>Pointer to read data buffer</td> </tr> </table>	uint8_t	DevNo	;	Device number	uint32_t	RAddr	;	Read start address	uint32_t	RCnt	;	Number of read bytes	uint8_t FAR*	pData	;	Pointer to read data buffer
uint8_t	DevNo	;	Device number														
uint32_t	RAddr	;	Read start address														
uint32_t	RCnt	;	Number of read bytes														
uint8_t FAR*	pData	;	Pointer to read data buffer														
Return value	<ul style="list-style-type: none"> Returns the result of the data read. <table border="0"> <tr> <td>FLASH_OK</td> <td>;</td> <td>Successful operation</td> </tr> <tr> <td>FLASH_ERR_PARAM</td> <td>;</td> <td>Parameter error</td> </tr> <tr> <td>FLASH_ERR_HARD</td> <td>;</td> <td>Hardware error</td> </tr> <tr> <td>FLASH_ERR_OTHER</td> <td>;</td> <td>Other error</td> </tr> </table>	FLASH_OK	;	Successful operation	FLASH_ERR_PARAM	;	Parameter error	FLASH_ERR_HARD	;	Hardware error	FLASH_ERR_OTHER	;	Other error				
FLASH_OK	;	Successful operation															
FLASH_ERR_PARAM	;	Parameter error															
FLASH_ERR_HARD	;	Hardware error															
FLASH_ERR_OTHER	;	Other error															
Remarks	<ul style="list-style-type: none"> The highest read address is flash memory size -1. 																

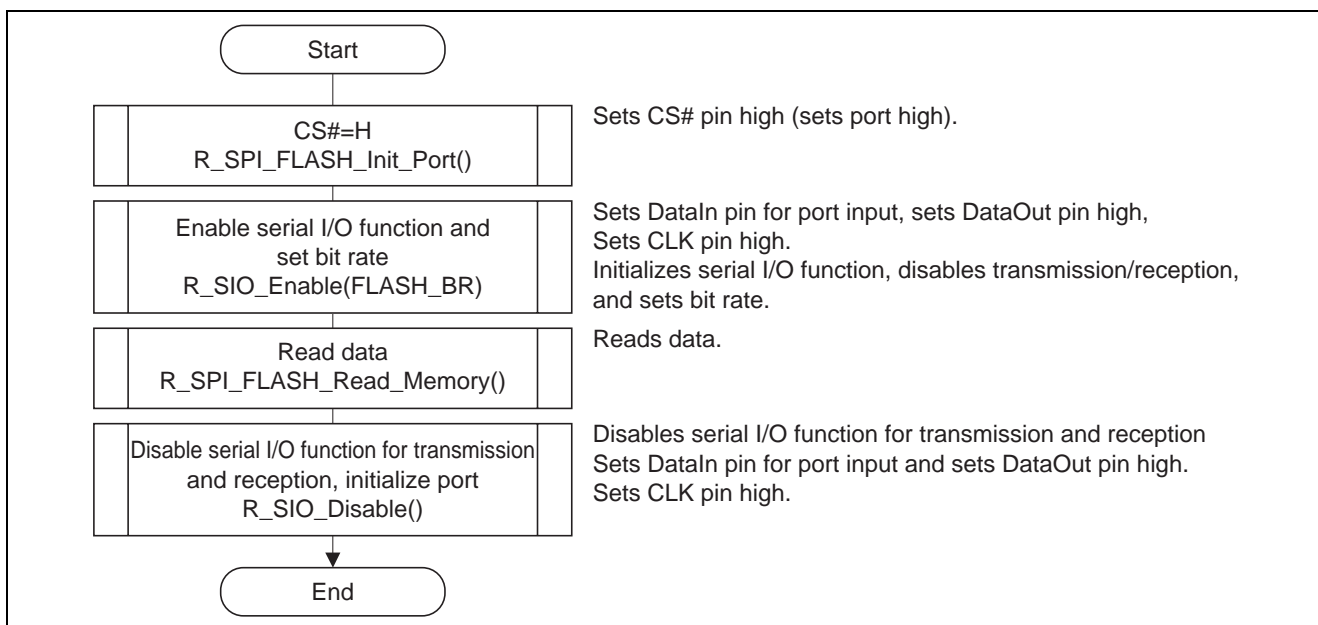


Figure 10 Data Read Processing Outline

5.9.4 Data Write Processing

R_SPI_FLASH_Write_Data

Synopsis	Writes data.		
Headers	R_SPI_FLASH_m45pe.h		
Declaration	error_t R_SPI_FLASH_Write_Data(uint8_t DevNo, uint32_t WAddr, uint32_t WCnt, uint8_t FAR* pData)		
Explanation	<ul style="list-style-type: none"> Writes the data from pData into flash memory the specified number of bytes starting at the specified address. 		
Arguments	uint8_t	DevNo	; Device number
	uint32_t	WAddr	; Write start address
	uint32_t	WCnt	; Number of bytes to write
	uint8_t FAR*	pData	; Pointer to write data buffer
Return value	<ul style="list-style-type: none"> Returns the result of the data write. 		
	FLASH_OK		; Successful operation
	FLASH_ERR_PARAM		; Parameter error
	FLASH_ERR_HARD		; Hardware error
	FLASH_ERR_OTHER		; Other error
Remarks	<ul style="list-style-type: none"> The highest read address is flash memory size -1. 		

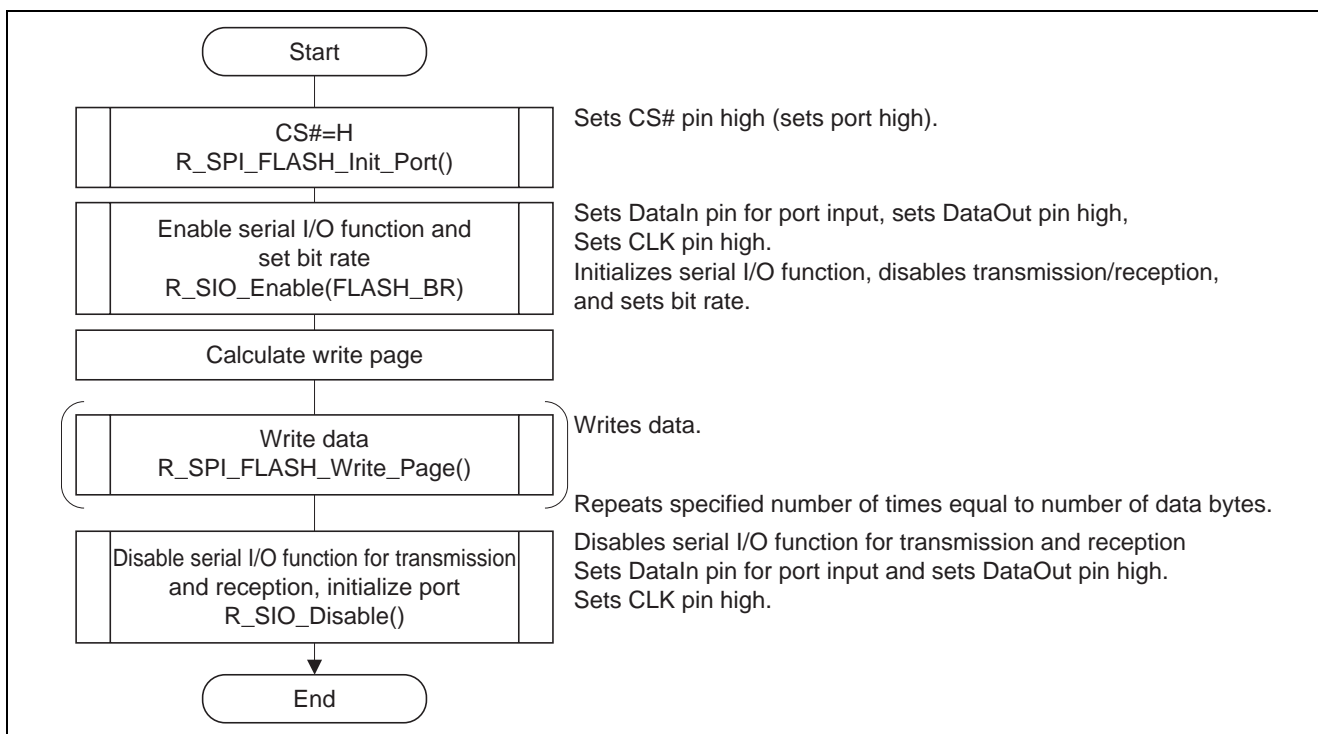


Figure 11 Data Write Processing Outline

5.9.5 Page Erasure Processing

R_SPI_FLASH_PageErase

Synopsis	Performs page erasure.
Headers	R_SPI_FLASH_m45pe.h
Declaration	error_t R_SPI_FLASH_PageErase(uint8_t DevNo, uint32_t EAddr)
Explanation	<ul style="list-style-type: none"> Erases all data in the specified page.
Arguments	uint8_t DevNo ; Device number uint32_t EAddr ; Erasure address
Return value	<ul style="list-style-type: none"> A description of the results of erasure is returned. FLASH_OK ; Successful operation FLASH_ERR_PARAM ; Parameter error FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error
Remarks	None

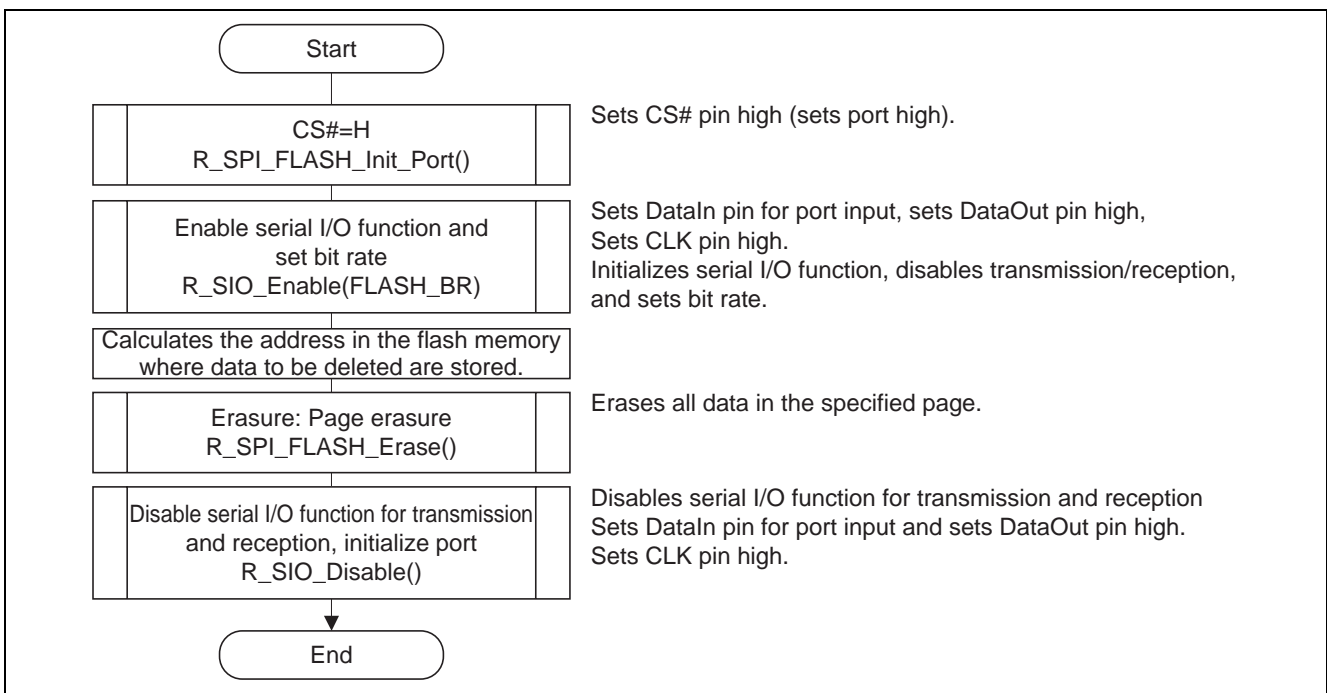


Figure 12 Page Erasure Processing Outline

5.9.6 Sector Erasure Processing

R_SPI_FLASH_SectorErase

Synopsis	Performs sector erasure.
Headers	R_SPI_FLASH_m45pe.h
Declaration	error_t R_SPI_FLASH_SectorErase(uint8_t DevNo, uint32_t EAddr)
Explanation	<ul style="list-style-type: none"> Erased all data in the specified sector.
Arguments	uint8_t DevNo ; Device number uint32_t EAddr ; Erasure address
Return value	<ul style="list-style-type: none"> A description of the results of erasure is returned. FLASH_OK ; Successful operation FLASH_ERR_PARAM ; Parameter error FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error
Remarks	None

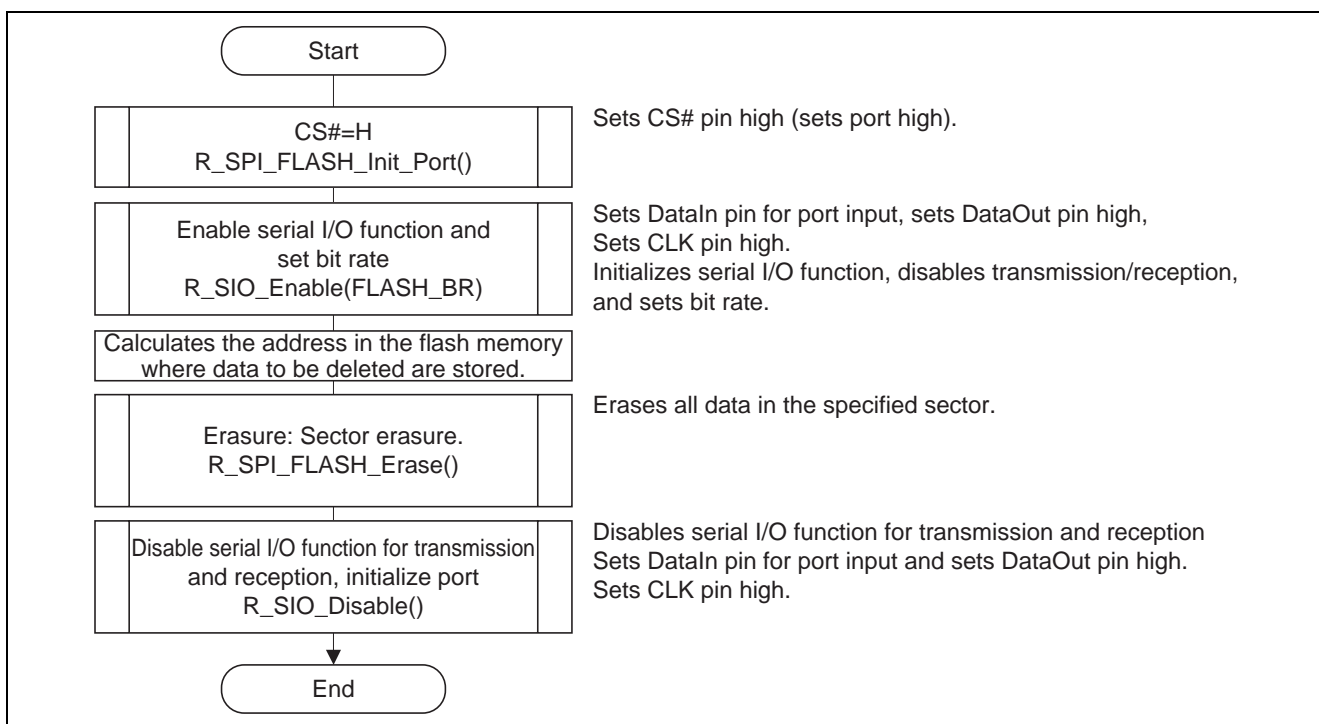


Figure 13 Sector Erasure Processing Outline

5.9.7 Deep Power-down Processing

R_SPI_FLASH_DeepPDown

Synopsis	Sets deep power-down mode.
Headers	R_SPI_FLASH_m45pe.h
Declaration	error_t R_SPI_FLASH_DeepPDown (uint8_t DevNo)
Explanation	<ul style="list-style-type: none"> • Sets deep power-down mode.
Arguments	uint8_t DevNo ; Device number
Return value	<ul style="list-style-type: none"> • Returns the result of the processing.
	FLASH_OK ; Successful operation
	FLASH_ERR_PARAM ; Parameter error
	FLASH_ERR_HARD ; Hardware error
	FLASH_ERR_OTHER ; Other error
Remarks	None

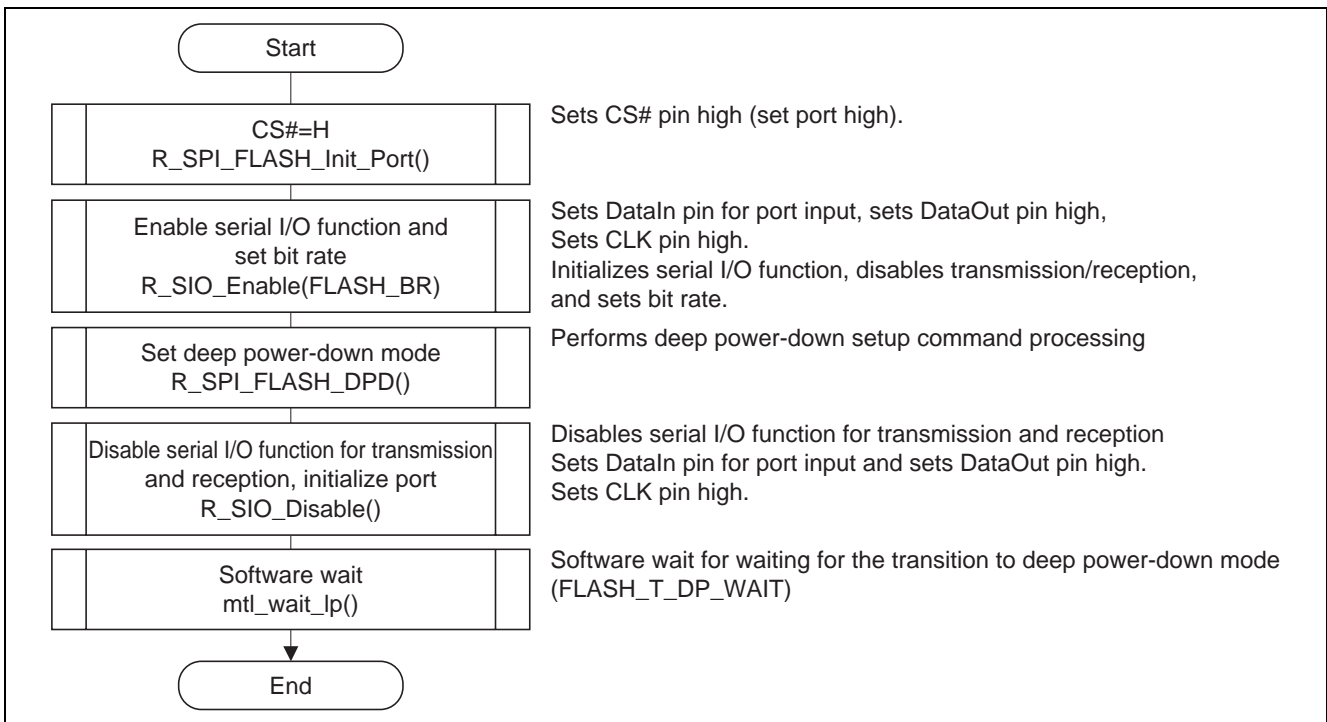


Figure 14 Deep Power-down Processing Outline

5.9.8 Deep Power-down Release Processing

R_SPI_FLASH_ReleaseDeepPDown

Synopsis	Releases deep power-down mode
Headers	R_SPI_FLASH_m45pe.h
Declaration	error_t R_SPI_FLASH_ReleaseDeepPDown (uint8_t DevNo)
Explanation	<ul style="list-style-type: none"> Releases deep power-down mode.
Arguments	uint8_t DevNo ; Device number uint8_t FAR* pData ; Pointer to read data buffer
Return value	<ul style="list-style-type: none"> Returns the result of the processing. FLASH_OK ; Successful operation FLASH_ERR_PARAM ; Parameter error FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error
Remarks	None

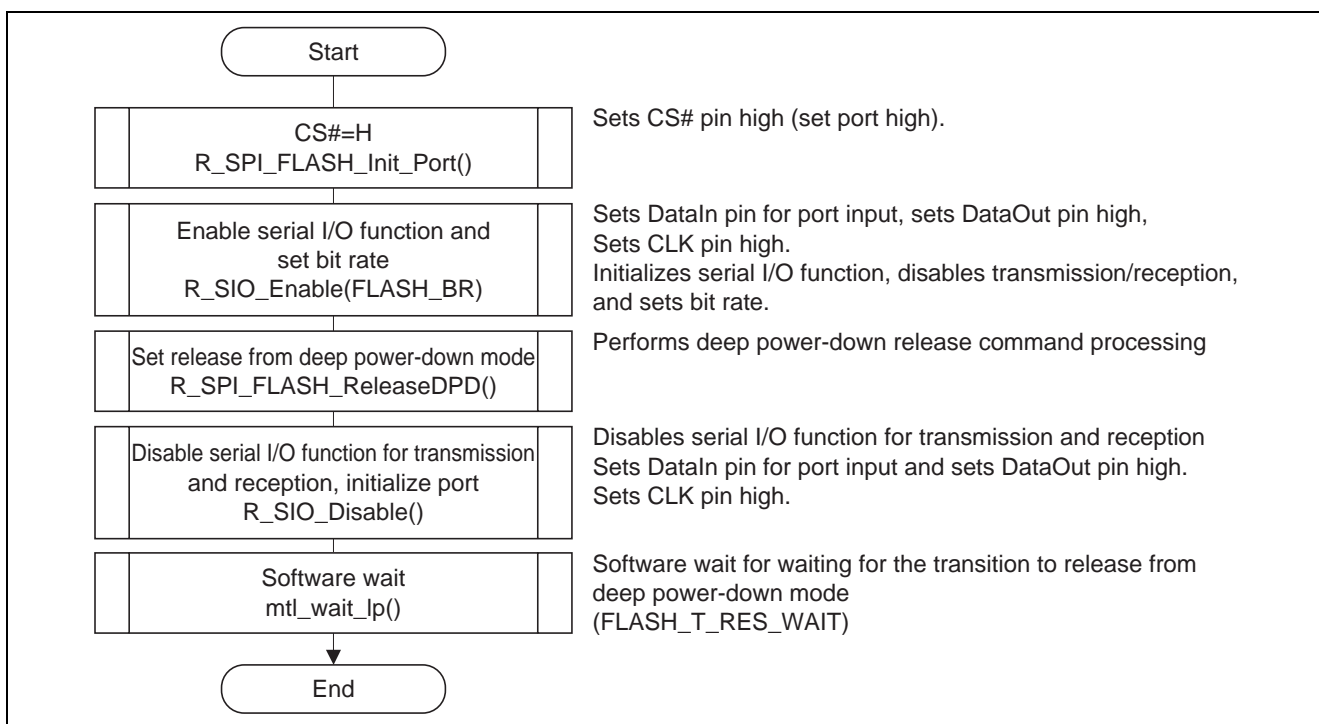


Figure 15 Deep Power-down Release Processing Outline

5.9.9 ID Read Processing

R_SPI_FLASH_ReadID

Synopsis	Reads ID.
Headers	R_SPI_FLASH_m45pe.h
Declaration	error_t R_SPI_FLASH_ReadID(uint8_t DevNo, uint8_t FAR* pData)
Explanation	<ul style="list-style-type: none"> The manufacturer ID and device ID are read out and stored in pData. Set up three bytes as a buffer for use in reading.
Arguments	uint8_t DevNo ; Device number uint8_t FAR* pData ; Pointer to read data buffer
Return value	<ul style="list-style-type: none"> Returns the result of the ID read. FLASH_OK ; Successful operation FLASH_ERR_PARAM ; Parameter error FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error
Remarks	<ul style="list-style-type: none"> This function is not supported (dependent on the device in use). Refer to the specifications of the flash memory in use.

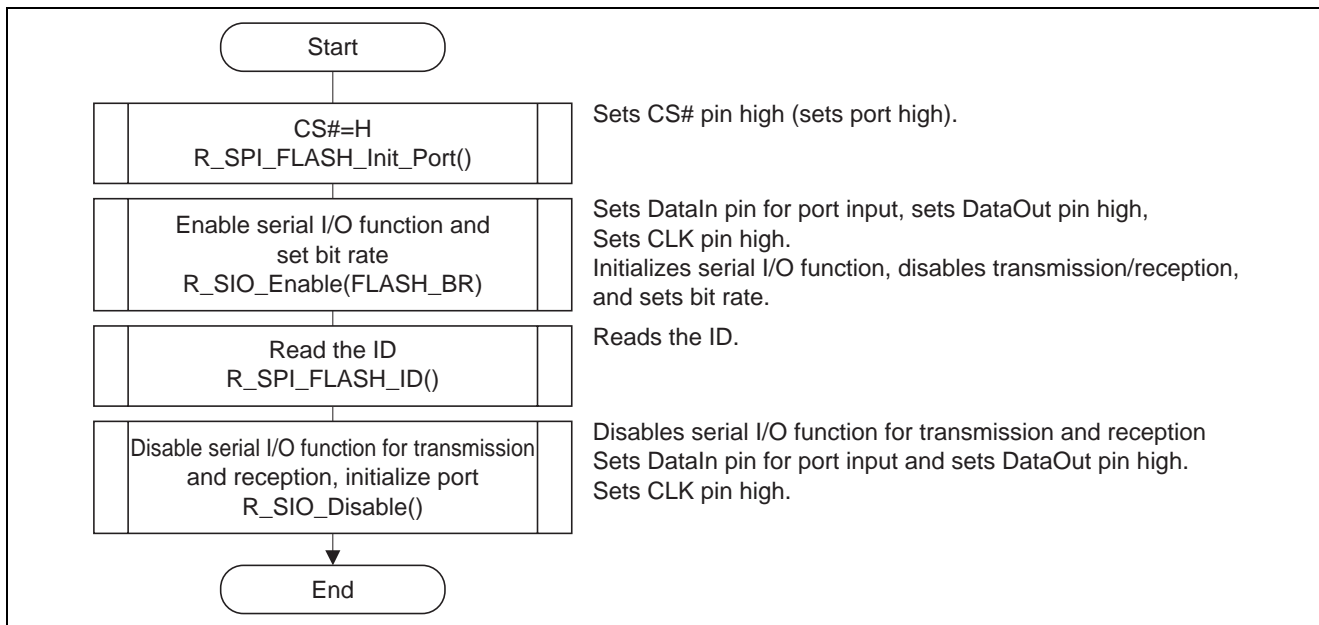


Figure 16 ID Read Processing Outline

5.9.10 Port Initialization Processing (Internal Function)

R_SPI_FLASH_Init_Port

Synopsis	Initializes ports (internal function).
Headers	R_SPI_FLASH_m45pe_io.h
Declaration	void R_SPI_FLASH_Init_Port(uint8_t DevNo)
Explanation	<ul style="list-style-type: none"> Initializes the port (CS#) of the specified device and sets it high.
Arguments	uint8_t DevNo ; Device number
Return value	void
Remarks	None

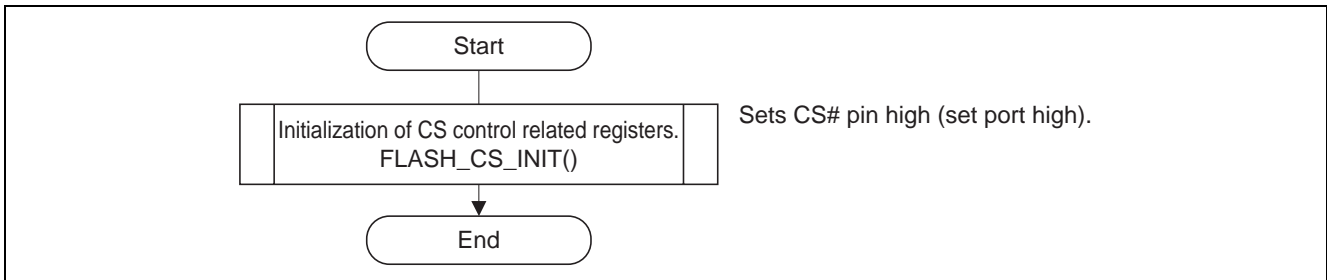


Figure 17 Port Initialization Processing Outline

5.9.11 **Command Send Processing (Internal Function)**

R_SPI_FLASH_Send_Cmd

Synopsis	Sends command (internal function).		
Headers	R_SPI_FLASH_m45pe_io.h		
Declaration	STATIC error_t R_SPI_FLASH_Send_Cmd(uint8_t Cmd, uint32_t Addr, uint8_t CmdSize)		
Explanation	<ul style="list-style-type: none"> • Sends the specified command. 		
Arguments	uint8_t	Cmd	; Command code
	uint32_t	Addr	; Address
	uint8_t	CmdSize	; Command size
Return value	<ul style="list-style-type: none"> • Returns the result of the command send. 		
	FLASH_OK		; Successful operation
	FLASH_ERR_HARD		; Hardware error
	FLASH_ERR_OTHER		; Other error
Remarks	None		

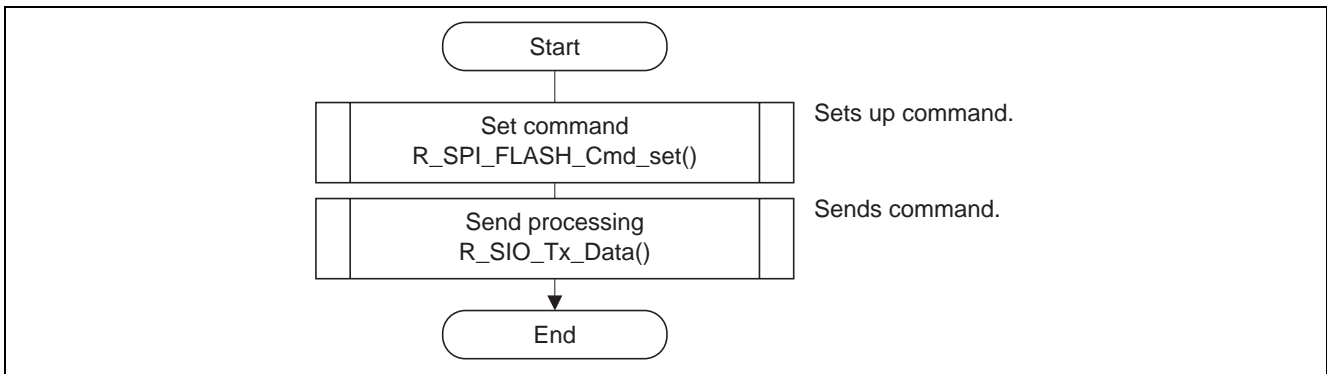


Figure 18 Command Send Processing Outline

5.9.12 Write Enable Command Processing (Internal Function)

R_SPI_FLASH_Write_En

Synopsis	Performs Write Enable command processing (internal function).
Headers	R_SPI_FLASH_m45pe_io.h
Declaration	STATIC error_t R_SPI_FLASH_Write_En(uint8_t DevNo)
Explanation	<ul style="list-style-type: none"> Sends a WREN command to enable the specified device for writes (setting the WEL bit).
Arguments	uint8_t DevNo ; Device number
Return value	<ul style="list-style-type: none"> Returns the result of command processing.
	FLASH_OK ; Successful operation
	FLASH_ERR_HARD ; Hardware error
	FLASH_ERR_OTHER ; Other error
Remarks	None

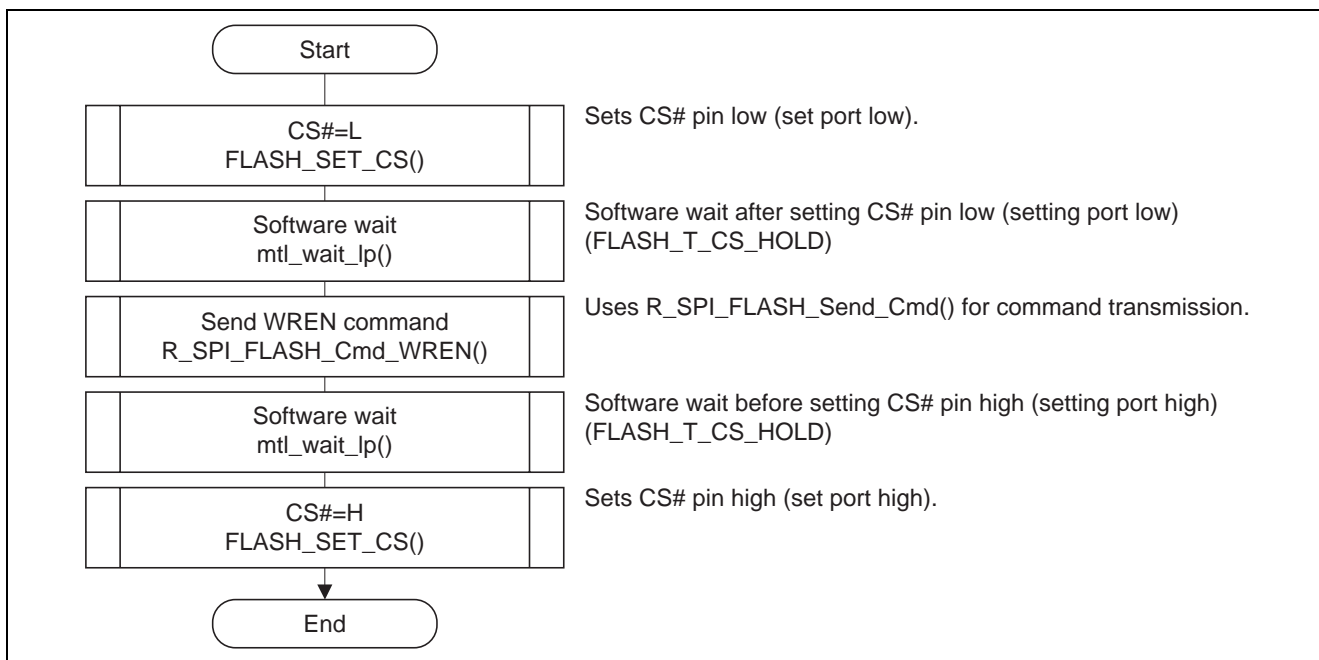


Figure 19 Write Enable Command Processing Outline

5.9.13 Write Disable Command Processing (Internal Function)

R_SPI_FLASH_Write_Di

Synopsis	Performs Write Disable command processing (internal function).
Headers	R_SPI_FLASH_m45pe_io.h
Declaration	STATIC error_t R_SPI_FLASH_Write_Di(uint8_t DevNo)
Explanation	<ul style="list-style-type: none"> Sends a WRDI command to disable the specified device for writes (clearing the WEL bit).
Arguments	uint8_t DevNo ; Device number
Return value	<ul style="list-style-type: none"> Returns the result of command processing.
	FLASH_OK ; Successful operation
	FLASH_ERR_HARD ; Hardware error
	FLASH_ERR_OTHER ; Other error
Remarks	None

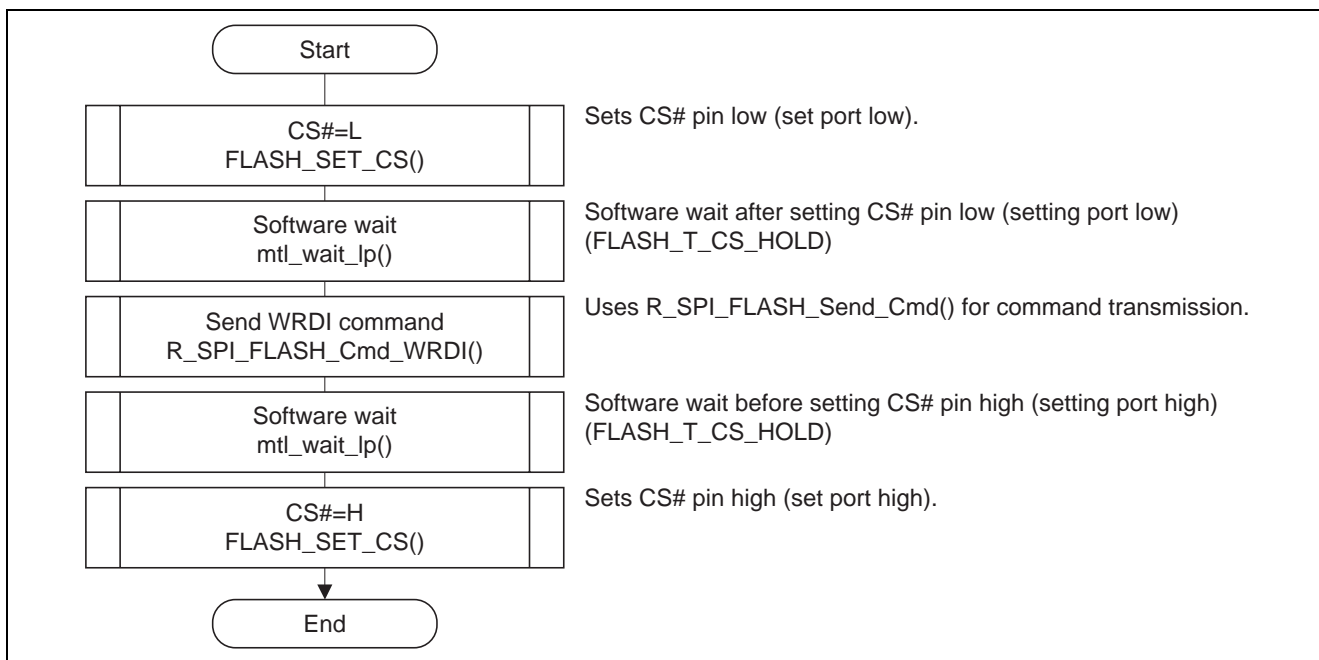


Figure 20 Write Disable Command Processing Outline

5.9.14 Read Status Register Command Processing (Internal Function)

R_SPI_FLASH_Read_StsReg

Synopsis	Performs Read Status Register command processing (internal function).
Headers	R_SPI_FLASH_m45pe_io.h
Declaration	error_t R_SPI_FLASH_Read_StsReg(uint8_t DevNo, uint8_t FAR* pStsReg)
Explanation	<ul style="list-style-type: none"> • Sends an RDSR command to read the data from the status register into pStsReg. Set up a 1-byte read buffer. • The read status buffer (pStatus) is loaded with the following information: <ul style="list-style-type: none"> Bit 7 to 2:Reserved (All "0") Bit 1:WEL 1: Internal Write Enable Latch is set 0: Internal Write Enable Latch is reset Bit 0:WIP 1: Program or Erase cycle is in progress 0: No Program or Erase cycle is in progress
Arguments	uint8_t DevNo ; Device number uint8_t FAR* pStsReg ; Pointer to buffer for storing read status
Return value	<ul style="list-style-type: none"> • Returns the result of command processing. FLASH_OK ; Successful operation FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error
Remarks	None

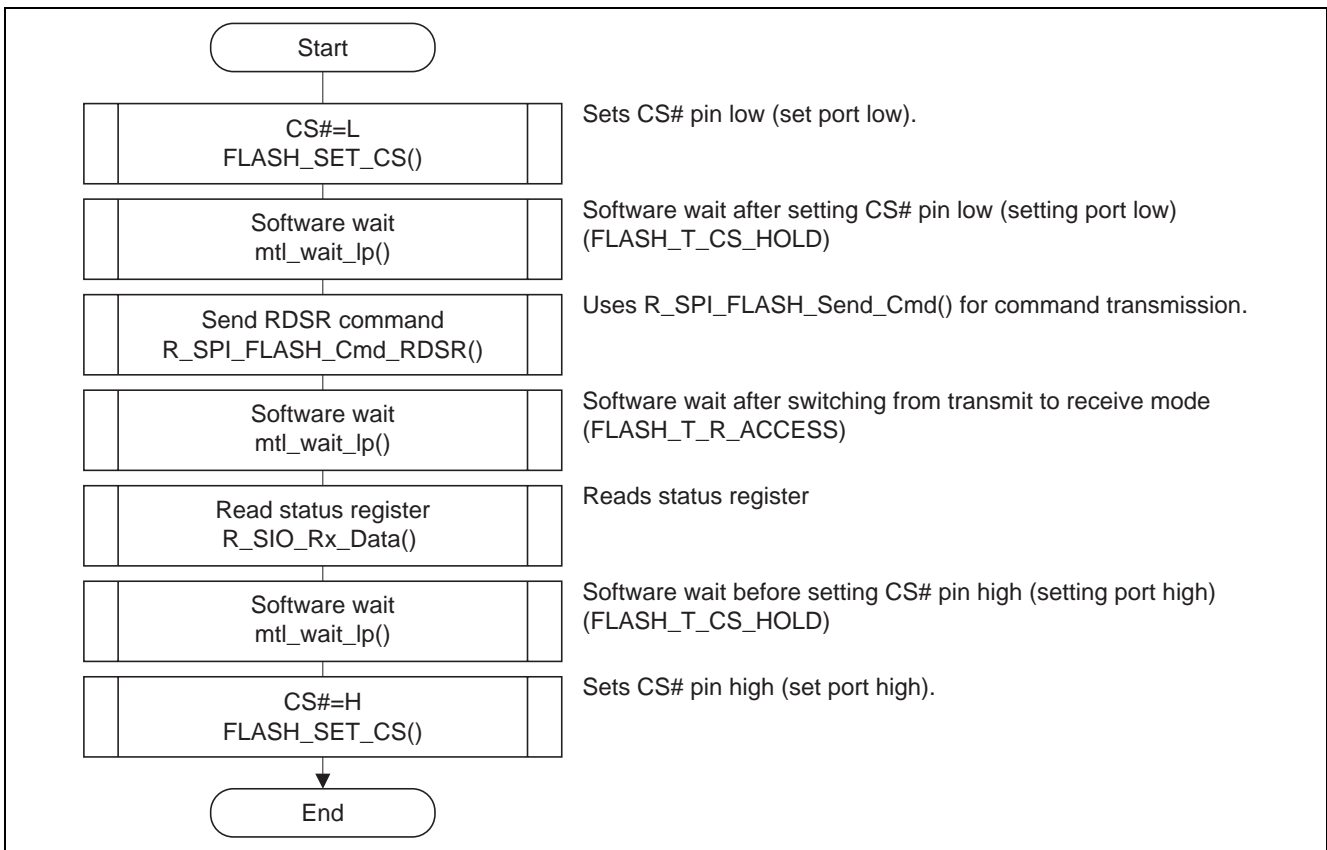


Figure 21 Read Status Register Command Processing Outline

5.9.15 Busy Wait Processing (Internal Function)

R_SPI_FLASH_Wait_Busy

Synopsis	Performs busy wait processing (internal function).
Headers	R_SPI_FLASH_m45pe_io.h
Declaration	STATIC error_t R_SPI_FLASH_Wait_Busy(uint8_t DevNo, uint16_t BusyTime, uint16_t BusyCnt)
Explanation	<ul style="list-style-type: none"> • Waits for the busy period at the BusyTime intervals if BusyCnt is found to be 0 with the Read Status Register command. • Waits for the busy period at the BusyTime intervals the number of times equal to BusyCnt if BusyCnt is found to be nonzero with the Read Status Register command.
Arguments	uint8_t DevNo ; Device number uint16_t BusyTime ; Wait time for status check uint16_t BusyCnt ; Counter
Return value	<ul style="list-style-type: none"> • Returns the result of wait processing. FLASH_OK ; Successful operation FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error
Remarks	None

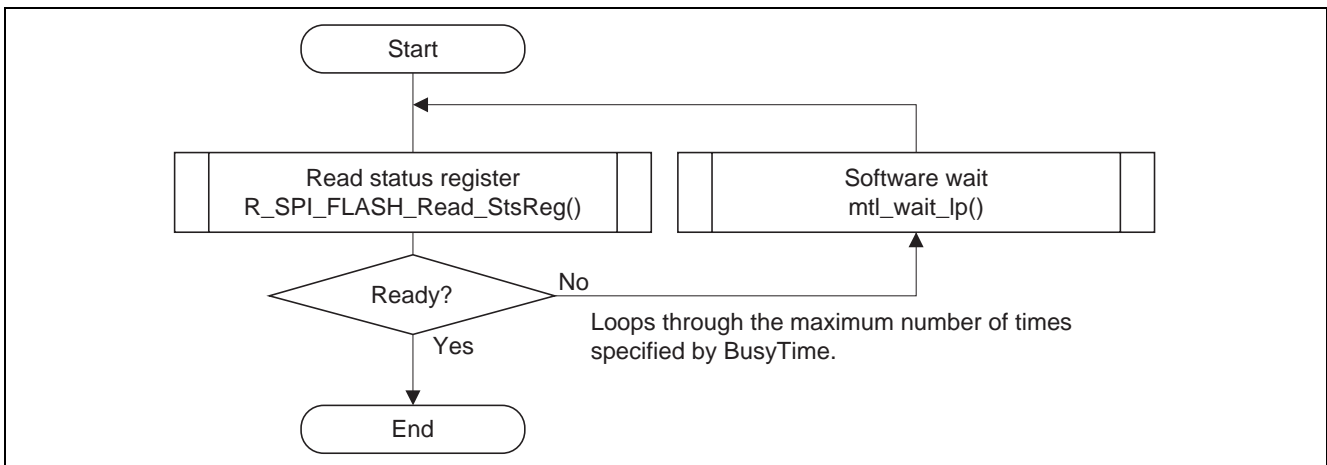


Figure 22 Busy Wait Processing Outline

5.9.16 Page Program Command Processing (Internal Function)

R_SPI_FLASH_Wait_Busy

Synopsis	Performs Page Program Command processing (internal function).
Headers	R_SPI_FLASH_m45pe_io.h
Declaration	STATIC error_t R_SPI_FLASH_Wait_Busy(uint8_t DevNo, uint16_t BusyTime, uint16_t BusyCnt)
Explanation	<ul style="list-style-type: none"> • Writes the specified number of bytes from pData into flash memory using the PP command starting at the specified address.
Arguments	<pre>uint8_t DevNo ; Device number uint32_t WAddr ; Write start address uint32_t WCnt ; Number of bytes to write uint8_t FAR* pData ; Pointer to write data buffer</pre>
Return value	<ul style="list-style-type: none"> • Returns the result of the write processing. <pre>FLASH_OK ; Successful operation FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error</pre>
Remarks	<ul style="list-style-type: none"> • Writes beyond a page are not permitted.

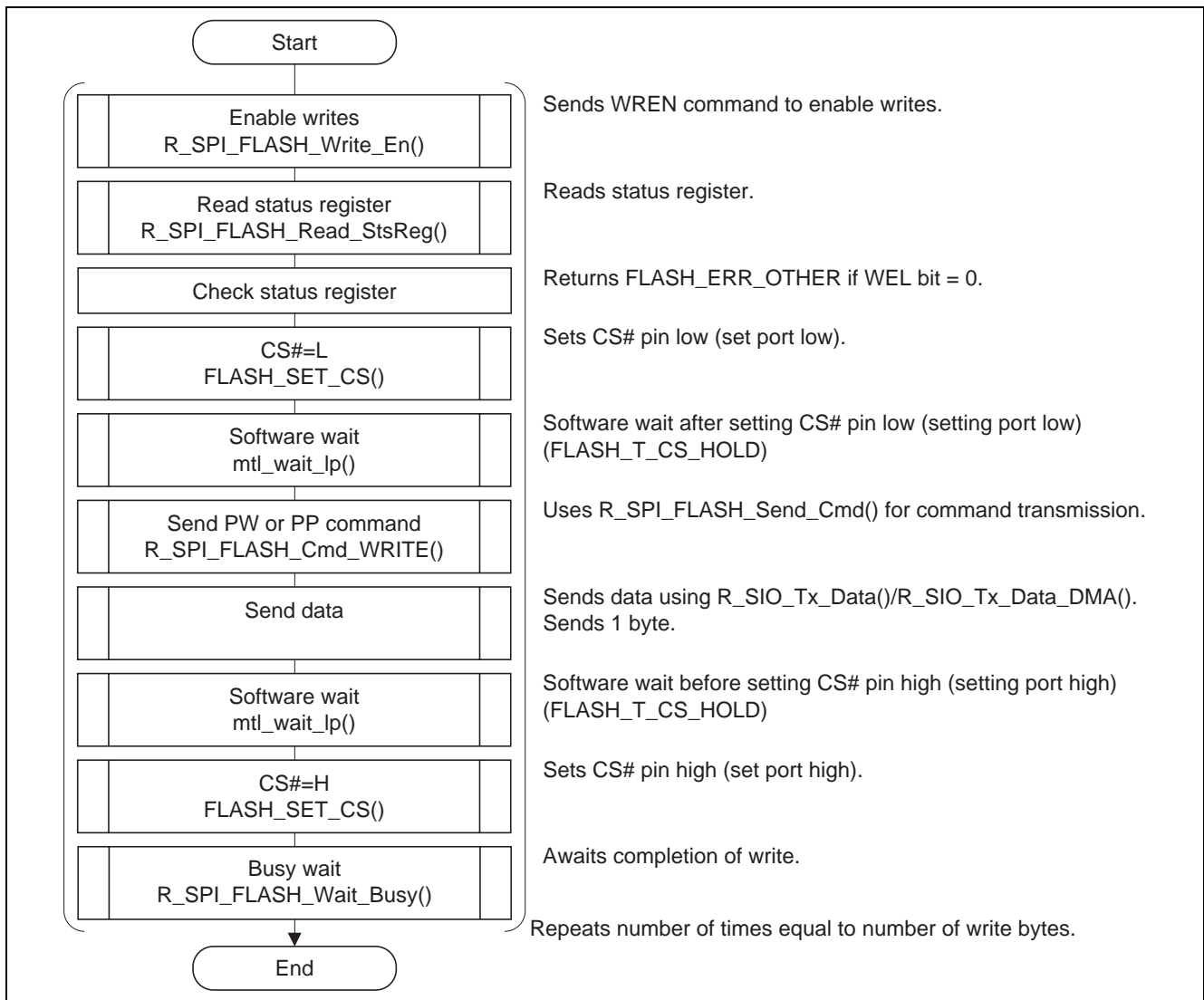


Figure 23 Page Program Command Processing Outline

5.9.17 **Read Command Processing (Internal Function)**

R_SPI_FLASH_Read_Memory

Synopsis	Performs Read command processing (internal function).		
Headers	R_SPI_FLASH_m45pe_io.h		
Declaration	error_t R_SPI_FLASH_Read_Memory(uint8_t DevNo, uint32_t RAddr, uint32_t RCnt, uint8_t FAR* pData)		
Explanation	<ul style="list-style-type: none"> • Reads the specified number of bytes from flash memory starting at the specified address, 1 byte at a time, using the READ command and stores the read data in pData. 		
Arguments	uint8_t	DevNo	; Device number
	uint32_t	RAddr	; Read start address
	uint32_t	RCnt	; Number of read bytes
	uint8_t FAR*	pData	; Pointer to read data buffer
Return value	<ul style="list-style-type: none"> • Returns the result of command processing. FLASH_OK ; Successful operation FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error		
Remarks	<ul style="list-style-type: none"> • The highest read address is flash memory size - 1. 		

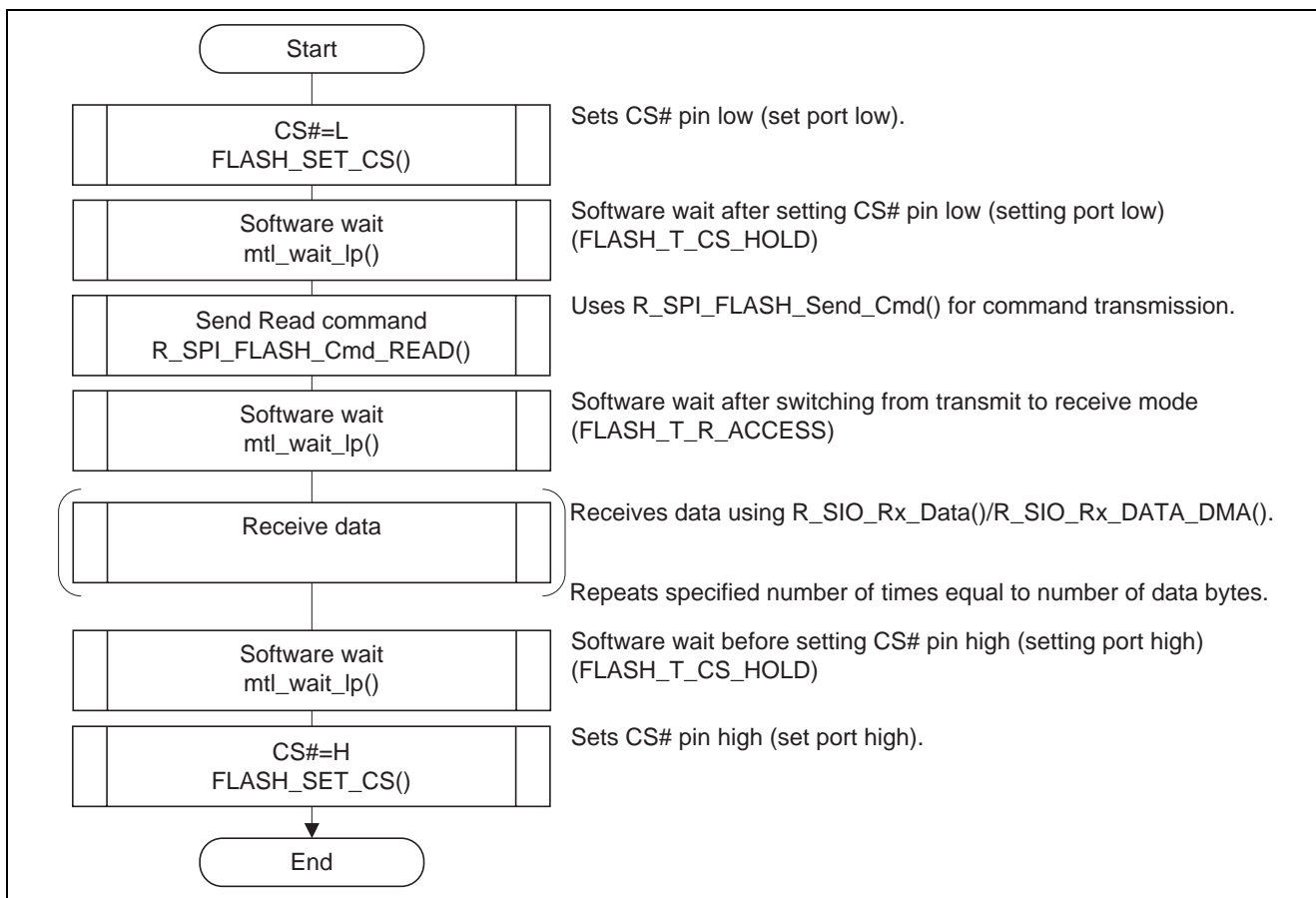


Figure 24 Read Command Processing Outline

5.9.18 Erase Command Processing (Internal Function)

R_SPI_FLASH_Erase

Synopsis	Performs Erase command processing (internal function)
Headers	R_SPI_FLASH_m45pe_io.h
Declaration	error_t R_SPI_FLASH_Erase(uint8_t DevNo, uint32_t EAddr, uint8_t Etype)
Explanation	<ul style="list-style-type: none"> • Erases the memory array by a Chip-Erase, Block-Erase, or Sector-Erase command. • Specify the following information for an erasure type (Etype). <ul style="list-style-type: none"> FLASH_P_ERASE: Page erasure FLASH_S_ERASE: Sector erasure
Arguments	uint8_t DevNo ; Device number uint32_t EAddr ; Erasure address uint32_t Etype ; Erasure type
Return value	<ul style="list-style-type: none"> • Returns the result of erase processing. FLASH_OK ; Successful operation FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error
Remarks	None

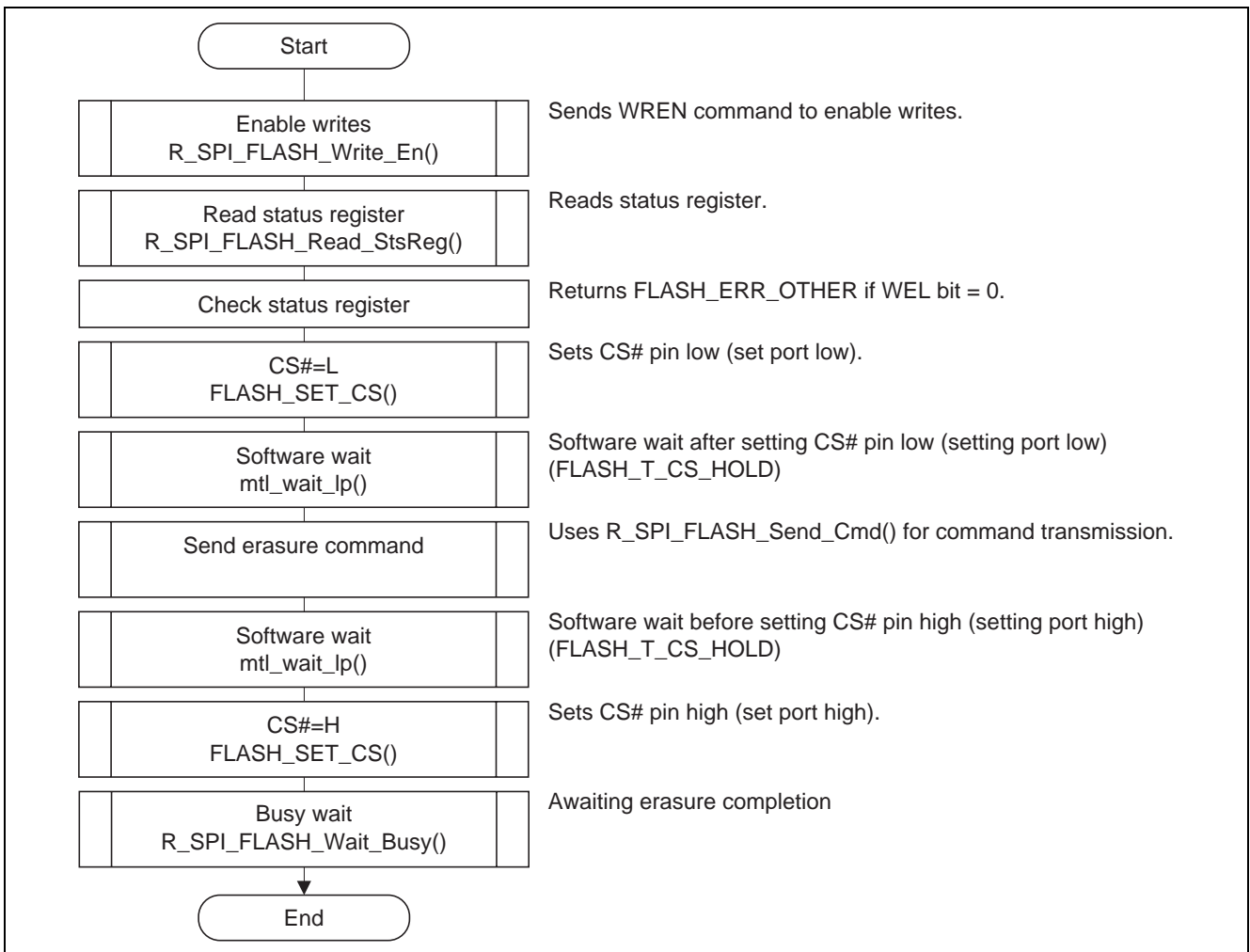


Figure 25 Erase Command Processing Outline

5.9.19 Deep Power-down Command Processing (Internal Function)

R_SPI_FLASH_DPD

Synopsis	Performs Deep Power-down command processing (internal function).
Headers	R_SPI_FLASH_m45pe_io.h
Declaration	error_t R_SPI_FLASH_DPD(uint8_t DevNo)
Explanation	<ul style="list-style-type: none"> • Sets deep power-down by using a DP command.
Arguments	uint8_t DevNo ; Device number uint8_t FAR* pData ; Pointer to read data buffer
Return value	<ul style="list-style-type: none"> • Returns the result of command processing. FLASH_OK ; Successful operation FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error
Remarks	None

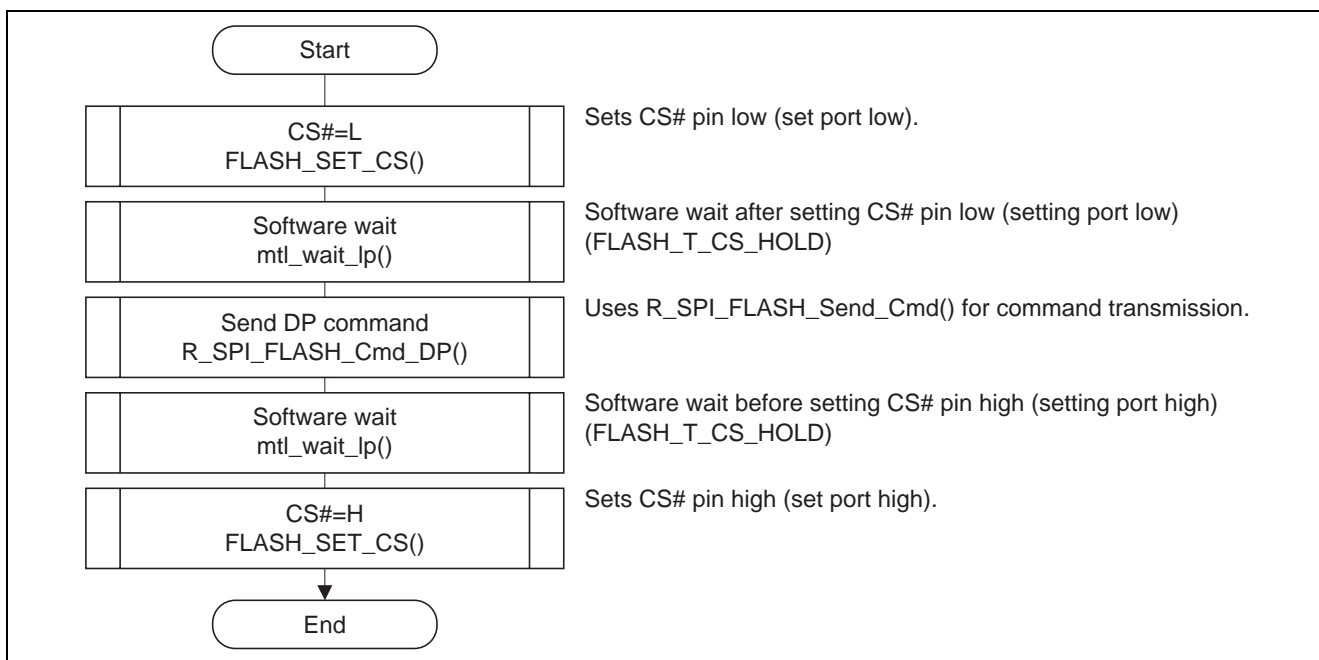


Figure 26 Deep Power-down Command Processing Outline

5.9.20 Deep Power-down Release Command Processing (Internal Function)

R_SPI_FLASH_ReleaseDPD

Synopsis	Performs Deep Power-down Release command processing (internal function).
Headers	R_SPI_FLASH_m45pe_io.h
Declaration	error_t R_SPI_FLASH_ReleaseDPD(uint8_t DevNo, uint8_t FAR* pData)
Explanation	<ul style="list-style-type: none"> Releases deep power-down by using a RES command, and then read out the signature and stores the result in pData. Set up a 1-byte read buffer.
Arguments	uint8_t DevNo ; Device number uint8_t FAR* pData ; Pointer to read data buffer
Return value	<ul style="list-style-type: none"> Returns the result of command processing. FLASH_OK ; Successful operation FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error
Remarks	None

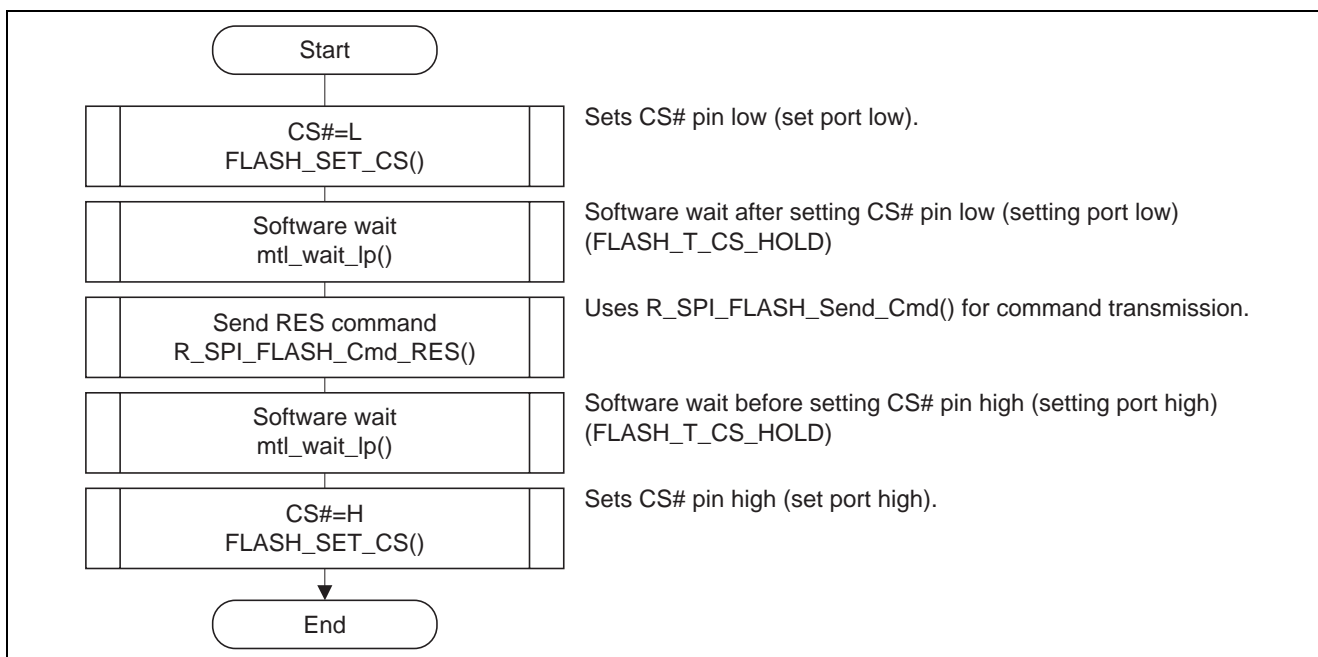


Figure 27 Deep Power-down Release Command Processing Outline

5.9.21 **Read ID Command Processing (Internal Function)**

R_SPI_FLASH_ID

Synopsis	Performs Read ID command processing (internal function).
Headers	R_SPI_FLASH_m45pe_io.h
Declaration	error_t R_SPI_FLASH_ID(uint8_t DevNo, uint8_t FAR* pData)
Explanation	<ul style="list-style-type: none"> This function uses the RDID command to read out the manufacturer ID and device ID, and stores the result in pData. Set up three bytes as a buffer for use in reading.
Arguments	uint8_t DevNo ; Device number uint8_t FAR* pData ; Pointer to read data buffer
Return value	<ul style="list-style-type: none"> Returns the result of command processing. FLASH_OK ; Successful operation FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error
Remarks	None

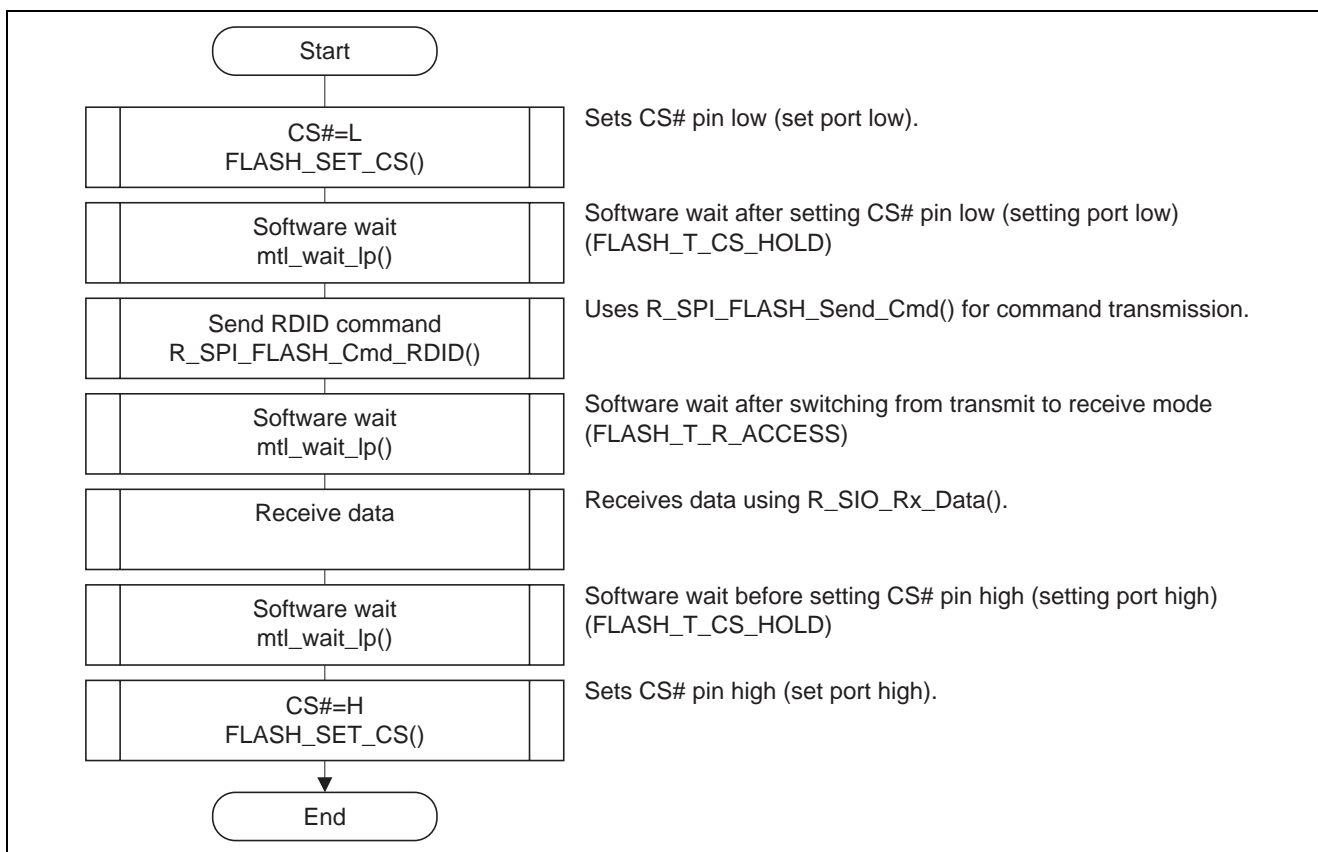


Figure 28 Read ID Command Processing Outline

5.9.22 **Command Setup Processing (Internal Function)**

R_SPI_FLASH_Cmd_set

Synopsis	Sets up command (internal function).		
Headers	R_SPI_FLASH_m45pe_io.h		
Declaration	STATIC void R_SPI_FLASH_Cmd_set(uint8_t Cmd, uint32_t Addr, uint8_t CmdSize)		
Explanation	<ul style="list-style-type: none"> • Sets a command and an address. Conversion is carried out according to the endian mode specified. 		
Arguments	uint8_t	Cmd	; Command (instruction code)
	uint32_t	Addr	; Address information
	uint8_t	CmdSize	; Command size
Return value	None		
Remarks	<ul style="list-style-type: none"> • The endian mode must be specified through MTL_MCU_LITTLE (mtl_com.h). 		

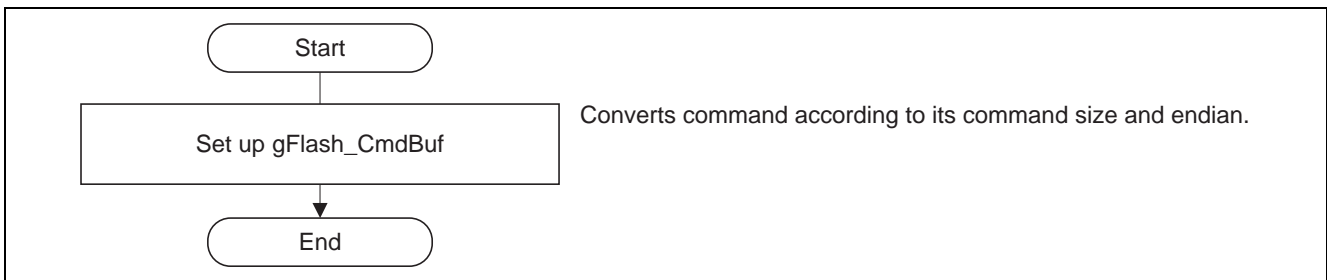


Figure 29 Command Setup Processing Outline

6. Application Example

This chapter gives an example of setting up the Serial Flash memory control section (the serial I/O control section is not covered).

For the serial I/O control section, refer to the application note for the individual MCU-specific clock synchronous single-master control software.

The baud rate is specified in this sample code because it is necessary to set it up according to the individual slave device.

The locations where settings are made are identified by the comments header `"/** SET **/"` in the defining file.

The common functions (e.g., `mtl_wait_lp()`) must be borrowed from the individual MCU-specific clock synchronous single-master control software.

6.1 Setting Up the Serial Flash memory Control Software

The settings to be made are identified by the comments header "/* SET */" in the individual file.

6.1.1 R_SPI_FLASH_m45pe.h

This is a definition file for this Serial Flash memory.

The settings to be made are identified by the comments header "/* SET */" in the file.

1. Defining the number of devices to be used and their device number

Specify the number of devices to be used and assign a device number to each of them.

Given below is an example of using one device and assigning a device number of 0.

A maximum of 2 devices can be controlled.

```

/*-----*/
/* Define number of required serial FLASH devices.(1~N devices) */
/* Define the device number in accordance with the number of serial FLASH
devices */
/* to be connected. */
/*-----*/
/* Define no. of devices */
#define FLASH_DEV_NUM 1 /* 1devices */

/* Define no. of slots */
#define FLASH_DEV0 0 /* Device 0 */
#define FLASH_DEV1 1 /* Device 1 */

```

2. Defining the size of the devices to be used

Specify the size of the devices to be used

Given below is an example of using 1 Mbit devices.

```

/*-----*/
/* Define the serial FLASH device. */
/*-----*/

#define M45PE10 /* 1Mbit ( 128kByte) */

```

6.1.2 R_SPI_FLASH_sfr.h

R_SPI_FLASH_sfr.h.XXX are made available for the purpose of evaluating the individual MCUs. Rename one of them to R_SPI_FLASH_sfr.h. If no header file is available that applies to the desired MCU, create your own R_SPI_FLASH_sfr.h while referring to these files.

The settings to be made are identified by the comments header "/* SET */" in the file.

1. Write Command Type Setting

Define the page write command (0x0A) if you intend to use it. Use the page program command (0x02) if the page write command is not defined.

For details of their difference, see the datasheet for the Serial Flash memory.

The page write command (0x0A) is used in the example below.

```

/*-----*/
/*  Select the write command type.                                */
/*  If you want to set the page write command(0x0A),            */
/*  define FLASH_0A_WRITE_SELECT                                */
/*  Otherwise page program command(0x02) is select.            */
/*-----*/

#define FLASH_0A_WRITE_SELECT

```

2. Setting up the Chip Select signal

Define the port to be used for the Chip Select signal.

When connecting a second device, define the second port.

Given below is a sample code for using port66.

```

/*-----*/
/*  Define the CS port.                                          */
/*-----*/
#define FLASH_DR_CS0      PORT6.DR.BIT.B6 /* FLASH CS0(Negative-true logic) */
#define FLASH_DDR_CS0    PORT6.DDR.BIT.B6 /* FLASH CS0(Negative-true logic) */

#if (FLASH_DEV_NUM > 1)
#define FLASH_DR_CS1      /* FLASH CS1(Negative-true logic) */
#define FLASH_DDR_CS1    /* FLASH CS1(Negative-true logic) */
#endif /* #if (FLASH_DEV_NUM > 1) */

```

3. Setting the baud rate

Set the baud rate in bits per second (bps).

The value to be set is dependent on the type of MCU and serial I/O to be used.

Given below is a sample code for the RX610's SCI, 3.125 Mbps (bits/second).

```
/*-----*/
/* Define the value of the bit rate register according to a communication
   baud rate. */
/* The possible maximum transfer frequency of CLK is depends on hardware
   circuit */
/* and MCU conditions. */
/* Refer to MCU hardware manual/memory card specifications and specify the
   buad rate. */

/* PCLK = 50MHz, n=0 for RX610 SCI */
#define FLASH_BR (uint8_t)0x03 /* BRR initial setting */
/* ++----- 3.125MHz */
```

Refer to the individual MCU hardware manual for the legitimate values.

6.1.3 R_SPI_FLASH_m45pe_io.h

The settings to be made are identified by the comments header "/* SET */" in the file.

1. Setting the value for time-out of erasure

Since times taken by erasure differ according to the device, reconsider the values used below as required.

The example below is written with 40 s as the period for time-out of erasure.

```
/*-----*/
/* Define the software timer value of erase busy waiting. */
/* If you want to wait till the flash comes to ready status without time out,
*/
/* comment the definition FLASH_EBUSY_WAIT_TIME. */
/*-----*/

#define FLASH_EBUSY_WAIT (uint16_t)40000 /* Erase busy waiting time 40000*
1ms = 40s */
```

2. Setting the value for time-out of writing

Since times taken by writing differ according to the device, reconsider the values used below as required.

The example below is written with 18 ms as the period for time-out of writing.

```
/* Transmit&receive waiting time */
#define FLASH_WBUSY_WAIT(uint16_t)18000 /* Write busy waiting time 18000*
1us = 18ms */
```


7. Usage Notes

7.1 Usage Notes to be Observed when Building the Sample Code

Include R_SPI_FLASH_m45pe.h when building this sample code into your application.

7.2 When Using a Cache-incorporated MCU

Specify a non-cache area for the buffer that is to be used for storing read/write data.

7.3 When Working with a Different Capacity within the Same Series

Reconsider the following definitions if you need to work with a different capacity but within the same series.

```
FLASH_MEM_SIZE  
FLASH_SECT_ADDR  
FLASH_WPAG_ADDR  
FLASH_ADDR_SIZE
```

Since there is a chance that definitions other than the above will also require reconsideration, obtain the datasheet for the memory you are using and adjust definitions as required.

7.4 When Using Other Types of Slave Devices

The sample code can control other slave devices on the same SPI bus.

Refer to this sample code when preparing slave device control programs for such devices.

Note that it is possible to set different baud rates for different slave device control programs.

Website and Support

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/inquiry>

All trademarks and registered trademarks are the property of their respective owners.

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	May 25, 2011	—	First edition issued
1.01	Aug 31, 2011	All	Header changed “RX Family” to “MCU”.
		1	Added “78K0R/KE3-L” in Target Device.
		3	Added “78K0R/Kx3-L” in Conditions for Checking the Operation.
		4	Added an application note for 78K0R/Kx3-L in Related Application Notes.
		12	Added the mention of “Maximum user stack size” in Table 6 Note.
		12	Added “78K0R/Kx3-L” in Sizes of Required Memory.
		13	Undated Application Note file name.
		13	Added R_SPI_FLASH_sfr.78k0r file.
48	Added “R_SPI_FLASH_m45pe_io.c”.		

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins

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

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

2. Processing at Power-on

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

- The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.

In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

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

4. Clock Signals

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

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

5. Differences between Products

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

- The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
 2. Renesas Electronics does not assume any liability for infringement of 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. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
 3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
 4. 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
 5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may 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.
 6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
 7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
Standard: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
High Quality: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
Specific: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
 8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
 9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or system 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
 11. This document may not be 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, or if you have any other inquiries.
- (Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.
(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.



SALES OFFICES

Renesas Electronics Corporation

<http://www.renesas.com>

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

Renesas Electronics America Inc.

2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited

1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited

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

Renesas Electronics Europe GmbH

Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.

7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.

Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited

Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852-2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.

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

Renesas Electronics Singapore Pte. Ltd.

1 HarbourFront Avenue, #06-10, Keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.

Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.

11F., Samik Laviel' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141