To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

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.

M16C Family

Control the Serial Flash of STMicroelectronics Using Clock Synchronous Serial I/O

Introduction

This document should be used for reference when implementing control of the M25P Series serial Flash manufactured STMicroelectronics, using the clock synchronous serial I/O of the M16C family manufactured by Renesas Technology Corp.

The M16C family incorporates a clock synchronous serial I/O. The M25P Series serial Flash can be controlled through the clock synchronous serial I/O and software.

This document describes sample programs for controlling the M25P Series serial Flash by using the clock synchronous serial I/O.

Target Device

The application examples described in this document are applicable when the following MCU and condition are used.

- MCU : M16C family
- Condition : Clock synchronous serial I/O is used
- Software Version : Ver.1.01

The programs can be executed by any M16C family MCU with the serial I/O. Note however that since some functions may be altered by function addition, etc., the functions should be confirmed against the MCU manual.

Be sure to perform evaluation sufficiently when using this application note.

Contents

1. Control Method for M25P Series Serial Flash

1.1 Overview of Operation

Control of the M25P Series serial Flash is implemented by using the clock synchronous Serial I/O in the M16C.

The sample programs execute the following control operations.

- Connects the S# pin of the serial Flash to a M16C port and controls it using output of the M16C general port.
- Controls data input/output by the clock synchronous serial I/O (using the internal clock).

  Assign the clock synchronous serial I/O pins for which CMOS output is possible and set the CMOS output to them, in order to implement the high-speed operation.
  In order to control the data transmission, the empty of transmit buffer is detected and interrupt is not used but transmit interrupt request bit is used.
  Therefore the register setting related to interrupt is described below.
  — Set the interrupt priority level to 000b (Level 0; Interrupt disable).
  — Set the transmit interrupt cause select bit to 0 (No data present in transmit buffer). (Set the DMA request cause to UART transmit interrupt request if DMA is used.)
- Control data transmission using DMAC as option.

Refer to the data sheets of the MCU and serial Flash and specify a usable clock frequency.

The connection method is described below.

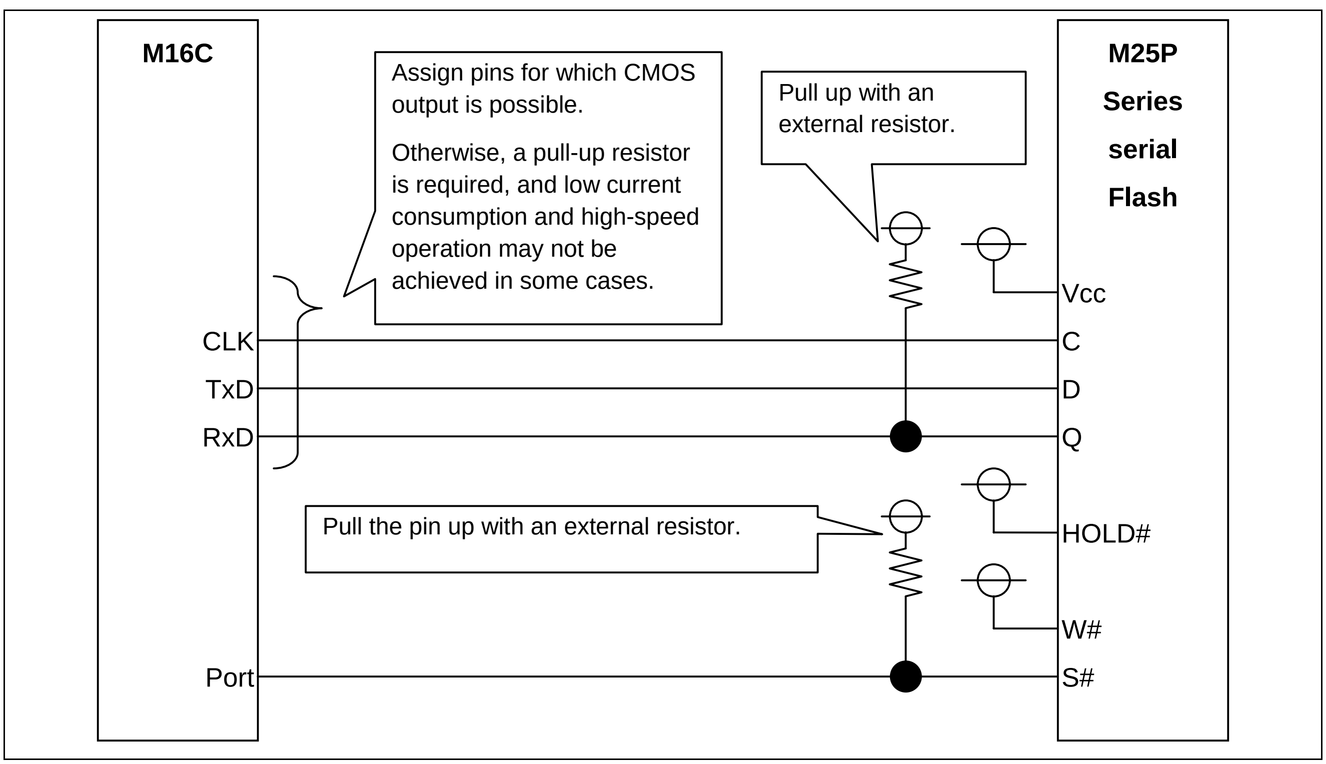
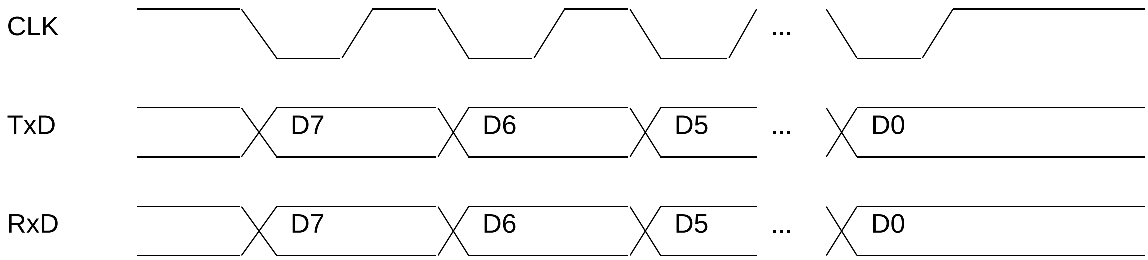


Figure 1.1 Serial Flash Connection Example

1.2 Signal Timing Generation of Clock Synchronous Serial I/O

Signals are generated at the following timing to satisfy the serial Flash timing.



- Transmission from MCU to serial Flash: Transmit data output at fall of transfer clock
- Reception from serial Flash to MCU: Receive data input at rise of transfer clock
- Transfer in MSB-first

The CLK pin level is high when transfer is not taking place.

Figure 1.2 Timing for Clock Synchronous Serial I/O of M16C

Check the data sheets of the MCU and serial Flash for the maximum clock frequency that can be used.

1.3 Control of S# Pin of Serial Flash

The S# pin of the serial Flash is connected to a M16C port and controlled using output of the M16C general port.

The period from the falling edge of the S# pin (port of M16C) of the serial Flash to the falling edge of the C pin (CLK of M16C) is controlled by inserting software wait cycles.

The period from the rising edge of the C pin (CLK of M16C) to the rising edge of the S# pin (port of M16C) is controlled by inserting software wait cycles.

Check the data sheet of the serial Flash and set the software wait time according to the system.

1.4 Processing after function operating

When function processing is begun, S# pin (Port of M16C) of Flash is set to high level first by setting the port function, and, next, C pin (CLK of M16C) of Flash is set to high level. Next, Serial I/O function is enabled and clock synchronous I/O mode is set. Command code etc. are output using serial I/O function after S# pin (Port of M16C) of Flash is set to low level.

After function processing is finished, S# pin (Port of M16C) of Flash is set to high level first and, next, Serial I/O function is disabled. Then the function is changed to general port, and Port/CLK/TxD pins are set to high level.

1.5 MCU Hardware Resources in Use

The hardware resources to be used are shown below.

In order to control the data transmission, the empty of transmit buffer is detected and interrupt is not used but transmit interrupt request bit is used.

Therefore the register setting related to interrupt is described below.

— Set the interrupt priority level to 000b (Level 0; Interrupt disable).
— Set the transmit interrupt cause select bit to 0 (No data present in transmit buffer). (Set the DMA request cause to UART transmit interrupt request if DMA is used.)

Table 1.1 Hardware Resources in Use

Resource in Use	Number of Used Resources
Clock synchronous serial I/O	One channel (essential)
Port (for control of the S# pin of serial Flash)	One port (essential)
DMAC	One channel (option)

The accessing mode between RAM and UART (transmit buffer or receive buffer) using DMAC is prepared as option.

1.6 M16C SFR (Peripheral Device Control Register) Setting - Clock Synchronous Serial I/O and Interrupt control Register

Set up the clock synchronous serial I/O as shown below to satisfy the serial Flash specifications/timing.

In order to control the data transmission, the empty of transmit buffer is detected and interrupt is not used but transmit interrupt request bit is used.

Therefore the register setting related to interrupt is described below.

- Set the interrupt priority level to 000b (Level 0; Interrupt disable).
- Set the transmit interrupt cause select bit to 0 (No data present in transmit buffer). (Set the DMA request cause to UART transmit interrupt request if DMA is used.)

1.6.1 M32C/87

An example of setting based on the register descriptions of (Table 17.2 Registers to Be Used and Setting in Clock Synchronous Serial I/O Mode) in the M32C/87 Group Hardware Manual Rev. 1.00 is shown in the table below.

Clock synchronous serial I/O other than UART2 (N channel open drain output) to be used are recommended.

Continuous receive mode should be disabled. The details please refer to the technical update TN-16C-A162A/J.

Table 1.2 Clock Synchronous Serial I/O Mode Settings

Register	Bit	Function and Setting
UiTB	7 to 0	Set the transmit data in these bits.
UiRB	7 to 0	The receive data is read from these bits.
	OER	Overrun error flag
UiBRG	7 to 0	Set the transfer speed in these bits. Clock frequency that can transfer data is different depending on the MCU.
UiMR	SMD2 to SMD0	Write 001b to these bits. (Clock synchronous serial I/O mode)
	CKDIR	Write 0 to this bit. (Internal clock) Set the clock frequency to UiBRG.
	IOPOL	Write 0 to this bit. (No reverse)
UiC0	CKS1, CKS0	Select the count source of UiBRG register in these bits.
	CRS	Write 0 to this bit. (This function is disabled because of CRD=1.)
	TXEPT	Transmit register empty flag (Read only)
	CRD	Write 1 to this bit. (CTS# and RTS# functions are disabled.)
	NCH	Write 0 to this bit. (CMOS output)
	CKPOL	Write 0 to this bit. Transmit data is output at falling edge of transfer clock and receive data is input at rising edge.
	UFORM	Write 1 to this bit. (MSB first)
UiC1	TE	0 is written to this bit at initialization. (Transmission disabled) Write 1 to this bit when transmission should be enabled.
	TI	Transmit buffer empty flag (Read only)
	RE	0 is written to this bit at initialization. (Reception disabled) Write 1 to this bit when reception should be enabled.
	RI	Receive complete flag (Read only)
	UiIRS	Write 0 to this bit at initialization. (No data present in UiTB register: TI=1)
	UiRRM	Write 0 to this bit. (Continuous receive mode is disabled.)
	UiLCH	Write 0 to this bit. (Data logic is not reversed.)
	SCLKSTPB	Write 0 to this bit.
UiSMR	7 to 0(Note 1)	Write 00 to these bits.
UiSMR2	7 to 0(Note 1)	Write 00 to these bits.
UiSMR3	7 to 0(Note 1)	Write 00 to these bits.
UiSMR4	7 to 0(Note 1)	Write 00 to these bits.

Note 1: Sample program doesn’t set 00 data to these registers because initial values of these registers after reset are 00.

The setting example of interrupt control register is shown in the table bellow.

In order to control the data transmission, the empty of transmit buffer is detected and interrupt is not used but transmit interrupt request bit is used.

Table 1.3 Interrupt Control Register Settings

Register	Bit	Function and Setting
SiTIC	ILVL2 to ILVL0	Write 000b to these bits. (Level 0: Interrupt is disabled.)
	IR	If this bit is 1, Interrupt is requested. Write 0 to this bit according to the needs.

1.6.2 M16C/62P

An example of setting based on the register descriptions of (Table 17.2 Registers to Be Used and Setting in Clock Synchronous Serial I/O Mode) in the M16C/62P Group Hardware Manual Rev. 2.41 is shown in the table below.

<u>Clock synchronous serial I/O other than UART2 (N channel open drain output) to be used are recommended.</u>

<u>Don't use SI/O3 and SI/O4.</u>

Table 1.4 Clock Synchronous Serial I/O Mode Settings

U2C1	TE	0 is written to this bit at initialization. (Transmission disabled) Write 1 to this bit when transmission should be enabled.
	TI	Transmit buffer empty flag (Read only)
	RE	0 is written to this bit at initialization. (Reception disabled) Write 1 to this bit when reception should be enabled.
	RI	Receive complete flag (Read only)
	U2IRS (Note1)	Write 0 to this bit at initialization. (No data present in transmit buffer: TI=1)
	U2RRM (Note1)	Write 0 to this bit at initialization. (Continuous receive mode is disabled.) Select UART2 continuous receive mode according to the usage.
	U2LCH	Write 0 to this bit. (Data logic is not reversed.)
	U2ERE	Write 0 to this bit. (Error signal output disabled)
UiSMR	7 to 0(Note 3)	Write 00 to these bits.
UiSMR2	7 to 0(Note 3)	Write 00 to these bits.
UiSMR3	7 to 0(Note 3)	Write 00 to these bits.
UiSMR4	7 to 0(Note 3)	Write 00 to these bits.
UCON	U0IRS	Set it as follows when UART0 is used. Write 0 to this bit at initialization. (No data present in transmit buffer: TI=1)
	U1IRS	Set it as follows when UART1 is used. Write 0 to this bit at initialization. (No data present in transmit buffer: TI=1)
	U0RRM (Note 2)	Set it as follows when UART0 is used. Write 0 to this bit at initialization. (Continuous receive mode is disabled.) Select UART0 continuous receive mode according to the usage.
	U1RRM (Note 2)	Set it as follows when UART1 is used. Write 0 to this bit at initialization. (Continuous receive mode is disabled.) Select UART1 continuous receive mode according to the usage.
	CLKMD0	Write 0 to this bit. (This function is disabled because of CLKMD1=1)
	CLKMD1	Write 0 to this bit. (CLK is output from only CLK1).
	RCSP	Write 0 to this bit. (This function is disabled because of CRD are disabled.)
	7	The read data is invalid. The write value should always be 0.

Note 1: Set it similarly to UCON (UART transmit and reception control register 2) for UART0 and UART1.
Note 2: Set it similarly to U2C1 (UART transmit and reception control register 1) for UART2.
Note 3: Sample program doesn't set 00 data to these registers because initial values of them after reset are 00.

The setting example of interrupt control register is shown in the table bellow.

In order to control the data transmission, the empty of transmit buffer is detected and interrupt is not used but transmit interrupt request bit is used.

Table 1.5 Interrupt Control Register Settings

Register	Bit	Function and Setting
SiTIC	ILVL2 to ILVL0	Write 000b to these bits. (Level 0: Interrupt is disabled.)
	IR	If this bit is 1, Interrupt is requested. Write 0 to this bit according to the needs.

1.6.3 M16C/30P

An example of setting based on the register descriptions of (Table 15.2 Registers to Be Used and Setting in Clock Synchronous Serial I/O Mode) in the M16C/30P Group Hardware Manual Rev. 1.11 is shown in the table below.

Clock synchronous serial I/O other than UART2 (N channel open drain output) to be used are recommended.

Table 1.6 Clock Synchronous Serial I/O Mode Settings

Register	Bit	Function and Setting
UiTB	7 to 0	Set the transmit data in these bits.
UiRB	7 to 0	The receive data is read from these bits.
	OER	Overrun error flag
UiBRG	7 to 0	Set the transfer speed in these bits. Clock frequency that can transfer data is different depending on the MCU.
UiMR	SMD2 to SMD0	Write 001b to these bits. (Clock synchronous serial I/O mode)
	CKDIR	Write 0 to this bit. (Internal clock) Set the clock frequency to UiBRG.
	IOPOL	Write 0 to this bit. (No reverse)
UiC0	CKS1, CKS0	Select the count source of UiBRG register in these bits.
	CRS	Write 0 to this bit. (This function is disabled because of CRD=1)
	TXEPT	Transmit register empty flag (Read only)
	CRD	Write 1 to this bit. (CTS# and RTS# functions are disabled.)
	NCH	Write 0 to this bit. (CMOS output)
	CKPOL	Write 0 to this bit. Transmit data is output at falling edge of transfer clock and receive data is input at rising edge.
	UFORM	Write 1 to this bit. (MSB first)
U0C1, U1C1	TE	0 is written to this bit at initialization. (Transmission disabled) Write 1 to this bit when transmission should be enabled.
	TI	Transmit buffer empty flag (Read only)
	RE	0 is written to this bit at initialization. (Reception disabled) Write 1 to this bit when reception should be enabled.
	RI	Receive complete flag (Read only)
	5 to 4	The read data are invalid. The write value should always be 0.
	U0LCH/U1LCH	Write 0 to this bit. (Data logic is not reversed.)
	U0ERE/U1ERE	Write 0 to this bit. (Error signal output disabled)
U2C1	TE	0 is written to this bit at initialization. (Transmission disabled) Write 1 to this bit when transmission should be enabled.
	TI	Transmit buffer empty flag (Read only)
	RE	0 is written to this bit at initialization. (Reception disabled) Write 1 to this bit when reception should be enabled.
	U2IRS (Note1)	Write 0 to this bit at initialization. (No data present in transmit buffer: TI=1)
	TI	Transmit buffer empty flag (Read only)
	U2RRM (Note1)	Write 0 to this bit at initialization. (Continuous receive mode is disabled.) Select UART2 continuous receive mode according to the usage.
	U2LCH	Write 0 to this bit. (Data logic is not reversed)
	U2ERE	Write 0 to this bit. (Error signal output disabled)

UiSMR	7 to 0(Note 3)	Write 00 to these bits.
UiSMR2	7 to 0(Note 3)	Write 00 to these bits.
UiSMR3	7 to 0(Note 3)	Write 00 to these bits.
UiSMR4	7 to 0(Note 3)	Write 00 to these bits.
UCON	U0IRS	Set it as follows when UART0 is used. Write 0 to this bit at initialization. (No data present in transmit buffer: TI=1)
	U1IRS	Set it as follows when UART1 is used. Write 0 to this bit at initialization. (No data present in transmit buffer: TI=1)
	U0RRM (Note 2)	Set it as follows when UART0 is used. Write 0 to this bit at initialization. (Continuous receive mode is disabled.) Select UART0 continuous receive mode according to the usage.
	U1RRM (Note 2)	Set it as follows when UART1 is used. Write 0 to this bit at initialization. (Continuous receive mode is disabled.) Select UART1 continuous receive mode according to the usage.
	CLKMD0	Write 0 to this bit. (This function is disabled because of CLKMD1=1.)
	CLKMD1	Write 0 to this bit. (CLK is output from only CLK1.)
	RCSP	Write 0 to this bit. (This function is disabled because of CRD are disabled.)
	7	The read data is invalid. The write value should always be 0.

Note 1: Set it similarly to UCON (UART transmit and reception control register 2) for UART0 and UART1.
Note 2: Set it similarly to U2C1 (UART transmit and reception control register 1) for UART2.
Note 3: Sample program doesn't set 00 data to these registers because initial values of them after reset are 00.

The setting example of interrupt control register is shown in the table bellow.

In order to control the data transmission, the empty of transmit buffer is detected and interrupt is not used but transmit interrupt request bit is used.

Table 1.7 Interrupt Control Register Settings

Register	Bit	Function and Setting
SiTIC	ILVL2 to ILVL0	Write 000b to these bits. (Level 0: Interrupt is disabled.)
	IR	If this bit is 1, Interrupt is requested. Write 0 to this bit according to the needs.

1.6.4 M16C/29

An example of setting based on the register descriptions of (Table 14.2 Registers to Be Used and Setting in Clock Synchronous Serial I/O Mode) in the M16C/29 Group Hardware Manual Rev. 1.00 is shown in the table below.

Don't use SI/O3 and SI/O4.

Table 1.8 Clock Synchronous Serial I/O Mode Settings

Register	Bit	Function and Setting
UiTB	7 to 0	Set the transmit data in these bits.
UiRB	7 to 0	The receive data is read from these bits.
	OER	Overrun error flag
UiBRG	7 to 0	Set the transfer speed in these bits. Clock frequency that can transfer data is different depending on the MCU.
UiMR	SMD2 to SMD0	Write 001b to these bits. (Clock synchronous serial I/O mode)
	CKDIR	Write 0 to this bit. (Internal clock) Set the clock frequency to UiBRG.
	7	Write 0 to this bit.
UiC0	CKS1, CKS0	Select the count source of UiBRG register in these bits.
	CRS	Write 0 to this bit. (This function is disabled because of CRD=1.)
	TXEPT	Transmit register empty flag (Read only)
	CRD	Write 1 to this bit. (CTS# and RTS# functions are disabled.)
	NCH	Write 0 to this bit. (CMOS output)
	CKPOL	Write 0 to this bit. Transmit data is output at falling edge of transfer clock and receive data is input at rising edge.
	UFORM	Write 1 to this bit. (MSB first)
U0C1, U1C1	TE	0 is written to this bit at initialization. (Transmission disabled) Write 1 to this bit when transmission should be enabled.
	TI	Transmit buffer empty flag (Read only)
	RE	0 is written to this bit at initialization. (Reception disabled) Write 1 to this bit when reception should be enabled.
	RI	Receive complete flag (Read only)
	7 to 4	These bits are always read as 0. The write value should always be 0.
U2C1	TE	0 is written to this bit at initialization. (Transmission disabled) Write 1 to this bit when transmission should be enabled.
	TI	Transmit buffer empty flag (Read only)
	RE	0 is written to this bit at initialization. (Reception disabled) Write 1 to this bit when reception should be enabled.
	RI	Receive complete flag (Read only)
	U2IRS (Note 1)	Write 0 to this bit at initialization. (No data present in transmit buffer: TI=1)
	U2RRM (Note 1)	Write 0 to this bit at initialization. (Continuous receive mode is disabled.) Select UART2 continuous receive mode according to the usage.
	U2LCH	Write 0 to this bit. (Data logic is not reversed)
	U2ERE	Write 0 to this bit. (Error signal output disabled)

U2SMR	7 to 0(Note 3)	Write 00 to these bits.
U2SMR2	7 to 0(Note 3)	Write 00 to these bits.
U2SMR3	7 to 0(Note 3)	Write 00 to these bits.
U2SMR4	7 to 0(Note 3)	Write 00 to these bits.
UCON	U0IRS	Set it as follows when UART0 is used. Write 0 to this bit at initialization. (No data present in transmit buffer: TI=1)
	U1IRS	Set it as follows when UART1 is used. Write 0 to this bit at initialization. (No data present in transmit buffer: TI=1)
	U0RRM (Note 2)	Set it as follows when UART0 is used. Write 0 to this bit at initialization. (Continuous receive mode is disabled.) Select UART0 continuous receive mode according to the usage.
	U1RRM (Note 2)	Set it as follows when UART1 is used. Write 0 to this bit at initialization. (Continuous receive mode is disabled.) Select UART1 continuous receive mode according to the usage.
	CLKMD0	Write 0 to this bit. (This function is disabled because of CLKMD1=1.)
	CLKMD1	Write 0 to this bit. (CLK is output from only CLK1.)
	RCSP	Write 0 to this bit. (This function is disabled because of CRD are disabled).
	7	The read data is invalid. The write value should always be 0.

Note 1: Set it similarly to UCON (UART transmit and reception control register 2) for UART0 and UART1.
Note 2: Set it similarly to U2C1 (UART transmit and reception control register 1) for UART2.
Note 3: Sample program doesn't set 00 data to these registers because initial values of them after reset are 00.

The setting example of interrupt control register is shown in the table bellow.

In order to control the data transmission, the empty of transmit buffer is detected and interrupt is not used but transmit interrupt request bit is used.

Table 1.9 Interrupt Control Register Settings

Register	Bit	Function and Setting
SiTIC	ILVL2 to ILVL0	Write 000b to these bits. (Level 0: Interrupt is disable)
	IR	If this bit is 1, Interrupt is requested. Write 0 to this bit according to the needs.

1.6.5 R8C/25

An example of setting based on the register descriptions of (Table 15.2 Registers to Be Used and Setting in Clock Synchronous Serial I/O Mode) in the R8C/25 Group Hardware Manual Rev. 2.00 is shown in the table below.

UART1can't be used, because it is not supported Clock synchronous.

Table 1.10 Clock Synchronous Serial I/O Mode Settings

Register	Bit	Function and Setting
UiTB	7 to 0	Set the transmit data in these bits.
UiRB	7 to 0	The receive data is read from these bits.
	OER	Overrun error flag
UiBRG	7 to 0	Set the transfer speed in these bits. Clock frequency that can transfer data is different depending on the MCU.
UiMR	SMD2 to SMD0	Write 001b to these bits. (Clock synchronous serial I/O mode)
	CKDIR	Write 0 to this bit. (Internal clock) Set the clock frequency to UiBRG.
	7	Write 0 to this bit.
UiC0	CKS1, CKS0	Select the count source of UiBRG register in these bits.
	2	Write 0 to this bit.
	TXEPT	Transmit register empty flag (Read only)
	4	This bit is always read as 0. The write value should always be 0.
	NCH	Write 0 to this bit. (CMOS output)
	CKPOL	Write 0 to this bit. Transmit data is output at falling edge of transfer clock and receive data is input at rising edge.
	UFORM	Write 1 to this bit. (MSB first)
UiC1	TE	0 is written to this bit at initialization. (Transmission disabled) Write 1 to this bit when transmission should be enabled.
	TI	Transmit buffer empty flag (Read only)
	RE	0 is written to this bit at initialization. (Reception disabled) Write 1 to this bit when reception should be enabled.
	RI	Receive complete flag (Read only)
	UiIRS	Write 0 to this bit at initialization. (No data present in transmit buffer: TI=1)
	UiRRM	Write 0 to this bit at initialization. (Continuous receive mode is disabled.) Select UARTi continuous receive mode according to the usage.
	7 to 6	These bits are always read as 0. The write value should always be 0.

The setting example of interrupt control register is shown in the table bellow.

In order to control the data transmission, the empty of transmit buffer is detected and interrupt is not used but transmit interrupt request bit is used.

Table 1.11 Interrupt Control Register Settings

Register	Bit	Function and Setting
SiTIC	ILVL2 to ILVL0	Write 000b to these bits. (Level 0: Interrupt is disabled.)
	IR	If this bit is 1, Interrupt is requested. Write 0 to this bit according to the needs.

1.7 M16C SFR (Peripheral Device Control Register) Setting - DMAC and Interrupt control Register

High-speed data transmission is possible using DMAC. The accessing mode between RAM and UART (transmit buffer or receive buffer) using DMAC is prepared as option.

1.7.1 M32C/87

Sample program doesn't support the DMA. Because continuous receive mode should be disabled. The details please refer to the technical update TN-16C-A162A/J.

1.7.2 M16C/62P

An example of setting based on the register descriptions in the M16C/62P Group Hardware Manual Rev. 2.41 is shown in the table below.

Table 1.14 DMAC Settings

Register	Bit	Function and Setting
UMiSL	DSEL3 to DSEL0	Select either UARTi transmit or UARTi receive according to the transfer mode. Write 0 to DMS bit because the factor is UART transmit or UART reception.
	5 to 4	These bits are always read as 0. The write value should always be 0.
	DMS	Write 0 to this bit because the factor is UART transmit or UART reception.
	DSR	Write 0 to this bit because software trigger is not used
DMiCON	DMBIT	Write 1 to this bit. (8 bit)
	DMASL	Write 0 to these bits. (Single transfer)
	DMAS	DMA request bit. Write 0 to this bit at initialization. (DMA Not requested)
	DMAE	Write 0 to this bit at initialization. (Disable) Write 1 to this bit when DMA is enabled
	DSD	0 is written to this bit at initialization. (Fixed) Select according to the source address
	DAD	0 is written to this bit at initialization. (Fixed) Select according to the destination address
	7 to 6	These bits are always read as 0. The write value should always be 0.
SARi	19 to 0	Set the source address of transfer.
	23 to 20	These bits are always read as 0. The write value should always be 0.
DARi	19 to0	Set the destination address of transfer.
	23 to 20	These bits are always read as 0. The write value should always be 0.
TCRi	15 to 0	Set the transfer count –1.

The setting example of interrupt control register is shown in the table bellow.

Table 1.15 Interrupt Control Register Settings

Register	Bit	Function and Setting
DMiIC	ILVL2 to ILVL0	Write 000b to these bits. (Level 0: Interrupt is disabled.)
	IR	If this bit is 1, Interrupt is requested. Write 0 to this bit according to the needs.

Note: UART1 is recommended not to use when DMA transfer is used.
When UART1 is in transmit state, DMA request factor select register is assigned to DMA0. When UART1 is in reception state, DMA request factor select register is assigned to DMA1. In order to DMA transfer is implemented, Both DMA0 and DMA1 have to be used and software has to be modified.

1.7.3 M16C/30P

An example of setting based on the register descriptions in the M16C/30P Group Hardware Manual Rev. 1.11 is shown in the table below.

Table 1.16 DMAC Settings

Register	Bit	Function and Setting
UMiSL	DSEL3 to DSEL0	Select either UARTi transmit or UARTi receive according to the transfer mode. Write 0 to DMS bit because the factor is UART transmit or UART reception.
	5 to 4	These bits are always read as 0. The write value should always be 0.
	DMS	Write 0 to this bit because the factor is UART transmit or UART reception.
	DSR	Write 0 to this bit because software trigger is not used
DMiCON	DMBIT	Write 1 to this bit. (8 bit)
	DMASL	Write 0 to these bits. (Single transfer)
	DMAS	DMA request bit. Write 0 to this bit at initialization. (DMA Not requested)
	DMAE	Write 0 to this bit at initialization. (Disable) Write 1 to this bit when DMA is enabled
	DSD	0 is written to this bit at initialization. (Fixed) Select according to the source address
	DAD	0 is written to this bit at initialization. (Fixed) Select according to the destination address
	7 to 6	These bits are always read as 0. The write value should always be 0.
SARi	19 to 0	Set the source address of transfer.
	23 to 20	These bits are always read as 0. The write value should always be 0.
DARi	19 to0	Set the destination address of transfer.
	23 to 20	These bits are always read as 0. The write value should always be 0.
TCRi	15 to 0	Set the transfer count –1.

The setting example of interrupt control register is shown in the table bellow.

Table 1.17 Interrupt Control Register Settings

Register	Bit	Function and Setting
DMiIC	ILVL2 to ILVL0	Write 000b to these bits. (Level 0: Interrupt is disable)
	IR	If this bit is 1, Interrupt is requested. Write 0 to this bit according to the needs.

Note: UART1 is recommended not to use when DMA transfer is used.
When UART1 is in transmit state, DMA request factor select register is assigned to DMA0. When UART1 is in reception state, DMA request factor select register is assigned to DMA1. In order to DMA transfer is implemented, Both DMA0 and DMA1 have to be used and software has to be modified.

1.7.4 M16C/29

An example of setting based on the register descriptions in the M16C/29 Group Hardware Manual Rev. 1.00 is shown in the table below.

Table 1.18 DMAC Settings

Register	Bit	Function and Setting
UMiSL	DSEL3 to DSEL0	Select either UARTi transmit or UARTi receive according to the transfer mode. Write 0 to DMS bit because the cause is UART transmit or UART reception.
	5 to 4	These bits are always read as 0. The write value should always be 0.
	DMS	Write 0 to this bit because the cause is UART transmit or UART reception.
	DSR	Write 0 to this bit because software trigger is not used
DMiCON	DMBIT	Write 1 to this bit. (8 bit)
	DMASL	Write 0 to these bits. (Single transfer)
	DMAS	DMA request bit. Write 0 to this bit at initialization. (DMA Not requested)
	DMAE	Write 0 to this bit at initialization. (Disable) Write 1 to this bit when DMA is enabled.
	DSD	0 is written to this bit at initialization. (Fixed) Select according to the source address.
	DAD	0 is written to this bit at initialization. (Fixed) Select according to the destination address.
	7 to 6	These bits are always read as 0. The write value should always be 0.
SARi	19 to 0	Set the source address.
	23 to 20	These bits are always read as 0. The write value should always be 0.
DARi	19 to0	Set the destination address.
	23 to 20	These bits are always read as 0. The write value should always be 0.
TCRi	15 to 0	Set the transfer count –1.

The setting example of interrupt control register is shown in the table bellow.

Table 1.19 Interrupt Control Register Settings

Register	Bit	Function and Setting
DMiIC	ILVL2 to ILVL0	Write 000b to these bits. (Level 0: Interrupt is disable)
	IR	If this bit is 1, Interrupt is requested. Write 0 to this bit according to the needs.

Note 1: UART1 is recommended not to use when DMA transfer is used.
When UART1 is in transmit state, DMA request cause select register is assigned to DMA0. When UART1 is in reception state, DMA request cause select register is assigned to DMA1. In order to DMA transfer is implemented, Both DMA0 and DMA1 have to be used and software has to be modified.

1.7.5 R8C/25

There isn't any DMAC function.

2. Sample Programs

Two or more of the same devices can be connected to the serial bus and controlled.

The sample programs execute the following:

- Data read processing
- Data write processing
- Write-protection processing through software protection
- Status read processing
- Deep power down processing
- Release deep power down processing
- ID read processing

2.1 Overview of Software Operations

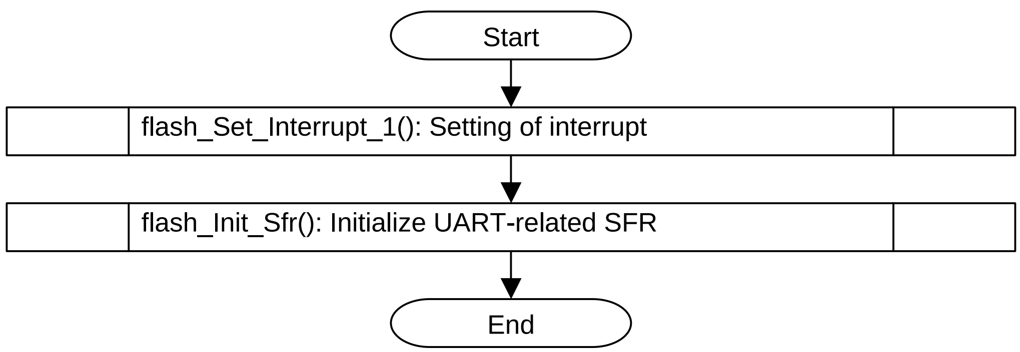
The operations roughly described below are performed.

(1) The driver initialization processing acquires the resources to be used by the driver and initializes them.
At this point, control signals (Port/CLK/TxD) connected to the serial Flash come to High.
(2) Function calls perform the following operations.
(a) The signals of pins connected to the serial Flash output to make serial Flash inactive state.
(b) Execute the processing of each function.
(c) Control signals (Port/CLK/TxD) connected to the serial Flash come to High.

2.2 Detailed Description of Functions

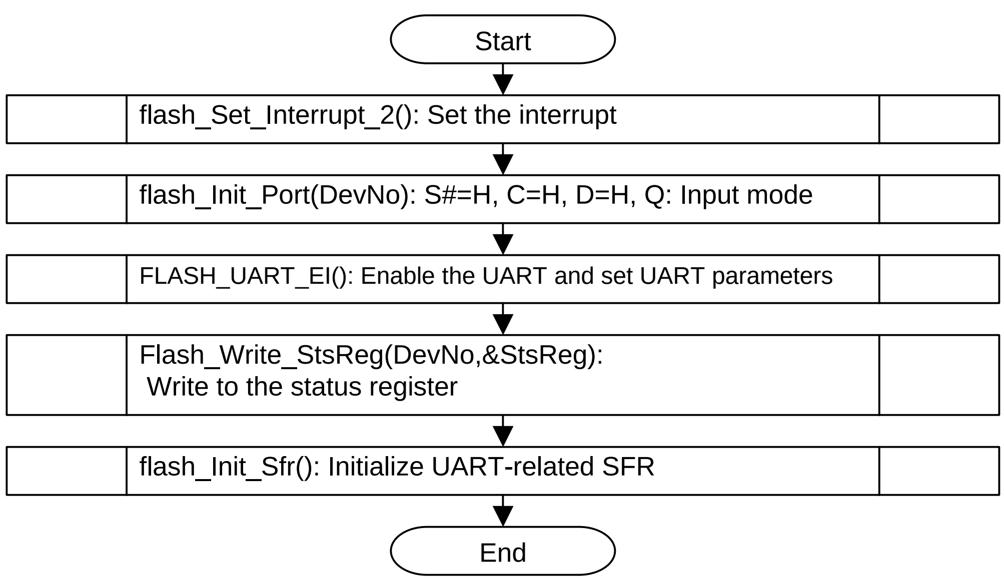
2.2.1 Driver Initialization Processing

Function Name
Flash driver initialization processing void flash_Init_Driver(void)
Arguments
None
Return Values
None
Operations
• Initializes the Flash driver. • Initializes the SFR for Flash control. • Call this function once at system activation.
Notes
None



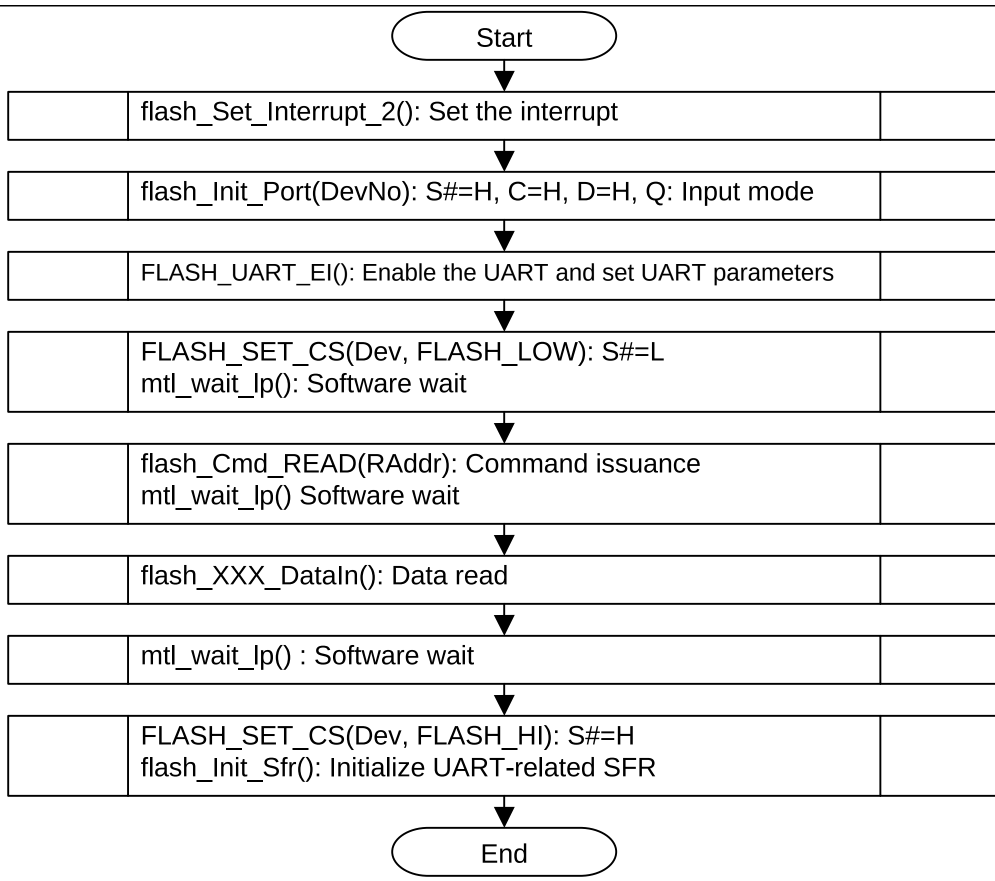
2.2.2 Write-Protection Setting Processing

Function Name
Write-protection setting processing signed short flash_Write_Protect(unsigned char DevNo, unsigned char WpSts)
Arguments
unsigned char DevNo ; Device number unsigned char WpSts ; Write-protection setting data
Return Values
Returns the write-protection setting result. FLASH_OK ; Successful operation FLASH_ERR_PARAM ; Parameter error FLASH_ERR_OTHER ; Other error
Operations
• Makes the write-protection setting. • The BP0, BP1 and BP2 bit of status register is set as follows by write-protection setting data (WpSts). WpSts=0: BP0=0, BP1=0, BP2=0 WpSts=1: BP0=1, BP1=0, BP2=0 WpSts=2: BP0=0, BP1=1, BP2=0 WpSts=3: BP0=1, BP1=1, BP2=0 WpSts=4: BP0=0, BP1=0, BP2=1 WpSts=5: BP0=1, BP1=0, BP2=1 WpSts=6: BP0=0, BP1=1, BP2=1 WpSts=7: BP0=1, BP1=1, BP2=1
Notes
SRWD is fixed 0. The Flash not assigned BP2 bit to status register should be set the WpSts among 0 to 3.



2.2.3 Data Read Processing

Function Name			
Data read processing signed short flash_Read_Data(unsigned char DevNo, unsigned long RAddr, unsigned long RCnt, unsigned char * pData)			
Arguments			
unsigned char	DevNo	;	Device number
unsigned long	RAddr	;	Read start address
unsigned long	RCnt	;	Number of bytes to be read
unsigned char FAR*	pData	;	Read data storage buffer pointer
Return Values			
Returns the read result.			
FLASH_OK		;	Successful operation
FLASH_ERR_PARAM		;	Parameter error
FLASH_ERR_HARD		;	Hardware error
FLASH_ERR_OTHER		;	Other error
Operations			
• Reads data from Flash in bytes. • Reads data from the specified address for the specified number of bytes.			
Notes			
• The maximum write address is Flash size – 1.			



2.2.4 Data Write Processing

Function Name

Data write processing

signed short flash_Write_Data(unsigned char DevNo, unsigned long WAddr, unsigned long WCnt, unsigned char FAR* pData)

Arguments

unsigned char	DevNo	;	Device number
unsigned long	WAddr	;	Write start address
unsigned long	WCnt	;	Number of bytes to be written
unsigned char FAR*	pData	;	Write data storage buffer pointer

Return Values

Returns the write result.

FLASH_OK	;	Successful operation
FLASH_ERR_PARAM	;	Parameter error
FLASH_ERR_HARD	;	Hardware error
FLASH_ERR_OTHER	;	Other error

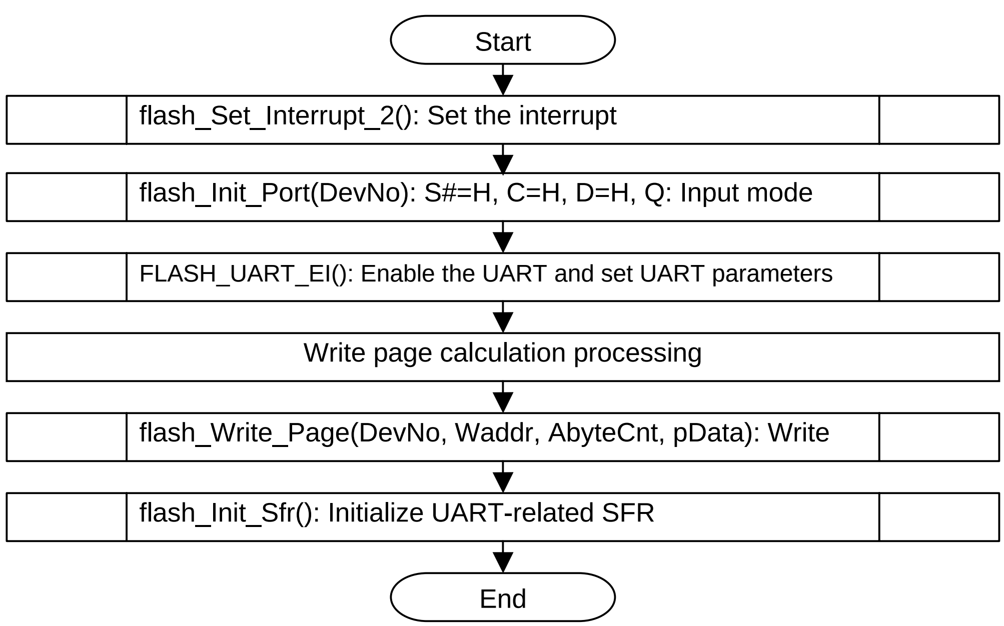
Operations

- Writes data to Flash in bytes.
- Writes data from the specified address for the specified number of bytes.

Notes

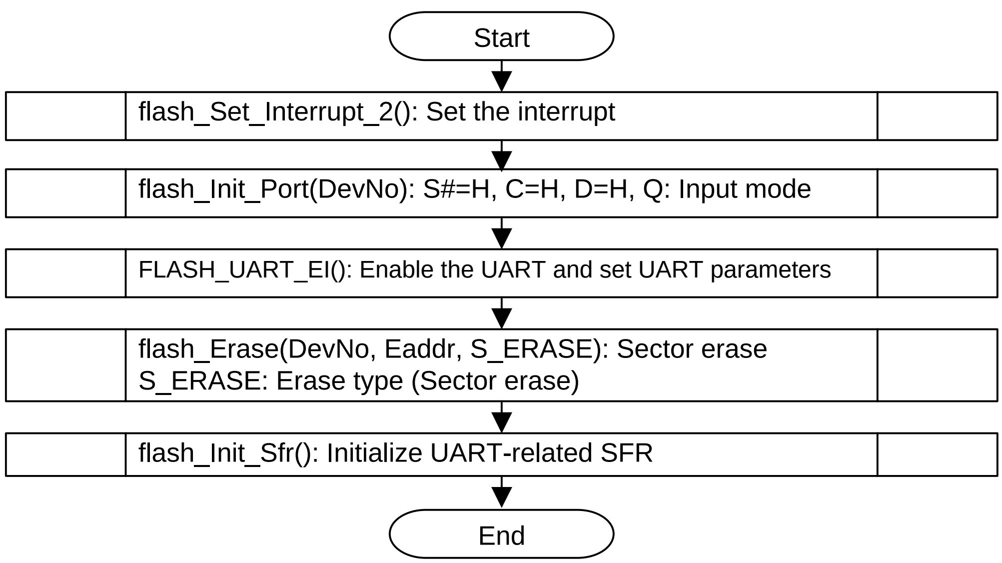
- Flash can be written to only when write-protection has been canceled.
- Data can’t be written to protected pages and error result isn’t returned.
- The maximum write address is Flash size – 1.

In a write to the serial Flash, the page rewrite method is used. The original data is divided into the page-unit data and then written to the Flash.



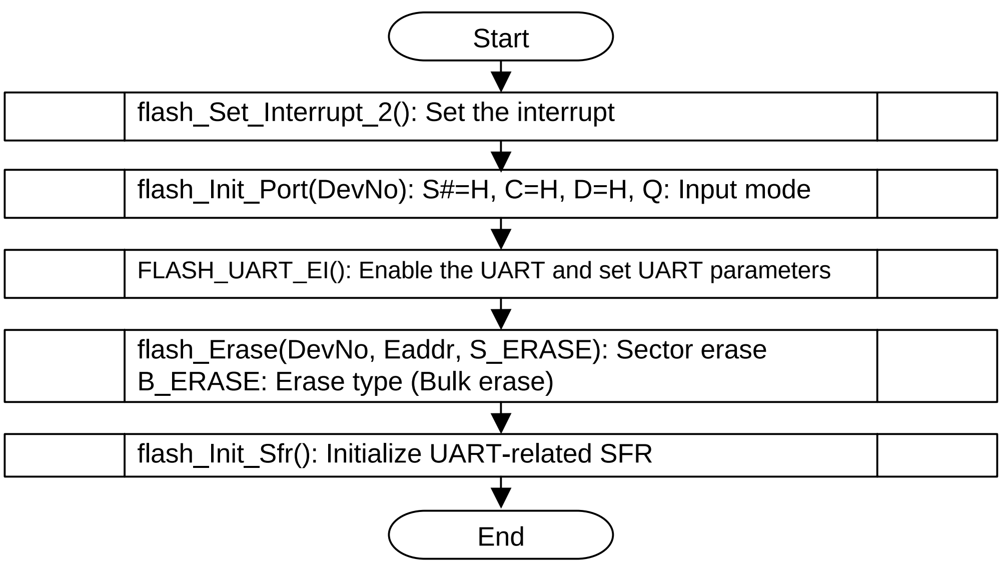
2.2.5 Sector Erase Processing

Function Name			
Sector erase processing			
signed short flash_SectorErase(unsigned char DevNo, unsigned long EAddr)			
Arguments			
unsigned char	DevNo	;	Device number
unsigned long	EAddr	;	Erase address
Return Values			
Returns the sector erase result.			
FLASH_OK		;	Successful operation
FLASH_ERR_PARAM		;	Parameter error
FLASH_ERR_HARD		;	Hardware error
FLASH_ERR_OTHER		;	Other error
Operations			
• Erase the sector data of specified address			
Notes			
• Flash can be erased to only when write-protection has been canceled.			
• Data of protected sector can't be erased and error result isn't returned.			



2.2.6 Bulk Erase Processing

Function Name
Sector erase processing signed short flash_BulkErase(unsigned char DevNo)
Arguments
unsigned char DevNo ; Device number
Return Values
Returns the bulk erase result. FLASH_OK ; Successful operation FLASH_ERR_PARAM ; Parameter error FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error
Operations
• Erase the all data of Flash
Notes
• Flash can be erased to only when write-protection has been canceled. • When the Flash is Write- protected, it can't be erased and error result isn't returned.



2.2.7 Status Read Processing

Function Name

Status read processing
signed short flash_Read_Status(unsigned char DevNo, unsigned char * pStatus)

Arguments

unsigned char	DevNo	;	Device number
unsigned char FAR*	pStatus	;	Read status storage buffer

Return Values

Returns the status register acquisition result.

FLASH_OK	;	Successful operation
FLASH_ERR_PARAM	;	Parameter error
FLASH_ERR_HARD	;	Hardware error
FLASH_ERR_OTHER	;	Other error

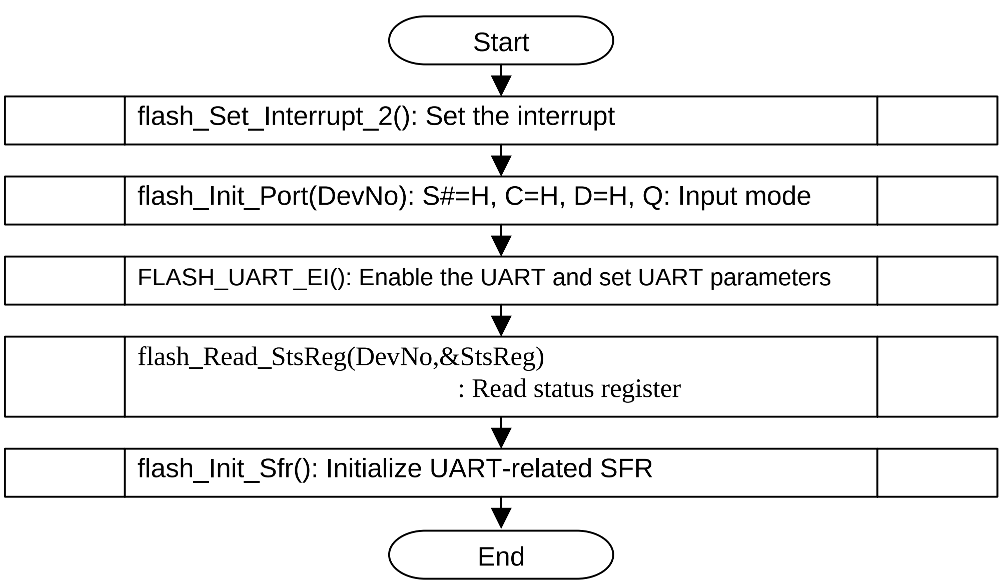
Operations

- Reads the status.
  Reads from the status register.
- The following information is stored in the read status storage buffer (pStatus).

Bit 7:	SRWD	0: Status register can be changed
		1: Status register cannot be changed
Bits 6 to 5:	Reserved (All 0)	
Bits 4 to 2:	BP2/BP1/BP0	
Bit 1:	WEL	0: Write disabled
		1: Write enabled
Bit 0:	WIP	1: During write operation

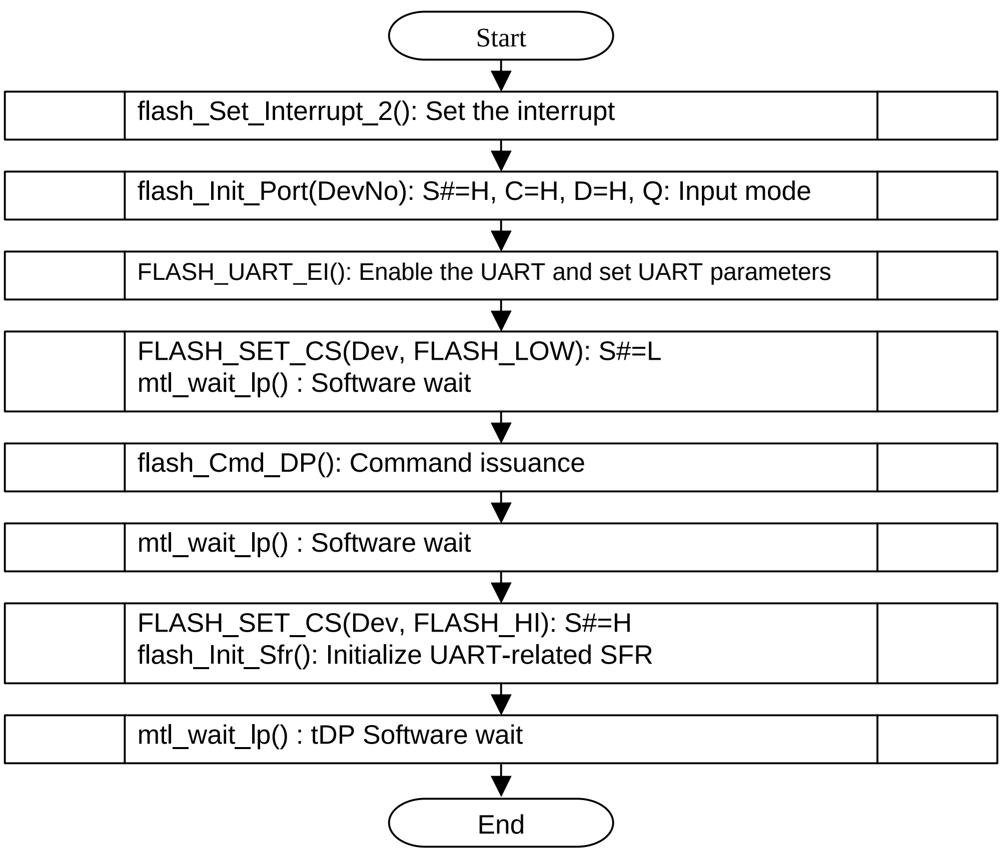
Notes

- Refer the specification of use Flash about relation between block protection bits (BP0, BP1 and BP2) and protection area.
- The information of BP2 bit isn't output from the Flash which status register doesn't include of BP2 bit.



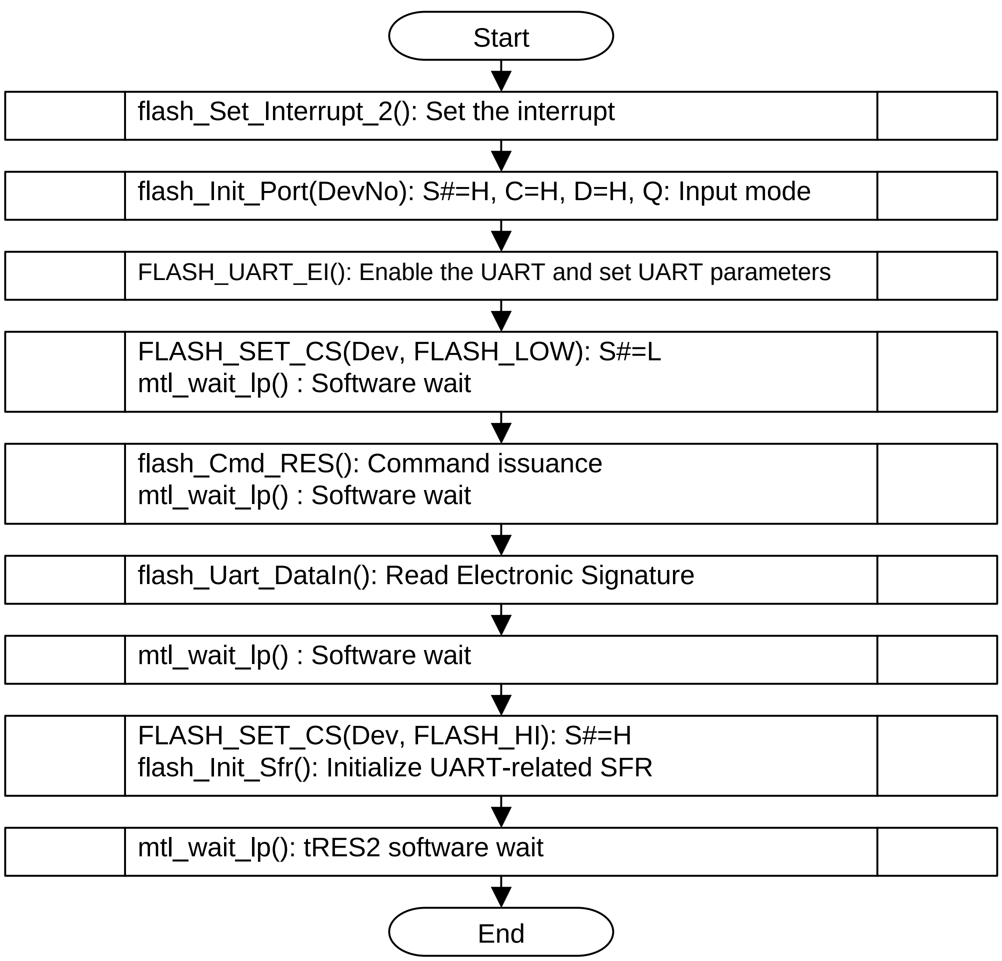
2.2.8 Deep power down Processing

Function Name
Deep power down processing signed short flash_DeepPDown(unsigned char DevNo)
Arguments
unsigned char DevNo ; Device number
Return Values
Returns the deep power down result. FLASH_OK ; Successful operation FLASH_ERR_PARAM ; Parameter error FLASH_ERR_HARD ; Hardware error FLASH_ERR_OTHER ; Other error
Operations
• Set the deep power down mode.
Notes
• None



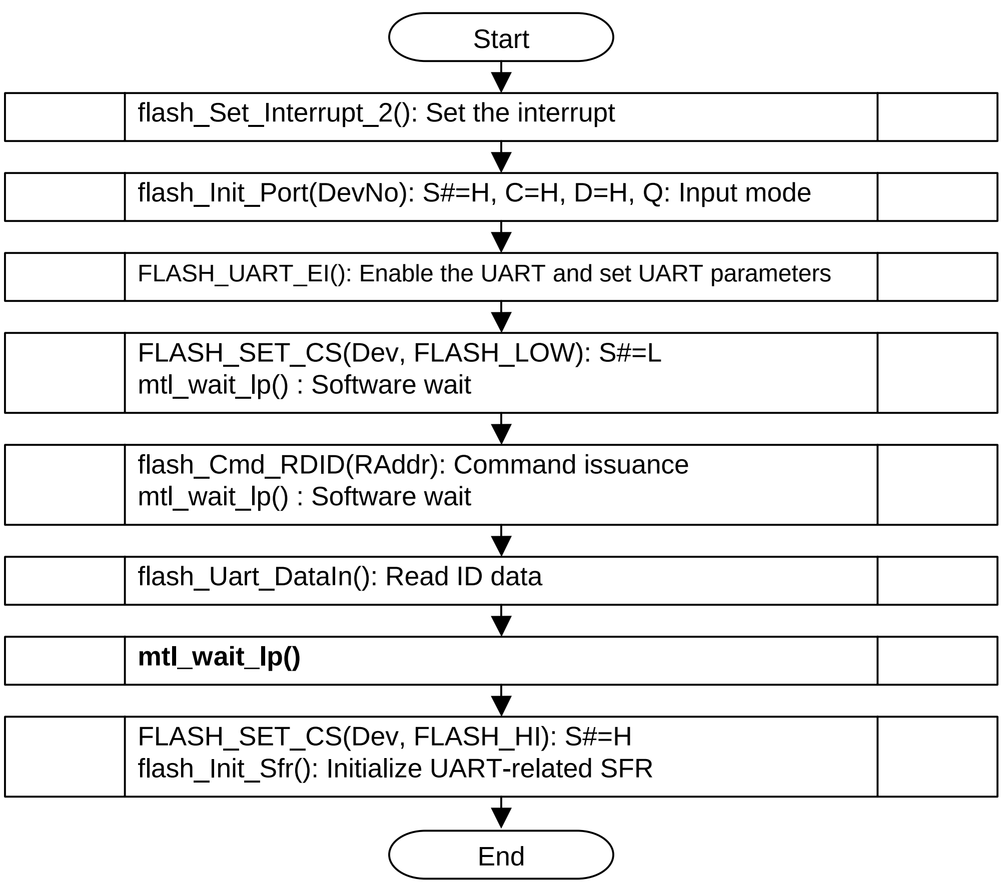
2.2.9 Release deep power down Processing

Function Name		
Release deep power down processing signed short flash_ReleaseDeepPDown(unsigned char DevNo, unsigned char FAR* pData)		
Arguments		
unsigned char DevNo	;	Device number
unsigned char FAR* pData	;	Electronic signature storage buffer pointer
Return Values		
Returns the release deep power down result.		
FLASH_OK	;	Successful operation
FLASH_ERR_PARAM	;	Parameter error
FLASH_ERR_HARD	;	Hardware error
FLASH_ERR_OTHER	;	Other error
Operations		
• Change mode from deep power down to standby. • Read electronic signature.		
Notes		
• None		



2.2.10 ID read Processing

Function Name			
ID read processing			
signed short flash_ReleaseDeepPDown(unsigned char DevNo, unsigned char FAR* pData)			
Arguments			
unsigned char	DevNo	;	Device number
unsigned char FAR*	pData	;	ID data storage buffer pointer ID data of 3 bytes are stored in the following order. (1) Manufacture ID (2) Memory type (3) Memory capacity
Return Values			
Returns the ID read result.			
FLASH_OK		;	Successful operation
FLASH_ERR_PARAM		;	Parameter error
FLASH_ERR_HARD		;	Hardware error
FLASH_ERR_OTHER		;	Other error
Operations			
• Read Manufacture ID and Device ID			
Notes			
• None			



2.2.11 Return Value Definition

```
#define FLASH_OK           (short)( 0) /* Successful operation    */
#define FLASH_ERR_PARAM    (short)(-1) /* Parameter error         */
#define FLASH_ERR_HARD     (short)(-2) /* Hardware error          */
#define FLASH_ERR_OTHER    (short)(-3) /* Other error             */
```

2.3 User Setting Examples

Setting examples when using the Renesas Technology MCU M16C/62P are shown below.
The location where a setting should be made is indicated by the comment of /** SET **/ in each file.

2.3.1 flash.h

(1) Definition of the number of devices used and device numbers

Specify the number of devices to be used and assign a number for each device.
In the example below, one device is used and 0 is assigned as the device number.
When using three or more, flash_io.h needs to be modified in addition to this file.

```
/*---------------------------------------------------------------------------------*/
/*  Define the number of the required serial Flash devices.(1 to N devices)      */
/*  Define the device number in accordance with the number of serial Flash devices */
/* to be connected.                                                                */
/*---------------------------------------------------------------------------------*/
/* Define number of devices */
#define FLASH_DEV_NUM         1      /* 1 device                                   */

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

(2) Definition of device used

Specify the device to be used.
In the example below, M25P10A device is used.

```
/*----------------------------------------------------------------------------- */
/*  Define the serial Flash device.                                             */
/*----------------------------------------------------------------------------- */
 //#define M25P05A      /* 512kbit ( 64kByte)*/ /** SET **/
#define M25P10A         /* 1Mbit ( 128kByte) */ /** SET **/
//#define M25P20        /* 2Mbit ( 256kByte) */ /** SET **/
//#define M25P40        /* 4Mbit ( 512kByte) */ /** SET **/
//#define M25P16        /* 16Mbit( 2MByte)   */ /** SET **/
//#define M25P32        /* 32Mbit( 4MByte)   */ /** SET **/
//#define M25P64        /* 64Mbit( 8MByte)   */ /** SET **/
```

(3) Definitions the way of interrupt setting of UART or DMA

Define the way of transmit interrupt control process.

This software controls the transmission processing by disabling the Interrupt Priority Select Bits and utilizing Interrupt Request Bit (IR) in Interrupt Control Register of UART or DMA.

The method of the interrupt disabling can be selected by the following three ways.

Select one of them according to the system.

Case 1. Set in the upper system and not setting in the device driver.
#define FLASH_IC_SETTING0 should be validated.

Case 2. Set when the device driver is initialized – in executing “flash_Init_Driver()”.
#define FLASH_IC _SETTING1 should be validated.

Case 3. Set when UART transfer – in executing “flash_Read_Data()”, “flash_Write_Data()”.
#define FLASH_IC _SETTING2 should be validated.

Case 2 and 3 can be validated at the same time.

Precaution

The followings are the interrupt setting sequence when the above Case 2 and/or 3 are selected:

Disable interrupt (DI)

Disable the Interrupt Priority Select Bits and clear the Interrupt Request Bit (IR) of Interrupt Control Register for UART or DMA.

Enable interrupt (EI)

Be careful when interrupts enable flag (I flag) is managed by a higher system.

```
/*----------------------------------------------------------------------------*/
/* The setting method of the interrupt when "FLASH_IC_SETTING1" and
"FLASH_IC_SETTING2" are */
/*  selected is as follows.                                                  */
/*  Interrupt disable (DI) -> interrupt setting -> interrupt enable (EI)    */
/*  When manage an interrupt enable flag (I flag) by a higher system, please
be careful.    */
/*  When interrupt it by a higher system and manage it, please choose
"FLASH_IC_SETTING0".    */
/*----------------------------------------------------------------------------*/
#define FLASH_IC_SETTING0     /* Doesn't set in this driver            */
//#define FLASH_IC_SETTING1   /* When the driver is initialized, it sets */
//#define FLASH_IC_SETTING2   /* When the resource is used, it sets     */
```

2.3.2 flash_sfr.h

Rename from flash_sfr.h.xxx (the header corresponding to the MCU) to flash_sfr.h and use it.

In the example below, the M16C/62P is used.

The sample program shows a description example in which UART 0 is used as the resource of the clock synchronous serial I/O. When DMAC is used it shows a description example in which DMA 0 is used.

No setting needs to be modified when the above resource is used.

(1) UART resource

```
/*----------------- UART definitions -----------------*/
#define FLASH_UART_STIC     s0tic /* UART TX interrupt control register      */

#define FLASH_UART_TXBUF    u0tb  /* UART transmit buffer register           */
#define FLASH_UART_TXBUFL   u0tbl /* UART transmit buffer register(lower
8bit)*/
#define FLASH_UART_RXBUF    u0rb  /* UART receive buffer register            */
#define FLASH_UART_BRG      u0brg /* UART bit rate generator                 */
#define FLASH_UART_MR       u0mr  /* UART transmit/receive mode register     */
#define FLASH_UART_C0       u0c0  /* UART transmit/receive control register 0*/
#define FLASH_UART_C1       u0c1  /* UART transmit/receive control register 1*/

#define FLASH_UART_TXEND    txept_u0c0  /* UART TX Reg. empty flag */
#define FLASH_UART_TXNEXT   ir_s0tic    /* UART TX complete flag   */
#define FLASH_UART_TI       ti_u0c1     /* UART TX complete flag   */
#define FLASH_UART_RXNEXT   ri_u0c1     /* UART RX complete flag   */
#define FLASH_UART_IRS      u0irs       /* UART transmit interrupt cause select
flag  */
#define FLASH_UART_RRM      u0rrm       /* UART continuous receive mode enable
flag  */
```

If another resource is used, make additions or modify the above program. Accordingly, also make additions or modify the /* UART setting */ definition with reference to section 1.6, M16C SFR (Peripheral Device Control Register) Setting - Clock Synchronous serial I/O and Interrupt control Register.

(2) DMAC resource

```
/*----------------- DMAC definitions -----------------*/
#ifdef FLASH_DMA_ON
#define FLASH_DMA_DMIC      dm0ic    /* DMA interrupt control register   */

#define FLASH_DMA_SL        dm0sl    /* DMA request cause select register*/
#define FLASH_DMA_CON       dm0con   /* DMA control register             */
#define FLASH_DMA_SAR       sar0     /* DMA source pointer               */
#define FLASH_DMA_DAR       dar0     /* DMA destination pointer          */
#define FLASH_DMA_TCR       tcr0     /* DMA transfer counter             */
#define FLASH_DMA_END       ir_dm0ic /* DMA interrupt request flag       */
```

If another resource is used, make additions or modify the above program. Accordingly, also make additions or modify the /* DMA setting */ definition with reference to section 1.7, M16C SFR (Peripheral Device Control Register) Setting - DMAC and Interrupt control Register

2.3.3 flash_io.h

Rename from flash_io.h.xxx (the header corresponding to the MCU) to flash_io.h and use it.
In the example below, the M16C/62P is used.

(1) Definition of resources used by UART of MCU used

Specify the resources of the MCU to be used.
In the example below, the clock synchronous serial I/O is used.

```
/*---------------------------------------------------------------------- */
/*  Define the combination of the MCU's resources.                       */
/*---------------------------------------------------------------------- */
//#define FLASH_OPTION_1         /* Low speed   */ /* UART      */
#define FLASH_OPTION_2           /* High speed  */ /* UART + DMAC */
```

(2) Definition of control ports of MCU used

Specify the control ports of the MCU to be used.
In the example below, RxD, TxD, CLK, and CS# of the clock synchronous serial I/O are assigned.
When two devices are connected, make a definition regarding CS1.
When using three or more, flash.h needs to be modified in addition to this file.

```
/*----------------------------------------------------------------------------*/
/*  Define the control port.                                                  */
/*----------------------------------------------------------------------------*/
#define FLASH_P_DATAO      p6_3          /* FLASH DataOut                */
#define FLASH_P_DATAI      p6_2          /* FLASH DataIn                 */
#define FLASH_P_CLK        p6_1          /* FLASH CLK                    */
#define FLASH_D_DATAO      pd6_3         /* FLASH DataOut                */
#define FLASH_D_DATAI      pd6_2         /* FLASH DataIn                 */
#define FLASH_D_CLK        pd6_1         /* FLASH CLK                    */

#define FLASH_P_CS0        p10_5    /* FLASH CS0 (Negative-true logic)   */
#define FLASH_D_CS0        pd10_5   /* FLASH CS0 (Negative-true logic)   */
#if (FLASH_DEV_NUM > 1)
#define FLASH_P_CS1        p10_1    /* FLASH CS1 (Negative-true logic)   */
#define FLASH_D_CS1        pd10_1   /* FLASH CS1 (Negative-true logic)   */
#endif   /* #if (FLASH_DEV_NUM > 1) */
```

(3) Definition the software timer value of erase busy waiting

Define the software timer value of erase busy waiting depending on the Flash.

```
/*------------------------------------------------------------------------*/
/* 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_TIME

#ifdef FLASH_EBUSY_WAIT_TIME
#define FLASH_EBUSY_WAIT(ushort)40000  /* Erase busy waiting time  40000* 1Ms
=   4s */
#else
#define FLASH_EBUSY_WAIT(ushort)0   /* Waiting without time out  */
#endif
```

2.3.4 mtl_com.h (Common Header File)

Rename from mtl_com.h.xxx (the header corresponding to the MCU) to mtl_com.h and use it.
In the example below, the M16C/62P is used.

(1)Definition of OS header file

This software is an OS-independent program.
In the example below, the OS is not used. (The system call of MR30 is not used.)

```
/* In order to use wai_sem/sig_sem/dly_tsk for microITRON (Real-Time OS)-
compatible, */
/* include the OS header file that contains the prototype declaration.
/* When not using the OS, put the following 'define' and 'include' as comments.
  */
//#define MTL_OS_USE                              /* Use OS            */
//#include <RTOS.h>                               /* OS header file    */
//#include "mtl_os.h"
```

(2) Definition of header file specifying common access area

Includes the header file in which the MCU registers are defined.
This file needs to be included because it is mainly used by the device driver for controlling the ports.
In the example below, the M16C/62P header file is included. Include the header file in accordance with the MCU.

```
/* In order to use definitions of MCU SFR area,                      */
/* include the header file of MCU SFR definition.                    */
#include "sfr62p.h"                        /* definition of MCU SFR   */
```

(3) Definition of loop timer

Include the header file below if software timer is used.
It is mainly used as wait time of device driver.
When software timer is not used, the define statement below should be a comment.
In the example below, software timer is used.

```
/* When not using the loop timer, put the following 'include' as comments.  */
#include "mtl_tim.h"
```

(4) Definition of endian type

This is the setting of FAT file system library for M16C family.
Specify the little endian if M16C family is used.

```
/* When using M16C or SuperH for Little Endian setting, define it. */
/* When using other MCUs, put 'define' as a comment.               */
#define MTL_MCU_LITTLE                  /* Little endian           */
```

(5) The fast processes of mtl_endi.c

This is the setting of FAT file system library for M16C family.

Specify the little endian if M16C family is used.

```
/* When using M16C, define it.                                */
/* It performs the fast processes of 'mtl_endi.c'.            */
#define MTL_ENDI_HISPEED        /* Uses the high-speed function. */
```

(6) Specification of standard library type used

Specify the standard library type used. When the processing below is used in the library provided with the compiler, the define statement below should be a comment.

The optimized library enabling high-speed processing is prepared.

The following example shows the standard library set.

```
/* Specify the standard library type used.                              */
/* When using the compiler-bundled library for the following processes,      */
/* put the following 'define' as comments.                               */
/* memcmp() / memcpy() / memset() / strcat() / strcmp() / strcpy() / strlen()*/
//#define MTL_USER_LIB                    /* Optimized library usage        */
```

(7) Definition of RAM area accessed by processing group used

Define the RAM area to be accessed by the user process group.

Standard functions and efficient operations for processes are applied.

If neither of them is defined, error is output when software is compiled

When accessing to the FAR RAM of M16C/60, M16C/30, or M16C/20, define ‘MTL_MEM_FAR’.

The following is a definition example of MTL_MEM_NEAR when M16C/60, M16C/30, M16C/20 or R8C is used.

```
/* Define the RAM area to be accessed by the user process.              */
/* Efficient operations for standard functions and processes are applied.   */
//#define MTL_MEM_FAR      /* Supports Far RAM area of M16C/60             */
#define MTL_MEM_NEAR       /* Supports Near RAM area. (Others)             */
```

Set only the above define statement and do not make any other modifications.

2.3.5 mtl_tim.h

(1) Definition of software timer

Sets the internal software timer used.
The following reference values are obtained at 24-MHz operation without wait.
The setting should be made in accordance with the system.

```
/* Define the counter value for the timer.                                       */
/* Specify according to the user MCU, clock and wait requirements.               */
/* Setting for 24MHz no wait                                                      */
#define MTL_T_1US          1                     /* 1-us loop count       */
#define MTL_T_2US          2                     /* 2-us loop count       */
#define MTL_T_4US          5                     /* 4-us loop count       */
#define MTL_T_5US          6                     /* 5-us loop count       */
#define MTL_T_10US         13                    /* 10-us loop count      */
#define MTL_T_20US         28                    /* 20-us loop count      */
#define MTL_T_30US         43                    /* 30-us loop count      */
#define MTL_T_50US         72                    /* 50-us loop count      */
#define MTL_T_100US        145                   /* 100-us loop count     */
#define MTL_T_200US        293                   /* 200-us loop count     */
#define MTL_T_300US        439                   /* 300-us loop count     */
#define MTL_T_400US     ( MTL_T_200US * 2 )      /* 400-us loop count     */
#define MTL_T_1MS          1471                  /* 1-ms loop count       */
```

2.3.6 Usage Notes

The sample programs show description example in which UART 0 is used as the resource of the clock synchronous serial I/O. When DMAC is used it shows a description example in which DMA 0 is used.

When using another resource, set the software in accordance with the hardware.

2.3.7 Notes at Embedment

To embed the sample programs, include flash.h.

2.3.8 Usage of Another M16C Family MCU

Usage of another M16C family MCU is supported easily.

The following files must be prepared.

(1) I/O module common definition equivalent of flash_io.h.xxx
Define the I/O pins to be used with reference to the SFR header of the MCU used.
(2) SFR common definition equivalent of flash_sfr.h.xxx
Define the UART/DMA to be used with reference to the SFR header of the MCU used.
(3) Header definition equivalent of mtl_com.h.xxx
Create and define a header for the MCU used.

Create the above files with reference to the provided programs.

In addition, specify the created header in flash_io.h, flash_sfr.h, and mtl_com.h.

2.3.9 File Configuration

\com	<DIR>		Directory for common functions
	mtl_com.c	mtl_com.h.common	Various definitions for common functions
	mtl_com.h.m16c29		M16C/29 Common header file
	mtl_com.h.m16c30p		M16C/30P Common header file
	mtl_com.h.m16c62p		M16C/62P Common header file
	mtl_com.h.m32c87		M32C/87 Common header file
	mtl_com.h.r8c25		R8C/25 Common header file
	mtl_mem.c		Common file
	mtl_os.c	mtl_os.h	Common file
	mtl_str.c		Common file
	mtl_tim.c	mtl_tim.h	Common file
	mtl_tim.h.sample		Common header file (Reference)
\sflash_spi	<DIR>		Serial Flash directory
	flash.h		Driver common definition
	flash_io.c		I/O module
	flash_io.h.m16c29		M16C/29 I/O module common definition
	flash_io.h.m16c30p		M16C/30P I/O module common definition
	flash_io.h.m16c62p		M16C/62P I/O module common definition
	flash_io.h.m32c87		M32C/87 I/O module common definition
	flash_io.h.r8c25		R8C/25 I/O module common definition
	flash_sfr.h.m16c29		M16C/29 SFR common definition
	flash_sfr.h.m16c30p		M16C/30P SFR common definition
	flash_sfr.h.m16c62p		M16C/62P SFR common definition
	flash_sfr.h.m32c87		M32C/87 SFR common definition
	flash_sfr.h.r8c25		R8C/25 SFR common definition
	flash_usr.c		Driver user I/F module
\sample	<DIR>		Sample program directory
	testmain.c		Sample program for operation verification Use this for operation verification.
	common.c	common.h	Various definitions for common functions

Website and Support

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

Inquiries
http://www.renesas.com/inquiry
csc@renesas.com

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Feb.20.07	—	First edition issued
1.01	Nov.09.07	P3	Section1.4 “High-z processing after function operating” is changed to “Processing after function operating” Contents of Section 1.4 is modified.
		P15	DMA setting of M32C/87 is deleted.
		P19	Changed three places in the next sentence. “Control signals (Port/CLK/TxD)) connected to the serial Flash come to High”
		P20-P30	The content “flash_Open_Port(DevNo): Make the ports Hi-z” is deleted from flow chart.
1.02	Feb.17.08	P1	Target Device Software Version was added.