M16C/5M Group

E²PROM Emulation Data Flash
Using the E²PROM Emulation Data Flash

R01AN0423EJ0100
Rev.1.00
Dec. 15, 2010

## 1. Abstract

This document describes the setting method and an application example of the E²PROM emulation data flash (hereafter referred to as E²dataFlash) in the M16C/5M Group.

## 2. Introduction

The application example described in this document applies to the following microcomputers (MCU):
- MCU: M16C/5M Group

This application note can be used with other M16C Family MCUs which have the same special function registers (SFRs) as the above group. Check the user's manual for any modifications to functions. Careful evaluation is recommended before using the program described in this application note.

# 3. Overview

The E2dataFlash is a data flash that utilizes the strengths of serial E2PROMs. The units erased are considerably smaller than that of the data flash, and the E2dataFlash can be programmed or erased without stopping the CPU.
Table 3.1 lists the E2dataFlash Specifications.

**Table 3.1 E2dataFlash Specifications**

| Item | Specification | |
|---|---|---|
| | ECC disabled | ECC enabled |
| Memory size | 4 Kbytes | 2 Kbytes |
| Block size | 32 bytes | 16 bytes |
| Number of blocks | 128 blocks | |
| Unit to be programmed | 2 bytes | 1 byte |
| Unit to be erased | 1 block (after erase, the memory value is 1.) | |
| How to control program and erase | By software commands | |
| Software commands | 4 | |
| Error correction | None | Corrects 1-bit error per byte |

ECC: Error Check and Correct

## 3.1 Block Configuration

When ECC is disabled, the E2dataFlash consists of 32 bytes × 128 blocks of flash memory and 16 bytes × 128 blocks when ECC is enabled.
Figure 3.1 shows the E2dataFlash Memory Configuration.

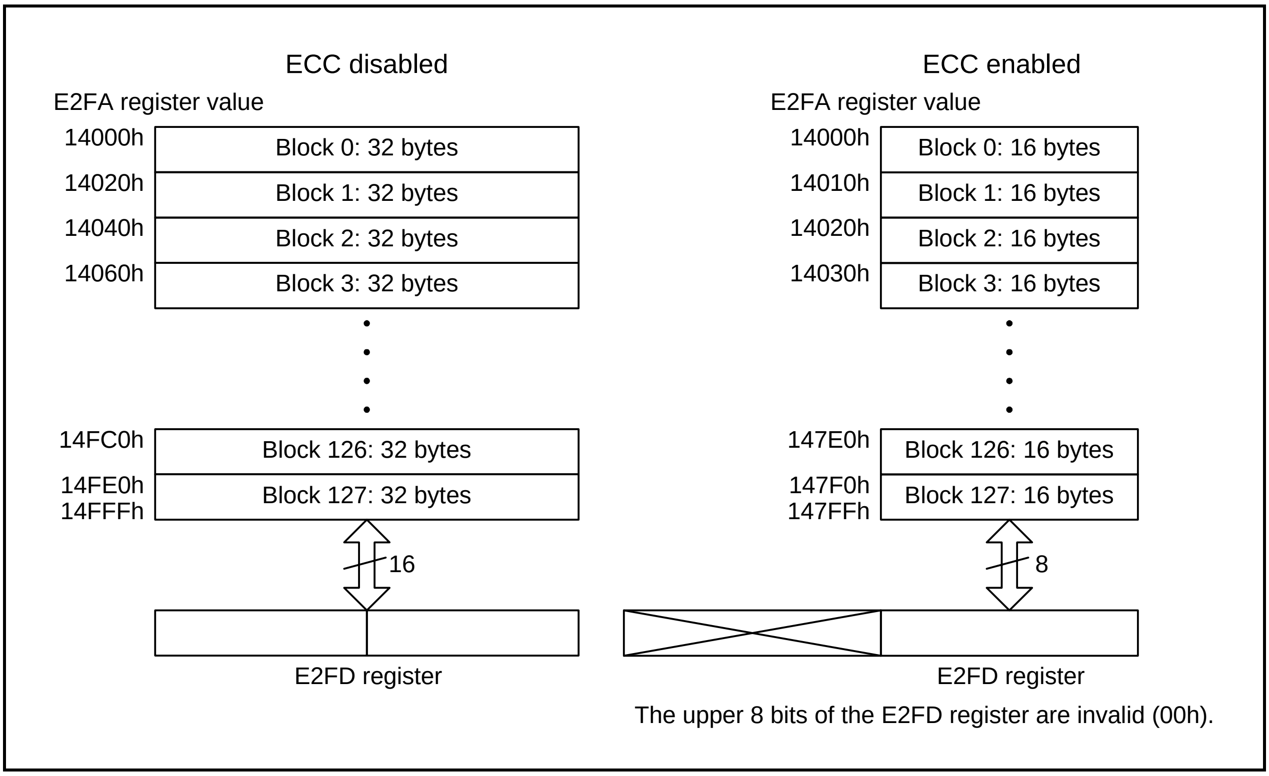



**Figure 3.1 E2dataFlash Memory Configuration**

## 3.2 E2dataFlash Access Method

The CPU indirectly accesses the E2dataFlash via registers E2FA, E2FI, and E2FD that are allotted for SFR space. Figure 3.2 shows Registers E2FA, E2FI, and E2FD.

b31 b0

E2dataFlash Address Register (E2FA) (1)

Set an address to be accessed.
The 2 upper bytes are read as an undefined value.

14000h to 14FFEh
(for ECC disabled)
14000h to 147FFh
(for ECC enabled)

b15 b0

E2dataFlash Command Register (E2FI) (2, 3)

Set a command for the flash memory.
The upper byte is read as an undefined value.

| Command | Setting value |
|---|---|
| Read | 00F3h |
| Program | 0040h |
| Block Erase | 0020h, 00D0h |
| Clear Status Register | 0050h |

b15 b0

E2dataFlash Data Register (E2FD) (2, 3, 4)

After a read command, the data of the address written to the E2FA register is read.
After a program command, the data set to this register is written to the address set to the E2FA register.

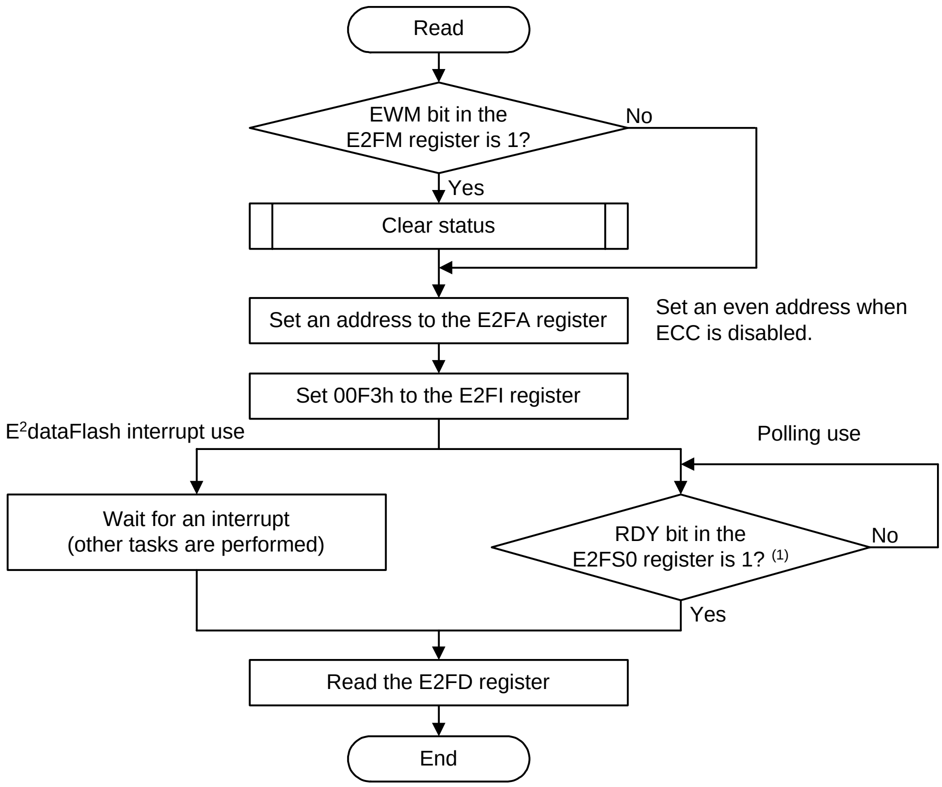
0000h to FFFFh
(for ECC disabled)
0000h to 00FFh
(for ECC enabled)

Notes:
1. When the ECC bit in the E2FM register is 0 (ECC disabled), set the E2FA register to an even address (LSB is 0) to write or read the E2dataFlash. When erasing, set the highest-order even address of the corresponding block.
2. A 16-bit write/read access to this register should be performed.
3. Two values need to be written for block erase.
   When writing to the E2FI register, check that the RDY bit in the E2FS0 register is 1.
4. In a program operation, the data should be written after 0040h is set to the E2FI register.

**Figure 3.2 Registers E2FA, E2FI, and E2FD**

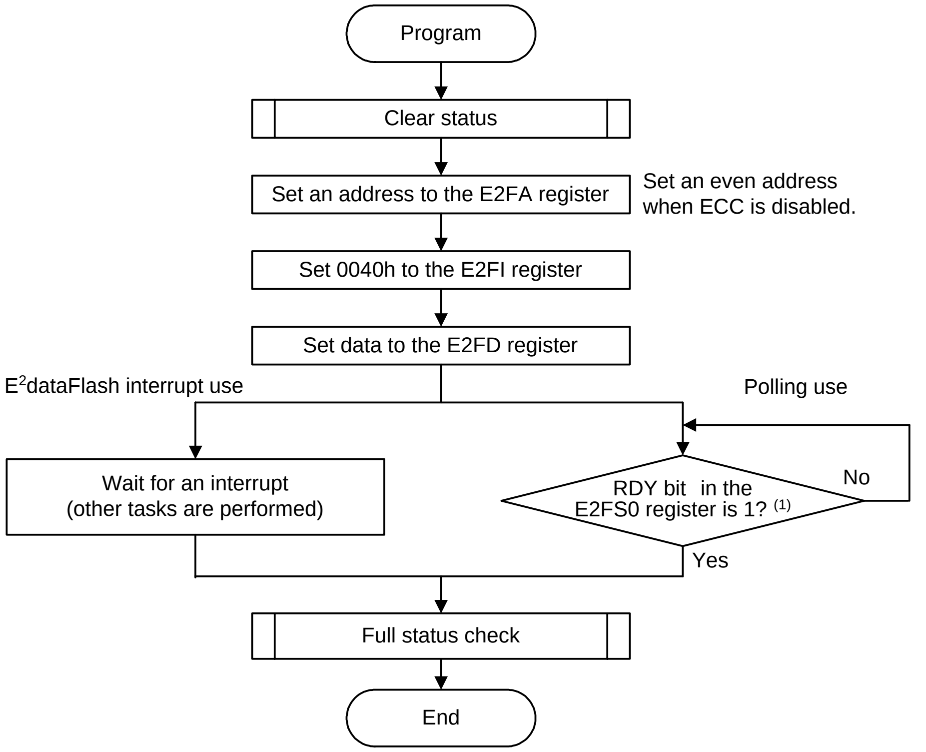
## 3.3 Operational Procedures

When operating the E2dataFlash, confirm that the RDY bit in the E2FS0 register is 1 (ready), and then execute the read, program, block erase, and clear status operations following the steps shown in Figure 3.3 to Figure 3.6.
Do not program or erase the E2dataFlash during programming or erasing program ROM 1, program ROM 2, or the data flash. Do not overwrite data to an address that has already been written.



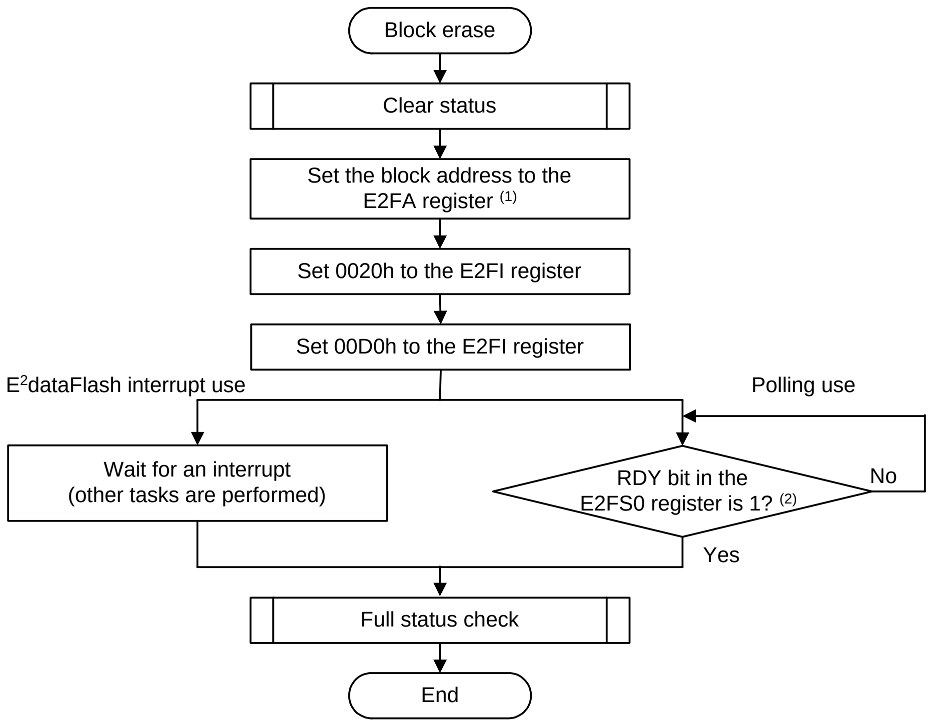

Note:
1. When reading the E2FS0 register after writing to the E2FI register, insert at least four NOP instructions before reading the E2FS0 register.

**Figure 3.3 Read Operation Example**




Note:
1. When reading the E2FS0 register after writing to the E2FD register, insert at least four NOP instructions before reading the E2FS0 register.

**Figure 3.4 Program Operation Example**




Notes:
1. The block address is the highest-order even address of the block to be erased.
   The memory value becomes 1 after erase.
2. When reading the E2FS0 register after writing to the E2FI register, insert at least four NOP instructions before reading the E2FS0 register.

**Figure 3.5 Block Erase Operation Example**

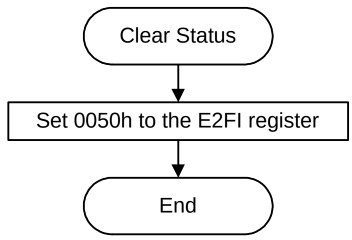



**Figure 3.6 Clear Status Operation Example**

## 3.4 Full Status Check

If an error occurs, the WERR or EERR bit in the E2FS0 register becomes 1, indicating the occurrence of an error. Therefore, the execution results can be confirmed by checking the status of these bits (full status check).
Figure 3.7 shows the Full Status Check.

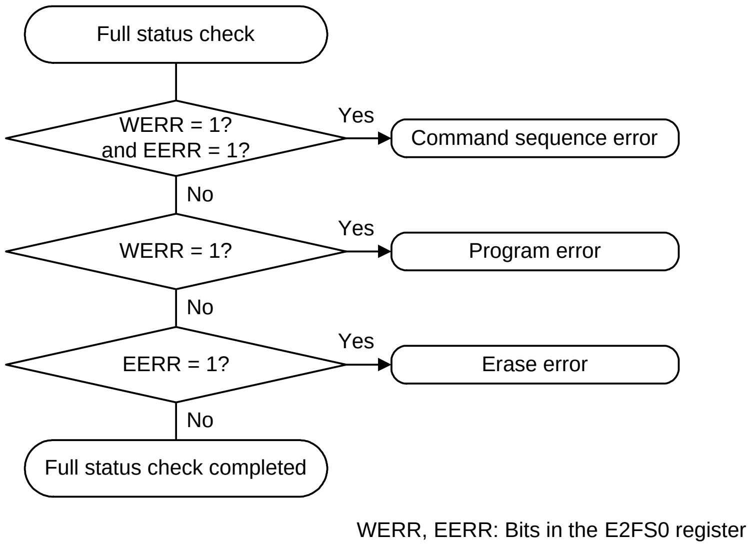



**Figure 3.7 Full Status Check**

### 3.4.1 Error Processing Procedures

When errors occur, follow the procedures below.
Do not execute the program command or block erase command when the WERR or EERR bit is 1 (completed in error). Execute each command after executing the clear status register command.

Command sequence error
(1) Execute the clear status register command and set the WERR or EERR bit to 0 (no error).
(2) Check to see that the command is written correctly and execute the command again.
A command sequence error occurs when writing data other than xxD0h and xxFFh in the second bus cycle of the block erase command. When writing xxFFh in the second bus cycle of the block erase command, the $E^2$dataFlash becomes the state before executing the command, and the command code written in the first bus cycle is cancelled.

Erase error
(1) Execute the clear status register command and set the EERR bit to 0 (no erase error).
(2) Execute the block erase command again.
Execute (1) and (2) until an erase error does not occur.
If an error still occurs after executing (1) and (2) three times, do not use that block.

Program error
(1) Execute the clear status register command and set the WERR bit to 0 (no write error).
(2) Execute the block erase command.
(3) Execute the program command.
If an error still occurs, do not use that address.

# 4. Application Example

The application example below shows $E^2$dataFlash settings and usable area.

ECC: Disabled
Area used: Block 0 (14000h to 1401Fh)

## 4.1 Application Example Outline

The application example outline is as follows:
(1) Initialize the CPU.
(2) Initialize the $E^2$dataFlash (ECC disabled).
(3) Erase block 0. When an error occurs, retry three times.
(4) If erasing is successful in step (3), set 32-byte data to block 0. If an error occurs, retry once.
(5) If writing is successful in step (4), read 32-byte data in block 0, and store the data to RAM.

If erasing fails during step (3), the program function and read function will not be executed. If writing fails during step (4), the read function will not be executed.

Figure 4.1 shows an Application Example Operation Outline.

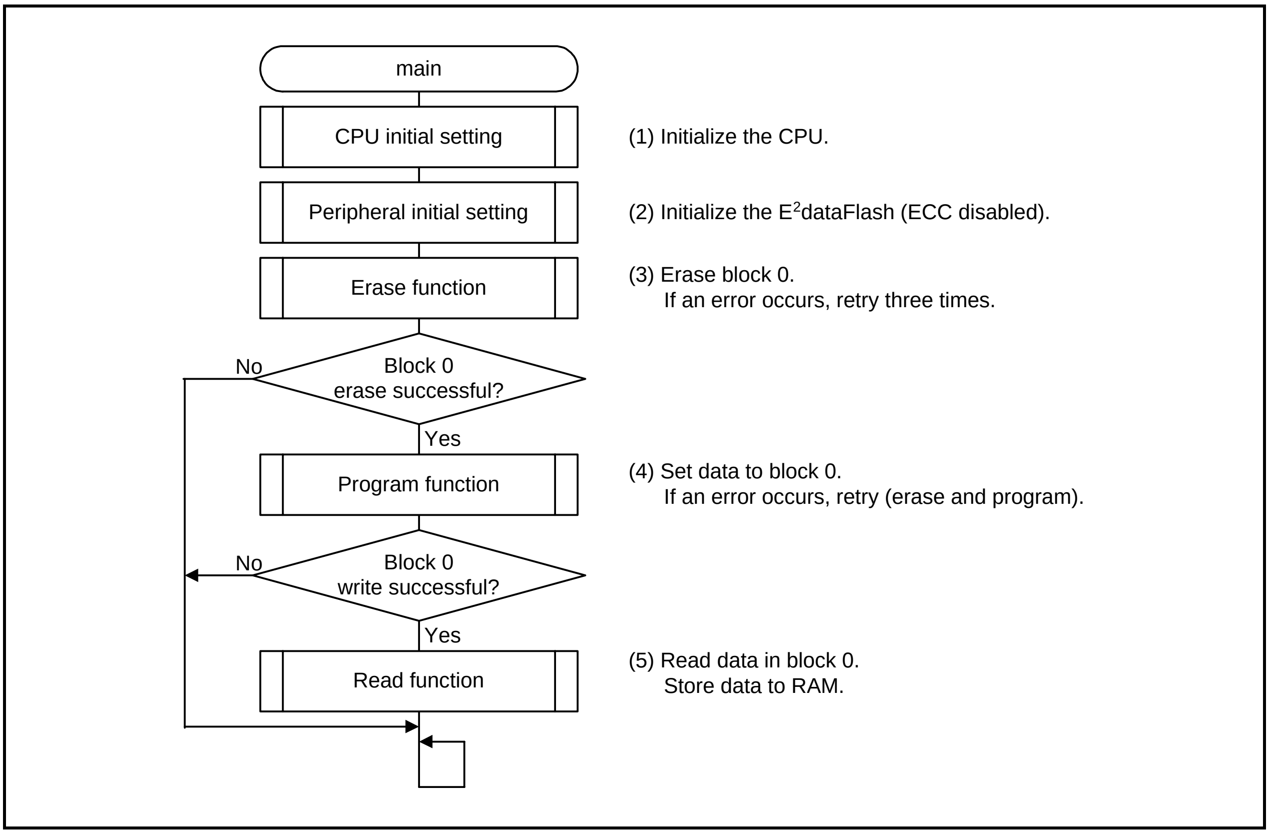



**Figure 4.1 Application Example Operation Outline**

## 4.2 Data Type Definition

This application note uses a data type that is redefined using typedef. Table 4.1 lists the Redefined Data Type.

**Table 4.1 Redefined Data Type**

| Data Type | Original Definition Data Type | Sign | Byte Length |
|---|---|---|---|
| int8_t | signed char | Signed | 1 |
| uint8_t | unsigned char | Unsigned | 1 |
| int16_t | signed short | Signed | 2 |
| uint16_t | unsigned short | Unsigned | 2 |
| int32_t | signed long | Signed | 4 |
| uint32_t | unsigned long | Unsigned | 4 |

## 4.3 Function Tables

| | | |
|---|---|---|
| Declaration | void make_data(uint16_t *data) | |
| Outline | Make write data | |
| Argument | Argument name | Meaning |
| | uint16_t *data | Write data table start address |
| Variable (global) | None | |
| Returned value | None | |
| Function | Creates 32-byte (1 word add space between × and 16) data to write to block 0 in the $E^2$dataFlash.<br>The sample program uses values from 0000h to 000Fh. | |

| | | | |
|---|---|---|---|
| Declaration | uint8_t erase_function(void) | | |
| Outline | Erase function | | |
| Argument | None | | |
| Variable (global) | None | | |
| Returned value | Type | Value | Meaning |
| | uint8_t | SUCCESS | Erase successful |
| | | FAIL | Erase failed |
| Function | Executes the block erase operation.<br>When the erase operation is completed, the returned value will be SUCCESS (erase successful).<br>When an error occurs during the erase operation, if an error still occurs after retrying three times, the returned value will be FAIL (erase failed). | | |

| | | | |
|---|---|---|---|
| Declaration | uint8_t block_erase_operation(uint16_t far* address) | | |
| Outline | Block erase operation | | |
| Argument | Argument name | | Meaning |
| | uint16_t far* address | | The highest-order even address of the block to be erased. |
| Variable (global) | None | | |
| Returned value | Type | Value | Meaning |
| | uint8_t | SUCCESS | Erase successful |
| | | FAIL | Erase failed |
| Function | Executes the block erase command, and erases the specified block.<br>After erasing, executes the full status check. | | |

| | | | |
|---|---|---|---|
| Declaration | uint8_t program_function(uint16_t *program_data) | | |
| Outline | Program function | | |
| Argument | Argument name | Meaning | |
| | uint16_t *program_data | Write data address | |
| Variable (global) | None | | |
| Returned value | Type | Value | Meaning |
| | uint8_t | SUCCESS | Write successful |
| | | FAIL | Write failed |
| Function | Executes program operation. When the write operation is completed, the returned value will be SUCCESS (write successful). If an error occurs during the write operation, the erase function is executed, and the program operation is executed again. Even if an error still occurs, the returned value will be FAIL (write failed). | | |

| | | | |
|---|---|---|---|
| Declaration | uint8_t program_operation(uint16_t far* address, uint16_t program_data) | | |
| Outline | Program operation | | |
| Argument | Argument name | Meaning | |
| | uint16_t far* address | Write address to E2dataFlash | |
| | uint16_t program_data | 2-byte write data | |
| Variable (global) | None | | |
| Returned value | Type | Value | Meaning |
| | uint8_t | SUCCESS | Write successful |
| | | FAIL | Write failed |
| Function | Executes the program command, and writes data to the specified address.<br>After writing, executes the full status check. | | |

| | | |
|---|---|---|
| Declaration | void read_function(void) | |
| Outline | Read function | |
| Argument | None | |
| Variable (global) | Variable name | Contents |
| | uint16_t read_buff[i] | Array to store the read data |
| Returned value | None | |
| Function | Executes the read operation, and reads the data in block 0.<br>Stores the read data in RAM. | |

| | | | |
|---|---|---|---|
| Declaration | uint16_t read_operation(uint16_t far* address) | | |
| Outline | Read operation | | |
| Argument | Argument name | | Meaning |
| | uint16_t far* address | | Read address of E2dataFlash |
| Variable (global) | None | | |
| Returned value | Type | Value | Meaning |
| | uint16_t | 0x0000 to 0xFFFF | 2-byte read data |
| Function | Executes the read command, and reads the data in the specified address.<br>Returned value will be 2-byte data read from the E2FD register. | | |

| | |
|---|---|
| Declaration | void clear_status(void) |
| Outline | Clear status operation |
| Argument | None |
| Variable (global) | None |
| Returned value | None |
| Function | Executes the clear status register command. |

| | | | |
|---|---|---|---|
| Declaration | uint8_t full_status_check(void) | | |
| Outline | Full status check | | |
| Argument | None | | |
| Variable (global) | None | | |
| Returned value | Type | Value | Meaning |
| | uint8_t | COMPLETE | No error |
| | | CMD_SEQ_ERR | Command sequence error |
| | | PRG_ERR | Program error |
| | | ERS_ERR | Erase error |
| Function | Executes the full status check, and returns the result. | | |

## 4.4 Sample Program Flowchart

Figure 4.2 to Figure 4.11 show sample program flowcharts.

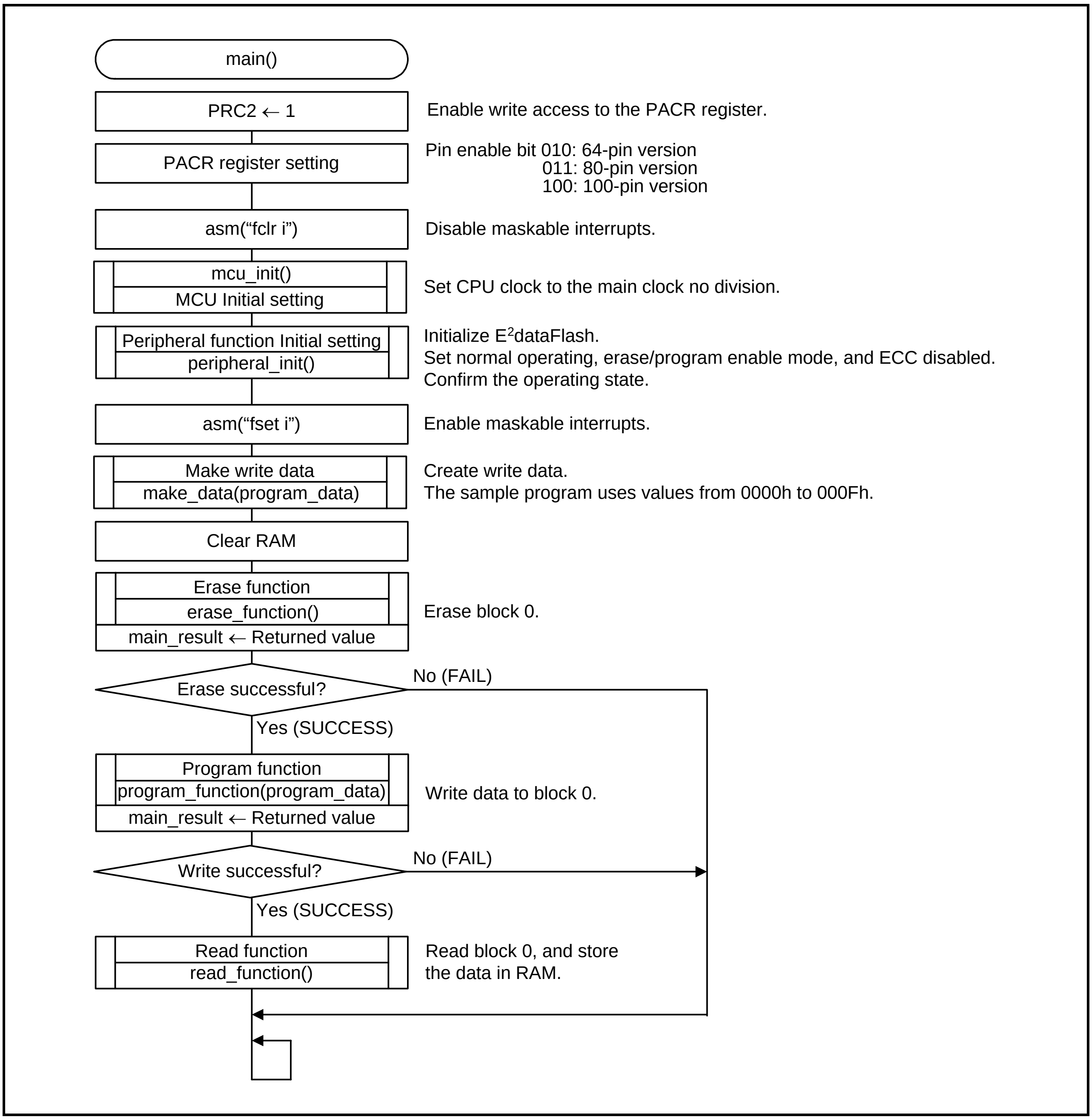



**Figure 4.2 Main Program**

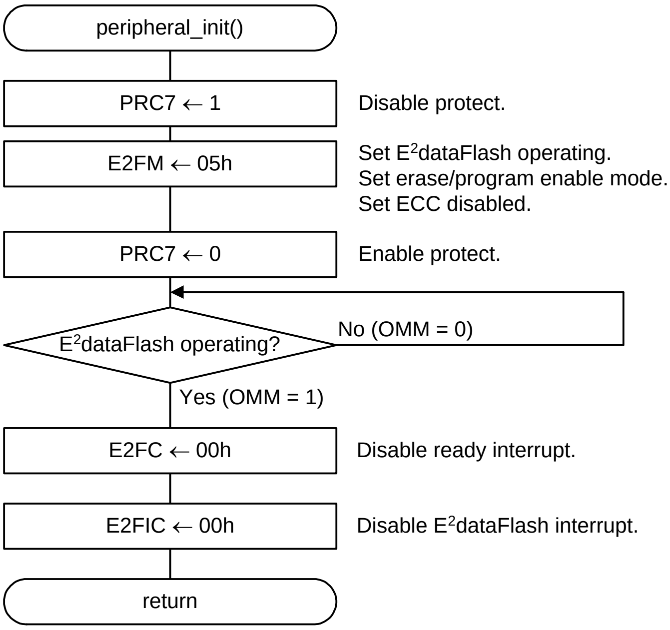



**Figure 4.3 Peripheral Function Initial Setting**

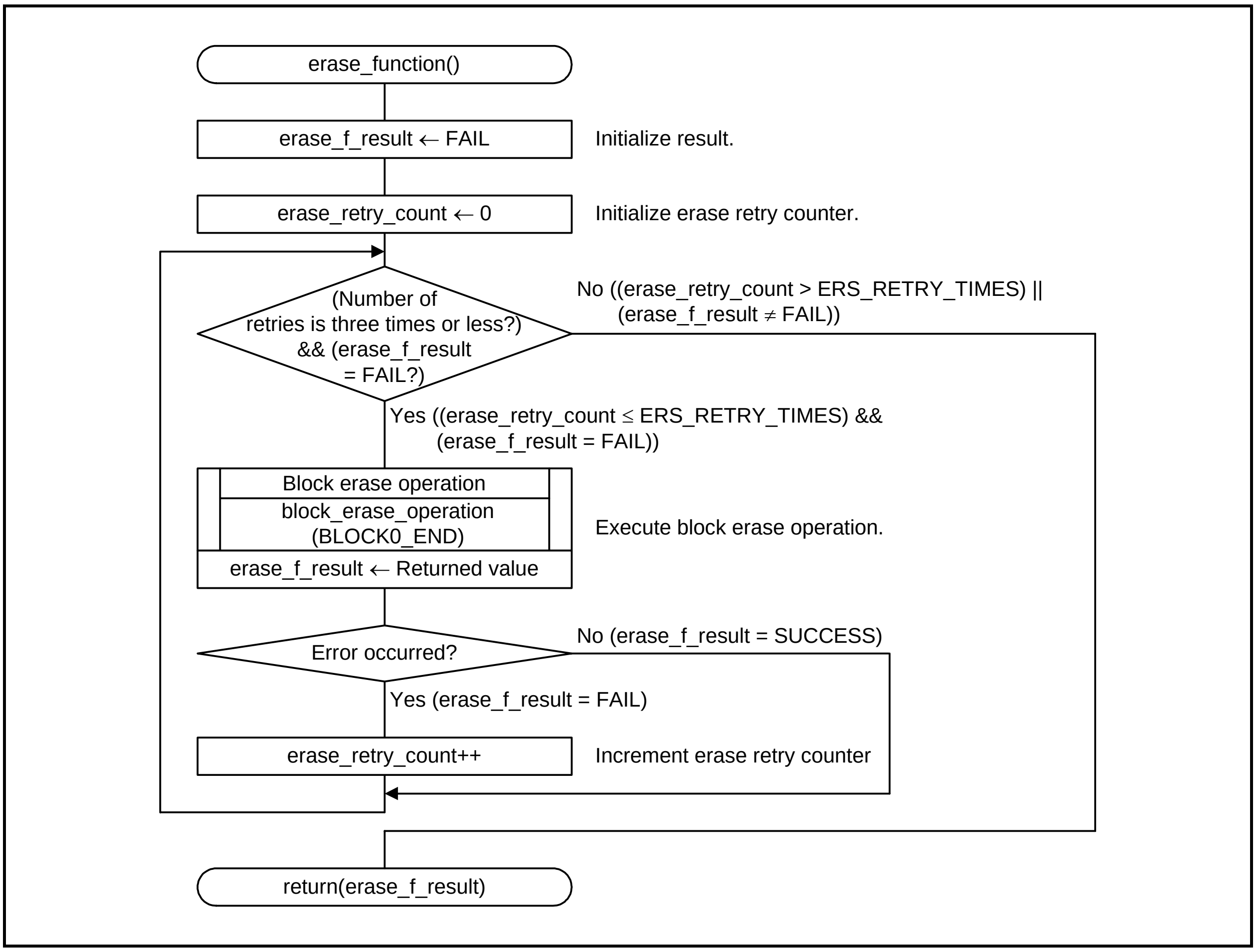



**Figure 4.4 Erase Function**

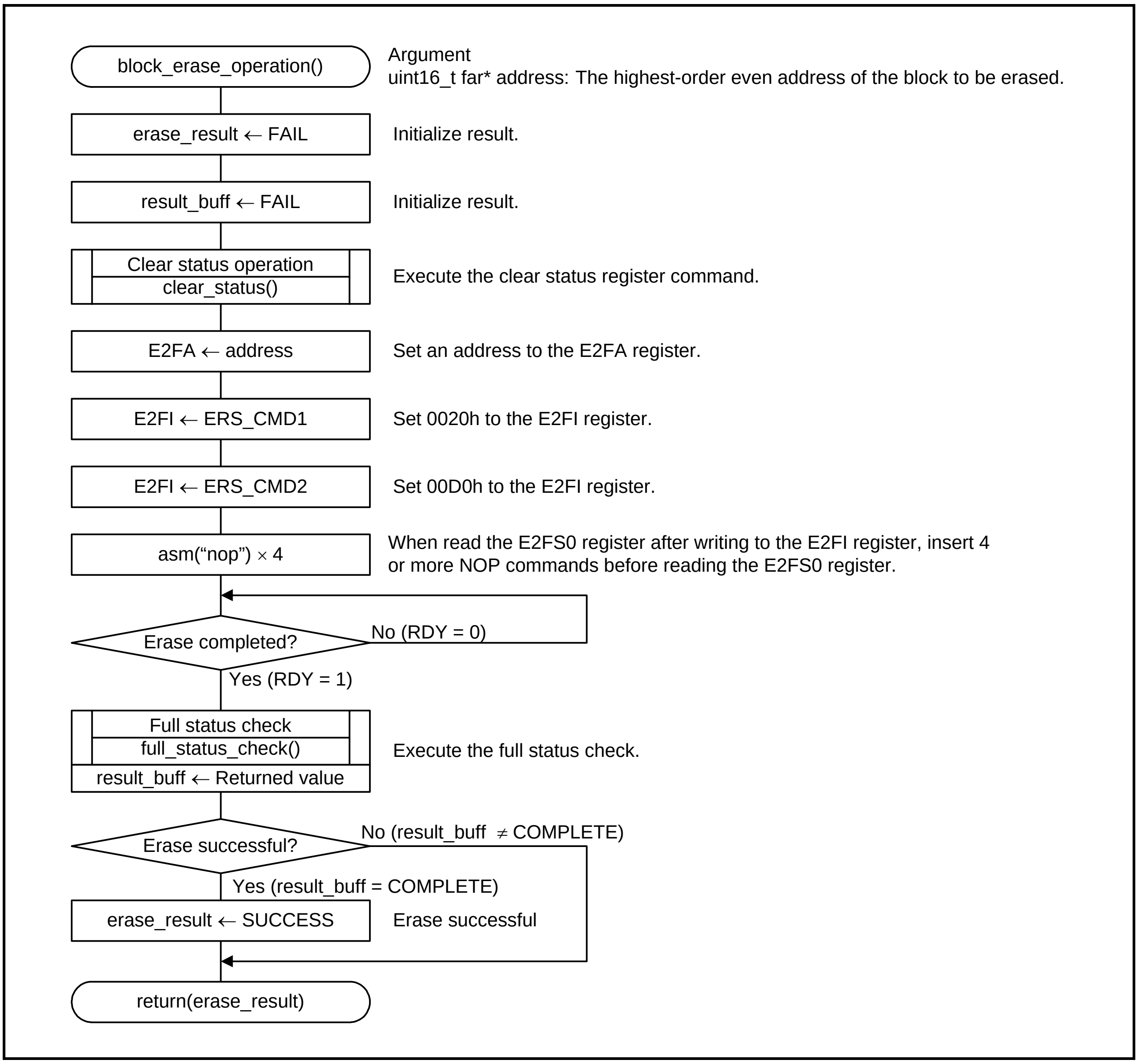



**Figure 4.5 Block Erase Operation**

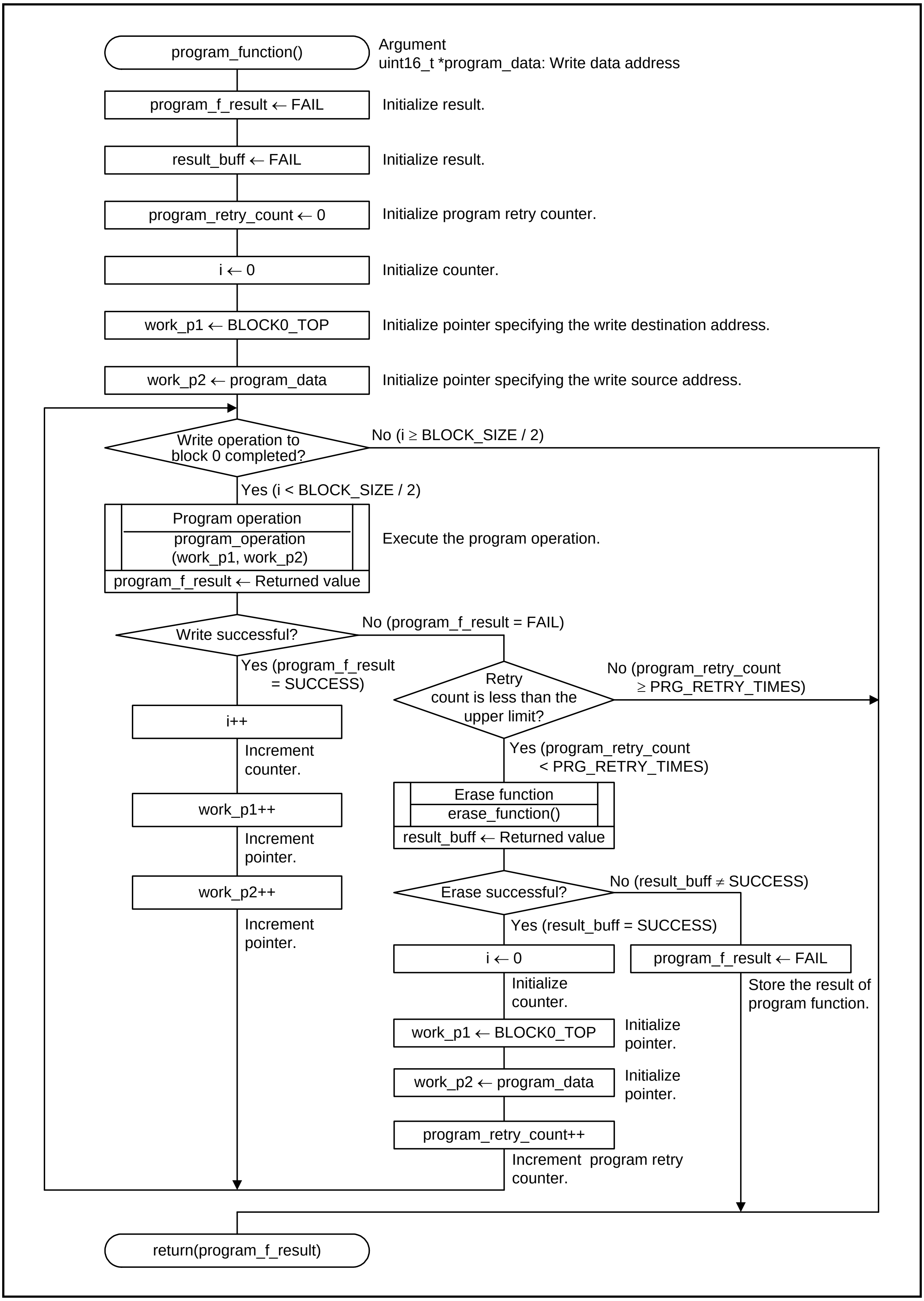



**Figure 4.6 Program Function**

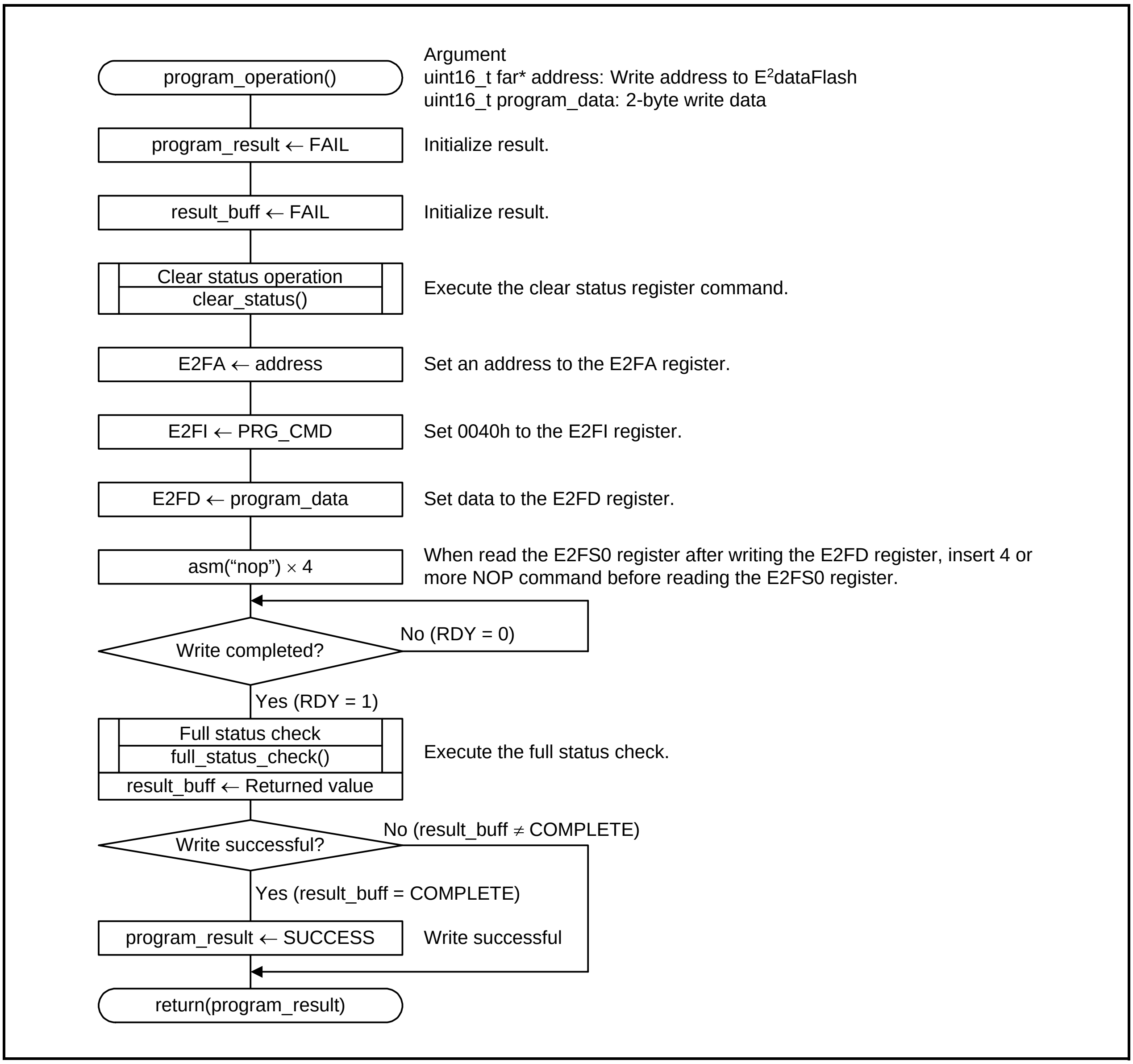



**Figure 4.7 Program Operation**

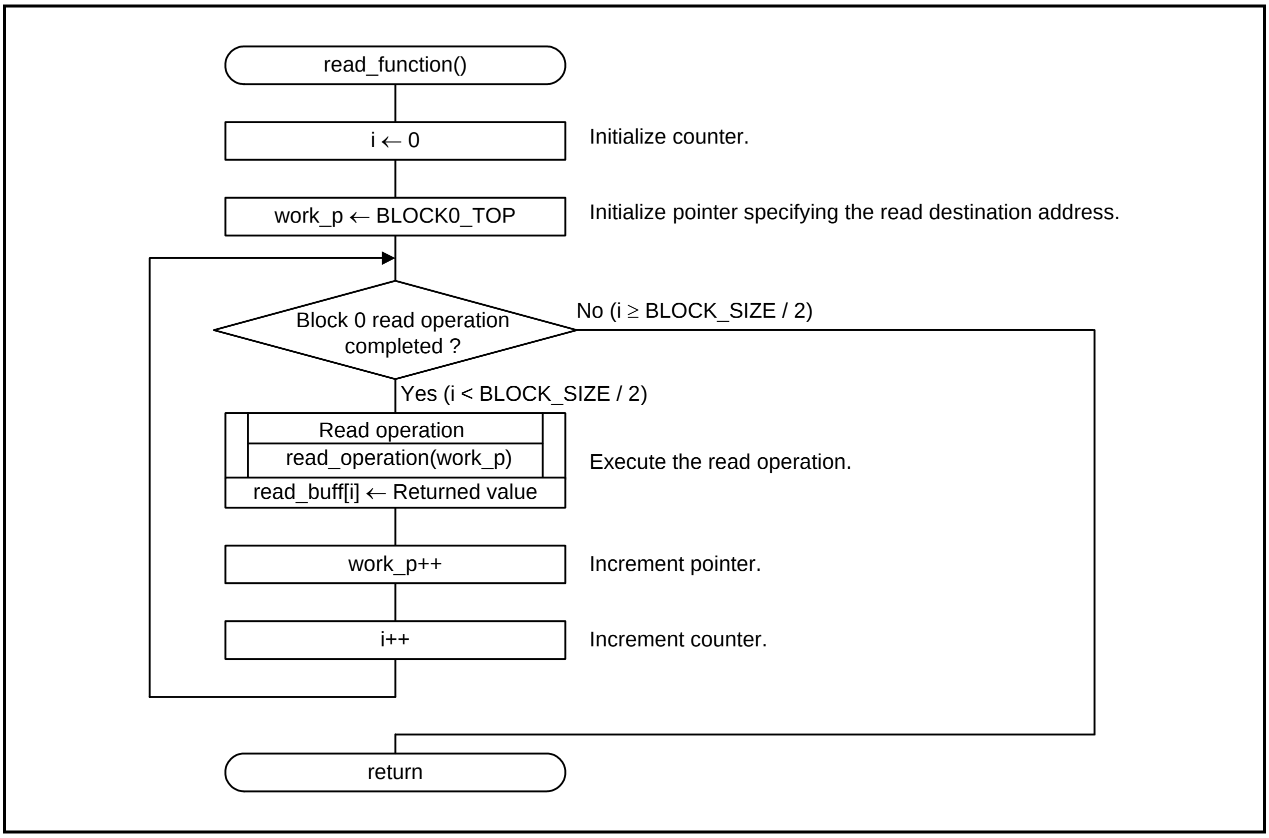



**Figure 4.8 Read Function**

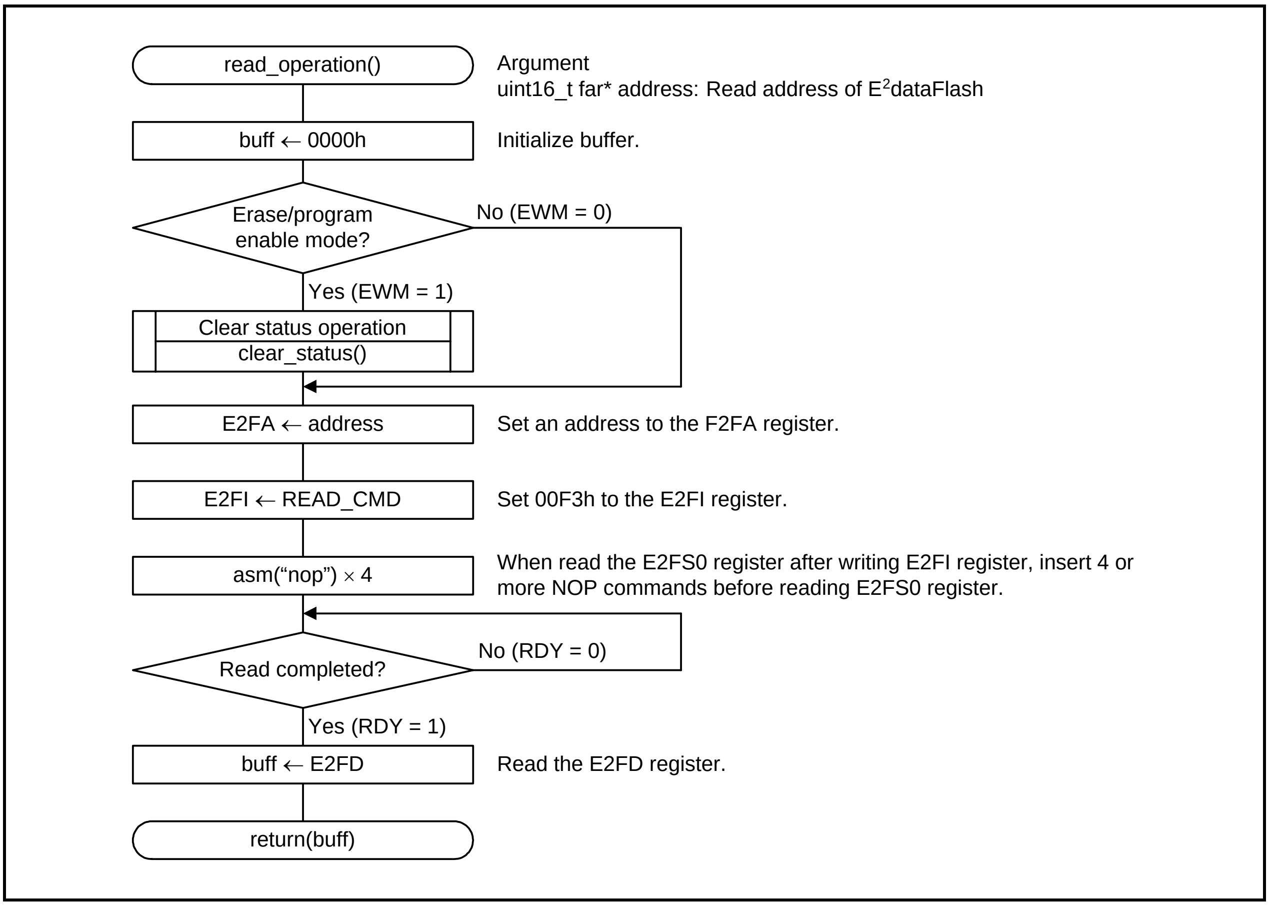



**Figure 4.9 Read Operation**

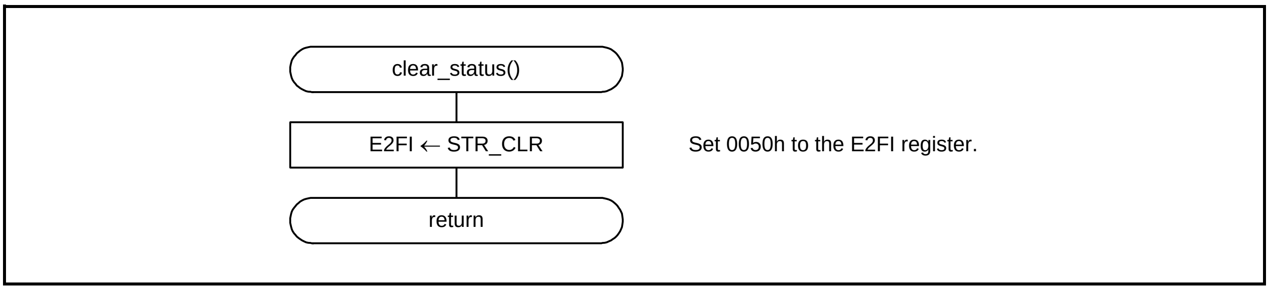



**Figure 4.10 Clear Status Operation**

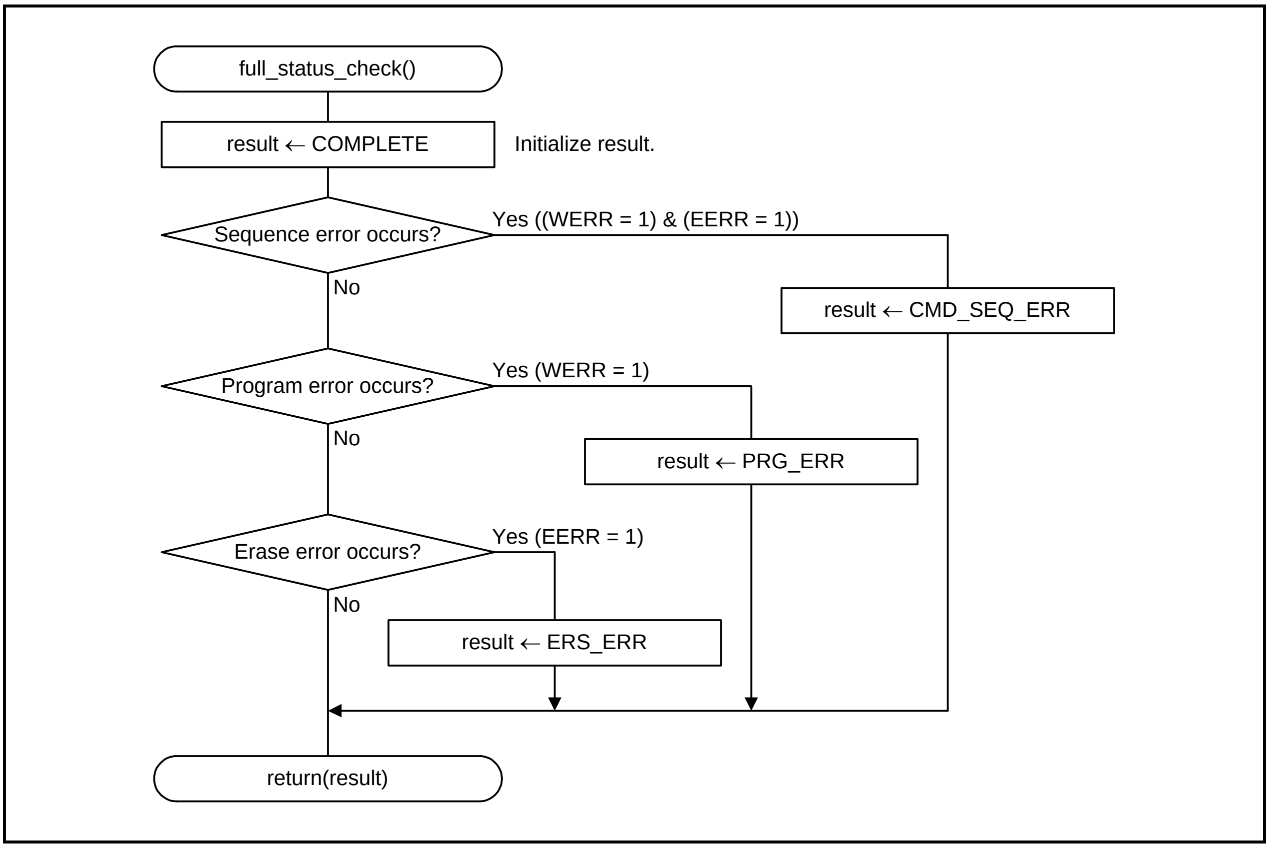



**Figure 4.11 Full Status Check**

# 5. Sample Program

A sample program can be downloaded from the Renesas Electronics website.
To download, click “Application Notes” in the left-hand side menu of the M16C Family page.

# 6. Reference Documents

Application Note
M16C/5M Group User’s Manual: Hardware Rev.1.01
The latest version can be downloaded from the Renesas Electronics website.

Technical News/Technical Update
The latest information can be downloaded from the Renesas Electronics website.

C Compiler Manual
M16C Series, R8C Family C Compiler Package V.5.45
C Compiler User’s Manual Rev.2.00
The latest version can be downloaded from the Renesas Electronics website.

# Website and Support

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

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

| Revision History | M16C/5M Group<br>Using the $E^2$PROM Emulation Data Flash |
|---|---|

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Dec 15, 2010 | — | First edition issued |
| | | | |

## 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 part number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different part numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different part numbers, implement a system-evaluation test for each of the products.

**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.**
7F, 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 Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141