# H8/300H Tiny Series

## User-Mode Flash Memory Reprogramming

### Introduction

Flash memory reprogramming is performed in user mode using asynchronous serial communication.

### Target Device

H8/3664F

### Contents

## 1. Specifications

- Flash memory is reprogrammed in user mode.
- Reprogram data in the flash memory of the master device are programmed into the flash memory of the slave device.
- Reprogram data is transferred by asynchronous serial communications using SCI3.
- Switch 0 (SW0) is connected to the $\overline{\text{IRQ0}}$ pin of the master device.
- When switch 0 (SW0) of the master is turned on, a flash memory reprogramming start command is sent from the master to the slave and the reprogramming of the slave-side flash memory starts.
- During flash memory reprogramming, P74 is high and P75 is low on both the master and slave; and after the flash memory reprogramming is finished, P74 goes low and P75 goes high.

Figure 1 shows an example configuration of an on-board reprogramming circuit.

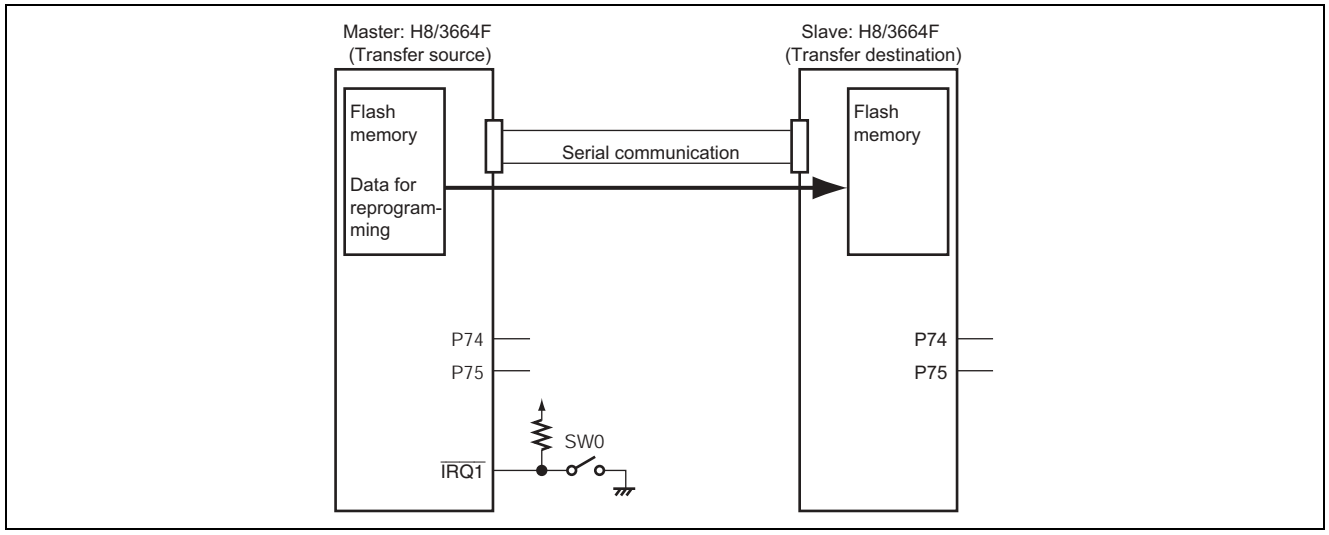



**Figure 1 Example Configuration of On-Board Reprogramming Circuit**

## 1.1 On-board Programming Operating Conditions

- CPU operation: User mode
- Operating voltage: 5.0 V
- Operating frequency: 16 MHz

## 1.2 On-Board Programming Mode

- User Mode
  It is assumed that a program/erase control program, reprogram-start command reception program, and RAM transfer program have already been programmed into the flash memory of the slave device in boot mode or writer mode.

## 1.3 Programming Method

The flash memory of the slave device is reprogrammed with the reprogram data received from the transfer source.

Communication with the transfer source is performed by asynchronous serial communication using SCI3. The master device is the transfer source, and the slave device is the transfer destination.

## 1.4 Flowchart of Reprogramming Procedure

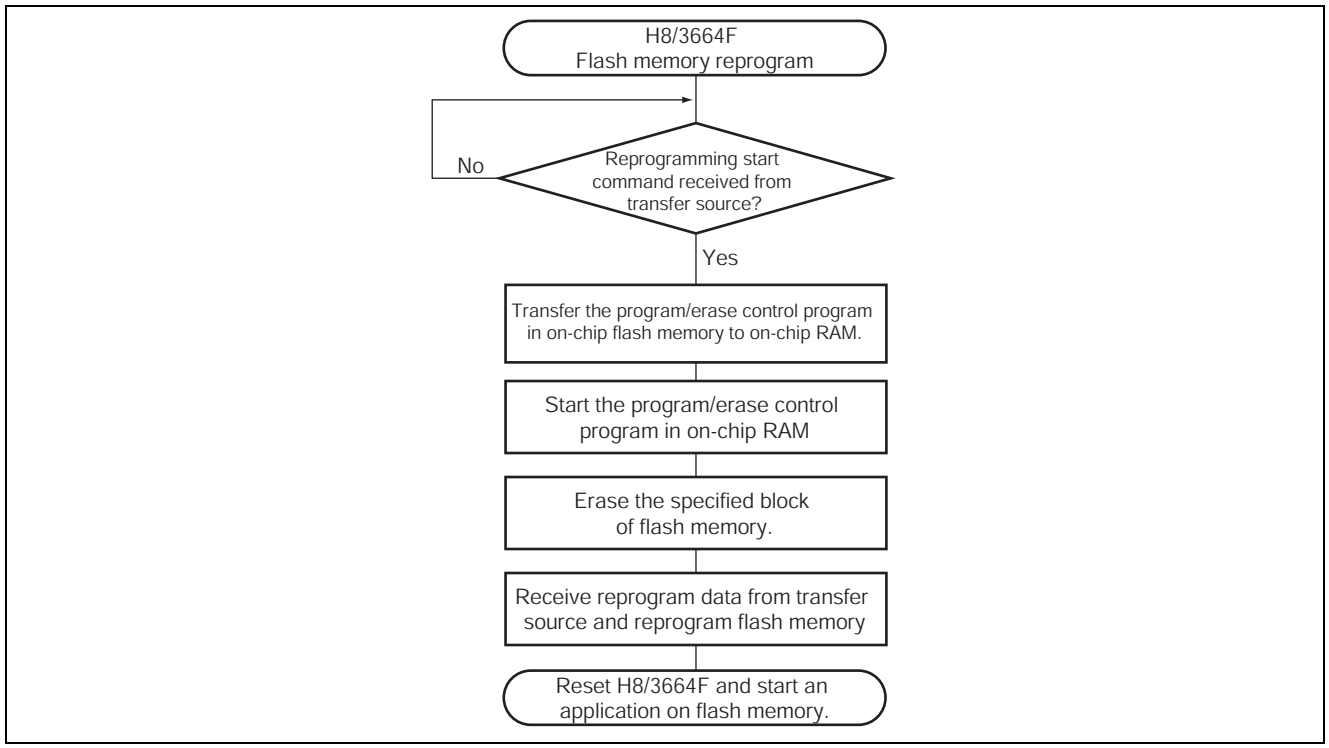



**Figure 2 User-Mode Programming Procedure (Slave Device)**

## 1.5 Diagram of Master-Slave Connection

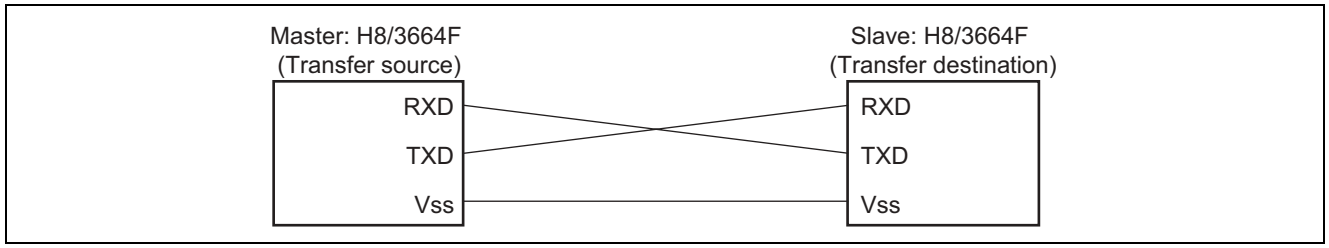



**Figure 3 Master-Slave Connection**

## 1.6 Communication Specifications

**Table 1 Communication Specifications**

| Item | Setting |
|---|---|
| Transfer speed | 38400 bps |
| Transfer method | Asynchronous serial communications |
| Data bits | 8 |
| Stop bit | 1 |
| Parity | None |

## 1.7 Communication Commands

**Table 2 Communication Commands**

| Communication Command | Description |
|---|---|
| H'00 | Normal communication (command name: OK command) |
| H'01 | Communication error (command name: NG command) |
| H'11 | Transmission start request |
| H'55 | Reprogram start command |
| H'77 | Erase command |
| H'88 | Program command |

## 1.8 Memory Allocation

Table 3 shows the erase blocks in flash memory of the H8/3664F.

**Table 3 Erase Blocks in Flash Memory**

| Block No. | Block (Size) | Address |
|---|---|---|
| 0 | EB0 (1 kbyte) | H'0000 to H'03FF |
| 1 | EB1 (1 kbyte) | H'0400 to H'07FF |
| 2 | EB2 (1 kbyte) | H'0800 to H'0BFF |
| 3 | EB3 (1 kbyte) | H'0C00 to H'0FFF |
| 4 | EB4 (28 kbytes) | H'1000 to H'7FFF |

Figure 4 shows the memory maps during normal operation of the H8/3664F and during flash memory reprogramming.

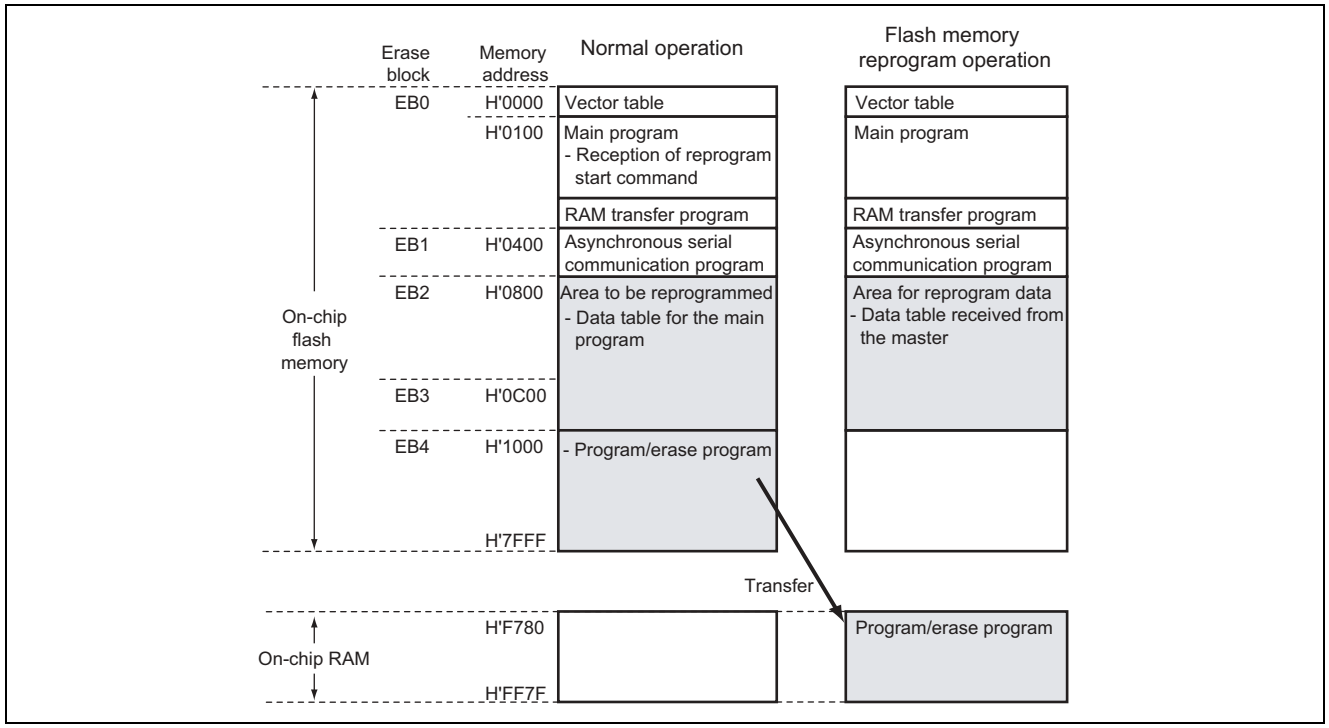



**Figure 4 Memory Maps (Slave Device)**

# 2. Description of Functions

## 2.1 Description of Registers

- Flash Memory Control Register 1 (FLMCR1)
  FLMCR1 causes the flash memory to make transitions to program mode, program-verify mode, erase mode, and erase-verify mode.
- Flash Memory Control Register 2 (FLMCR2)
  FLMCR2 indicates the state of reprogramming/erasure of flash memory. FLMCR2 is a read-only register and should not be written to.
- Block Specification Register 1 (EBR1)
  EBR1 specifies an erase block in flash memory. When the SWE bit in FLMCR1 is 0, EBR1 is initialized to H'00. Two or more bits of this register should not be set to 1 at a time. If set, EBR1 will be cleared to 0 automatically.
- Flash Memory Power Control Register (FLPWCR)
  FLPWCR selects whether the flash memory is placed in power-down mode when the LSI enters sub-active mode. In power-down mode, the power supply circuitry is partially stopped, but reading of flash memory is possible.
- Flash Memory Enable Register (FENR)
  FENR controls access to the flash memory control registers FLMCR1, FLMCR2, EBR1, and FLPWCR from the CPU.

## 2.2 Programming and Erasure in User Mode

Also in user mode, a desired block can be erased or reprogrammed on-board by branching to a porgram/erase program prepared by the user. The branching condition settings and the means for supplying on-board reprogramming data must be prepared by the user. Also, where necessary, a porgram/erase program or a program for loading the porgram/erase program from an external source must be programmed into a part of flash memory. Since the flash memory cannot be read during reprogramming or erasre, the program/erase program should be transferred to and executed from on-chip RAM, as is done in boot mode. A program/erase program should be prepared as described in section 2.3, Program/Erase Program.

## 2.3 Program/Erase Program

On-board flash memory programming and erasure are implemented by software using the CPU. The flash memory enters program mode, program-verify mode, erase mode, or erase-verify mode according to the setting of FLMCR1. A boot-mode program control program and user-mode program/erase program perform programming and erasure by combining the above modes. Flash memory should be programmed and erased in accordance with the following notes on "program/program-verify" and "erase/erase-verify", respectively.

- **Program/Program-Verify**
  (1) Programming must be performed on the erased part of flash memory; do not program the addresses that have already been programmed.
  (2) Programming should be performed in 128-byte units. Even when data less than 128 bytes are to be programmed, it is necessary to transfer 128 bytes of data to the flash memory. Addresses for which there is no data for programming should be programmed with H'FF.
  (3) The following data storage areas should be allocated in RAM: a 128-byte programming data area, a 128-byte reprogramming data area, and a 128-byte additional programming data area. Computation of reprogramming data and additional programming data should be as shown in tables 4 and 5, respectively.
  (4) 128 bytes of data should be transferred continuously in byte units from the reprogramming data area or additional programming data area to the flash memory. The program address and 128 bytes of data are latched in the flash memory. The lower 8 bits of the first address of the transfer destination area in the flash memory should be either H'00 or H'80.
  (5) The time during which the P bit is set is the programming time. Programming times are as shown in table 6.
  (6) The watchdog timer is set to prevent excessive programming resulting from program runaway or other causes. The overflow period should be set to approx. 6.6 ms.
  (7) For a dummy write to a verify address, one byte data, H'FF, should be written to an address whose lowermost bit is B'0. The verify data is read as a word from the address to which a dummy write was performed.
  (8) The program/program-verify sequence should not be repeated more than 1000 times for the same bit.

**Table 4 Reprogramming Data Computation**

| Programmed Data | Result of Verification | Reprogramming Data | Remarks |
|---|---|---|---|
| 0 | 0 | 1 | Programming completed |
| 0 | 1 | 0 | Bit requires reprogramming |
| 1 | 0 | 1 | — |
| 1 | 1 | 1 | Remains in an erased state. |

**Table 5 Additional Programming Data Computation**

| Reprogramming Data | Result of Verification | Additional Programming Data | Remarks |
|---|---|---|---|
| 0 | 0 | 0 | Bit requires additional programming |
| 0 | 1 | 1 | Additional programming is not required. |
| 1 | 0 | 1 | Additional programming is not required. |
| 1 | 1 | 1 | Additional programming is not required. |

**Table 6 Programming Time**

| n (Number of Programming Operations) | Programming | Additional Programming | Remarks |
|---|---|---|---|
| 1 to 6 | 30 [µs] | 10 [µs] | — |
| 7 to 1000 | 200 [µs] | — | — |

- **Erase/Erase-Verify**
  (1) There is no need to pre-write (seting erase block data to all 0) prior to erasure.
  (2) Erasure is performed in block units. Only one erase block should be be selected using the block specification register 1 (EBR1). Even when multiple blocks are to be erased, they must be erased in sequence one block at a time.
  (3) The time during which the E bit is set is the erase time.
  (4) The watchdog timer is set to prevent excessive erasure resulting from program runaway or other causes. The overflow period should be set to approx. 19.8 ms.
  (5) For a dummy write to a verify address, one byte of H'FF should be written to an address whose lowermost bit is B'0. The verify data is read as a word from the address to which a dummy write was performed.
  (6) If the read data is found to have not been erased, erase mode should be set again, and the erase/erase-verify sequence should be repeated in the same way. However, the number of repetitions should not exceed 100.

- **Interrupts During Programming/Erasure of Flash Memory**
  During flash memory programming/erasure or execution of the program/erase control program, all interrupt requests, including NMI, should be disabled for the following reason.
  (1) If an interrupt occurs during programming/erasure, operation conforming to the normal program/erase algorithm cannot be guaranteed.

# 3. Description of Operation

## 3.1 Normal Operation

1. The normal application normally accesses a data table in flash memory; the data table is received from the master and reprogrammed.
2. In the flash memory of the slave device, a program/erase control program, reprogram-start command reception program, and RAM transfer program must be programmed in advance.
3. Data communication between master and slave is performed by asynchronous serial communications using SCI3.

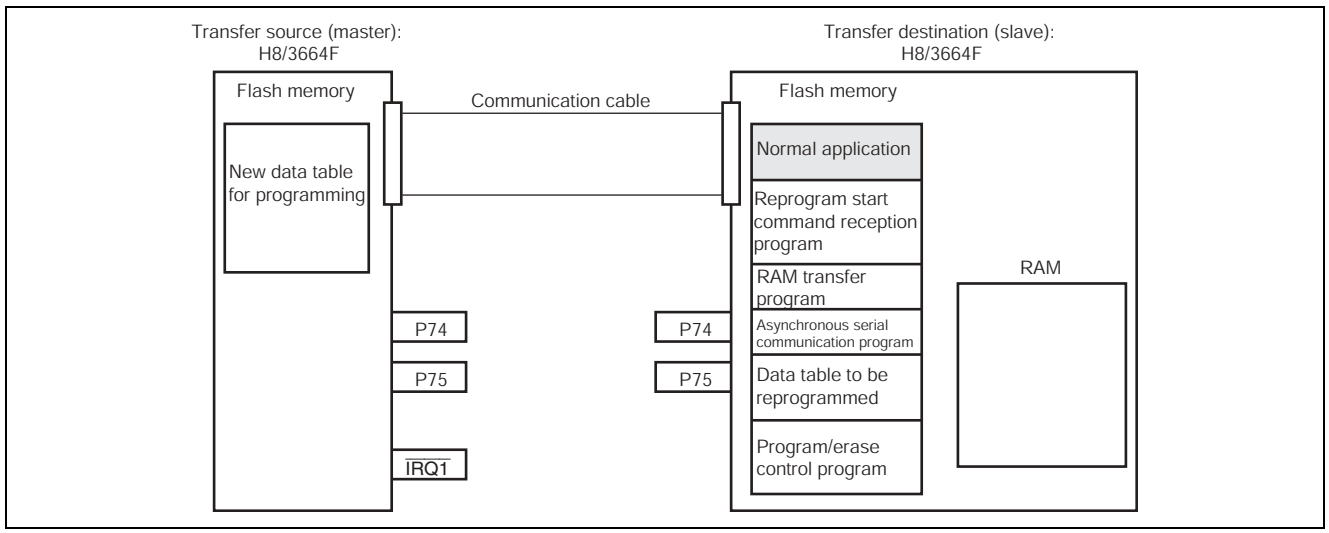



**Figure 5 Normal Operation**

## 3.2 Preparations for On-board Reprogramming

1. When a falling edge is detected on the master's $\overline{\text{IRQ1}}$ pin, the master sends a reprogram start command H'55.
2. At this time on the master device, P74 is high and P75 is low.

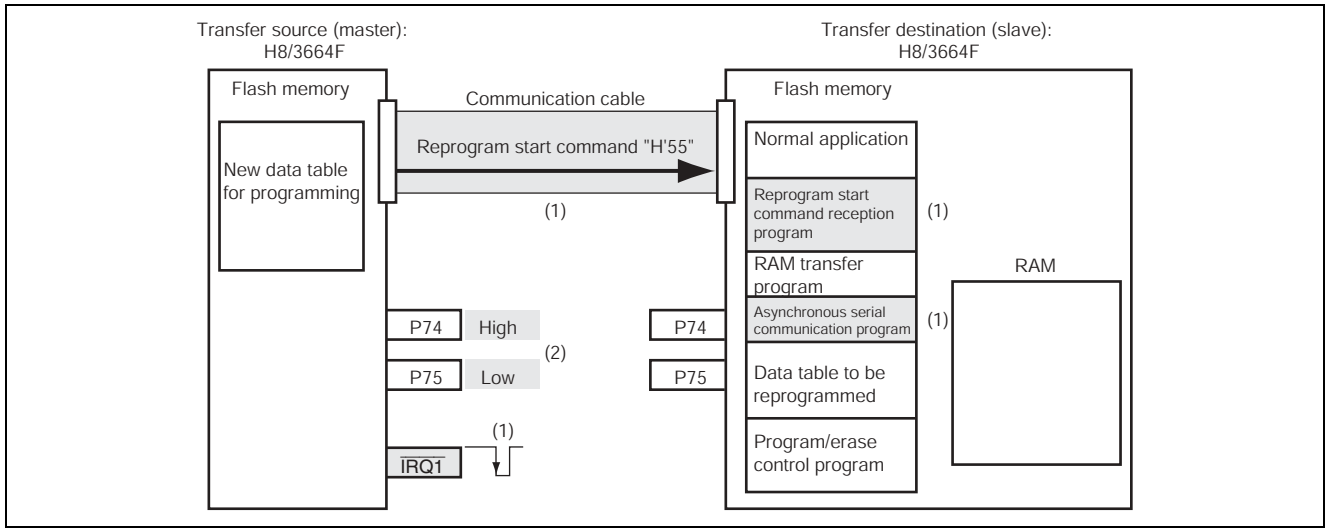



**Figure 6 Preparations for On-board Reprogramming**

## 3.3 Starting On-Board Reprogramming

1. On receiving the command H'55, the slave device starts the RAM transfer program, and transfers the program/erase control program to the on-chip RAM.
2. At this time on the slave device, P74 is high and P75 is low.

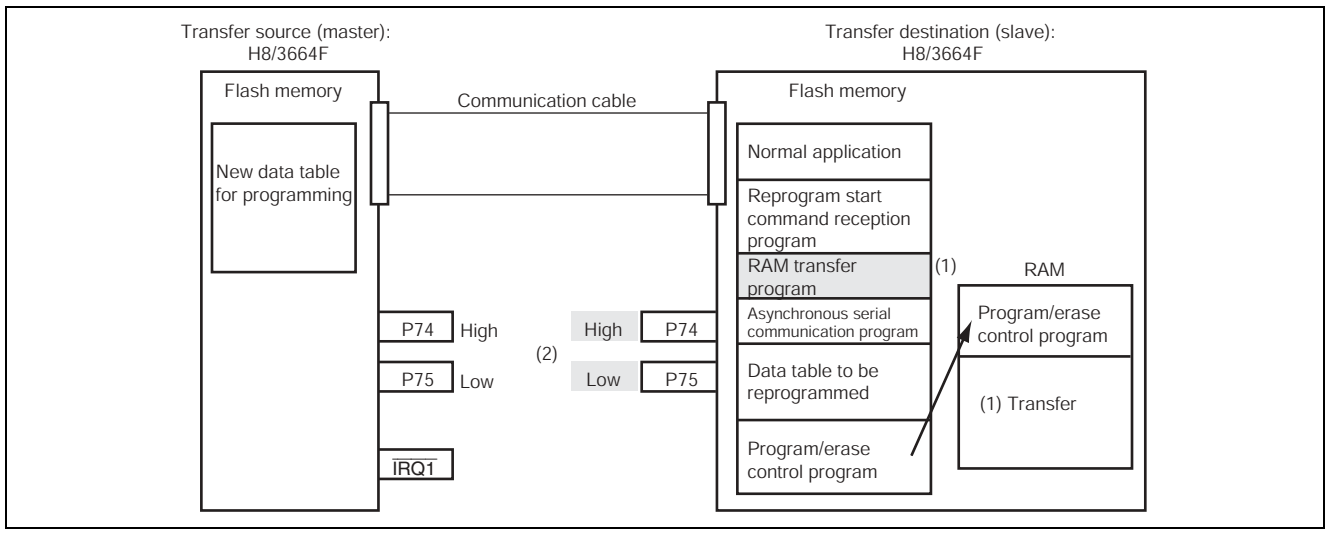



**Figure 7 Starting On-Board Reprogrammig**

## 3.4 Running the Program/Erase Control Program

1. After the transfer by the RAM transfer program has ended, execution branches to the program/erase control program in RAM.

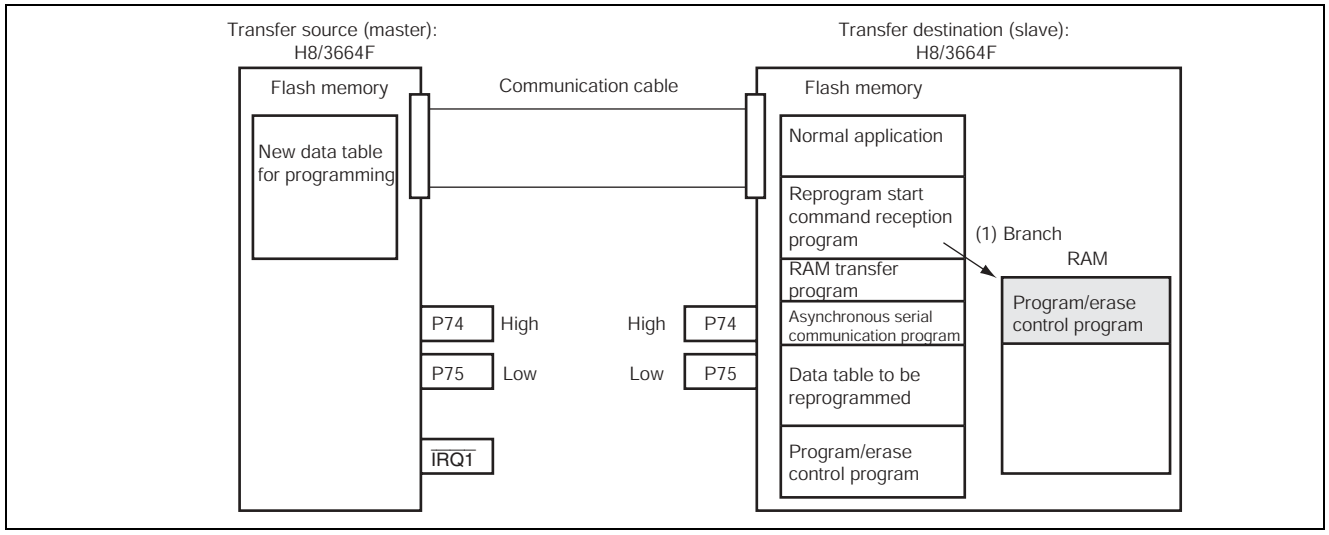



**Figure 8 Running the Program/Erase Control Program**

## 3.5 Flash Memory Erasure

1. The slave receives an erase command H'77 from the master.
2. The program/erase control program erases the specified block of the flash memory.

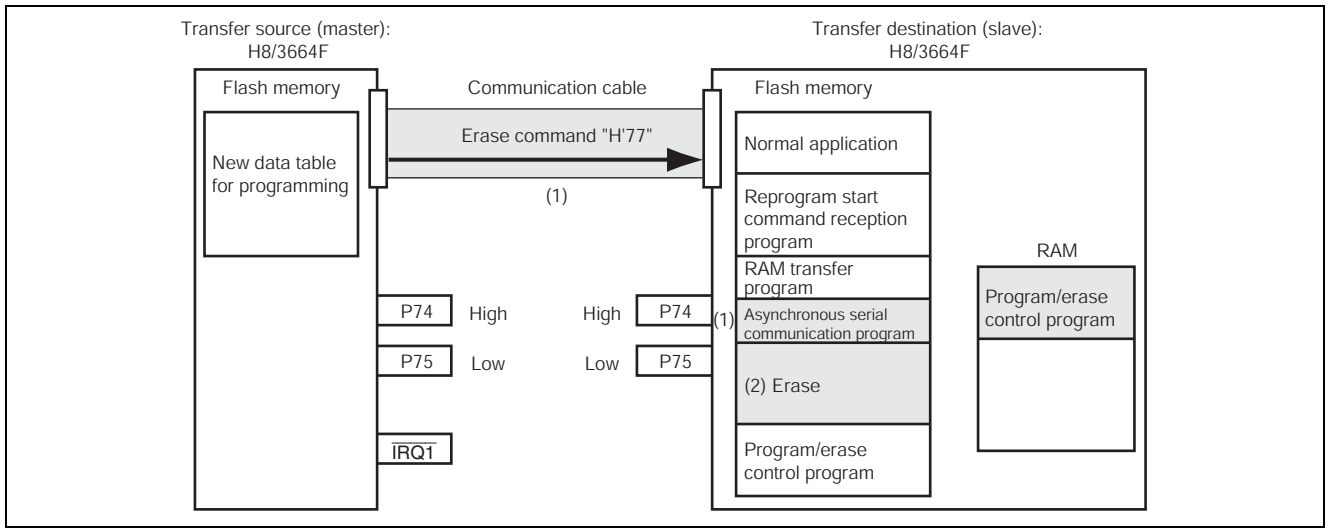



**Figure 9 Flash Memory Erasure**

## 3.6 Flash Memory Programming

1. The slave receives a program command H'88 from the master.
2. The program/erase control program receives a new data table from the master, and programs it into the flash memory.
3. After the programming is finished, P74 is low and P75 is high for both master and slave.

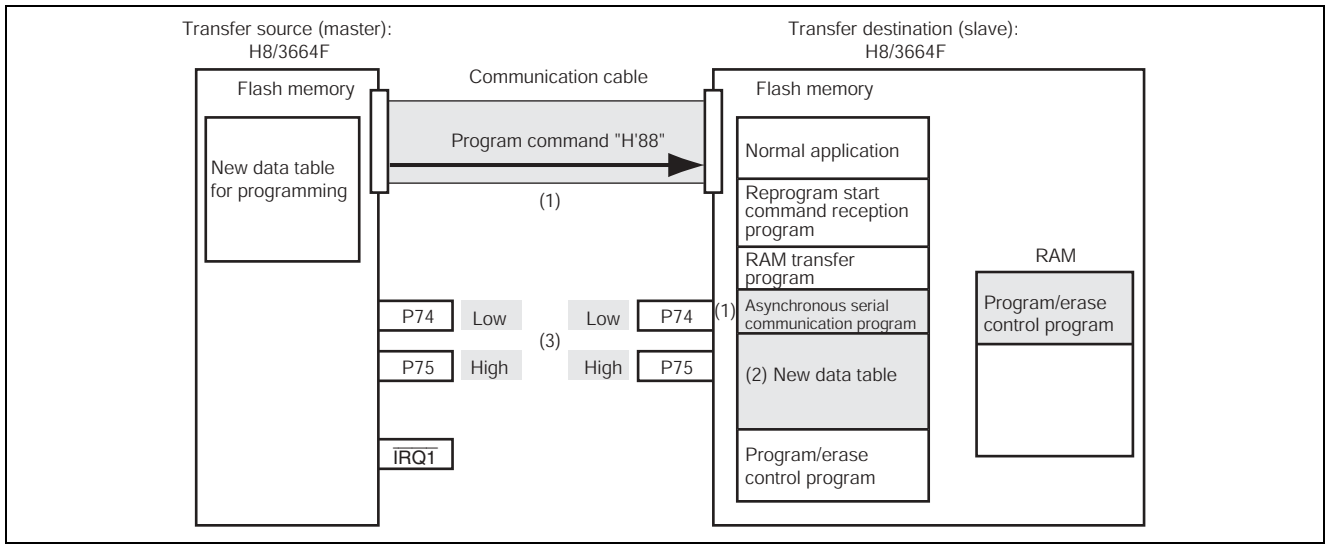



**Figure 10 Flash Memory Programming**

## 3.7 Running the Normal Application Program

1. The slave H8/3664F is reset, and the normal application that accesses the new data table is started.

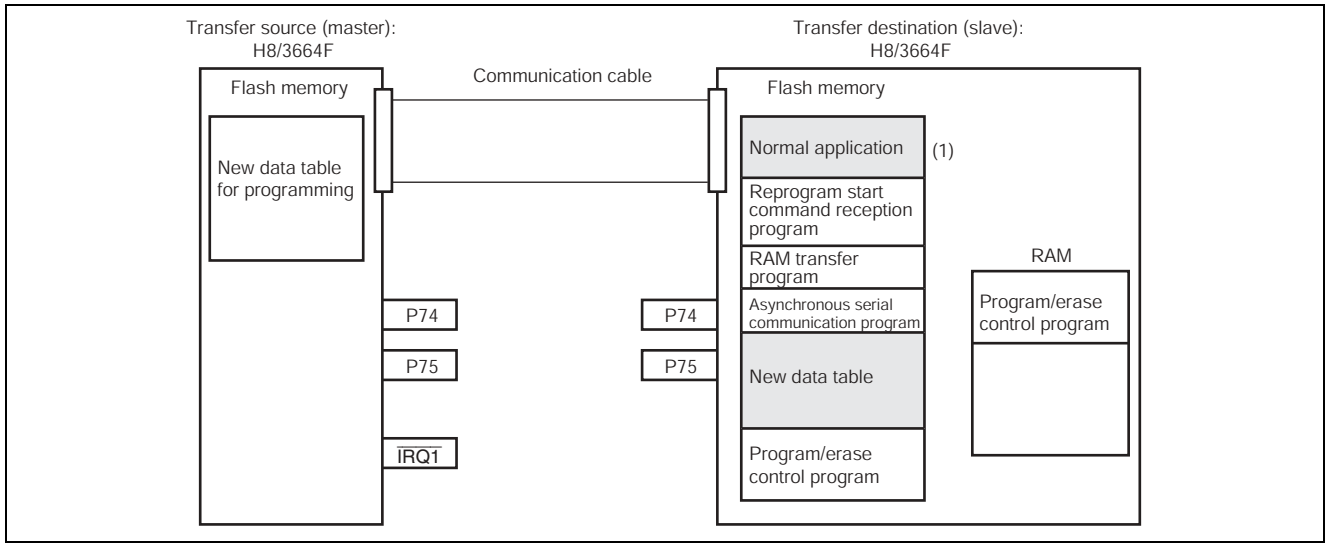



**Figure 11 Running the Normal Application Program**

# 4. Sequence Diagrams

## 4.1 Normal Operation

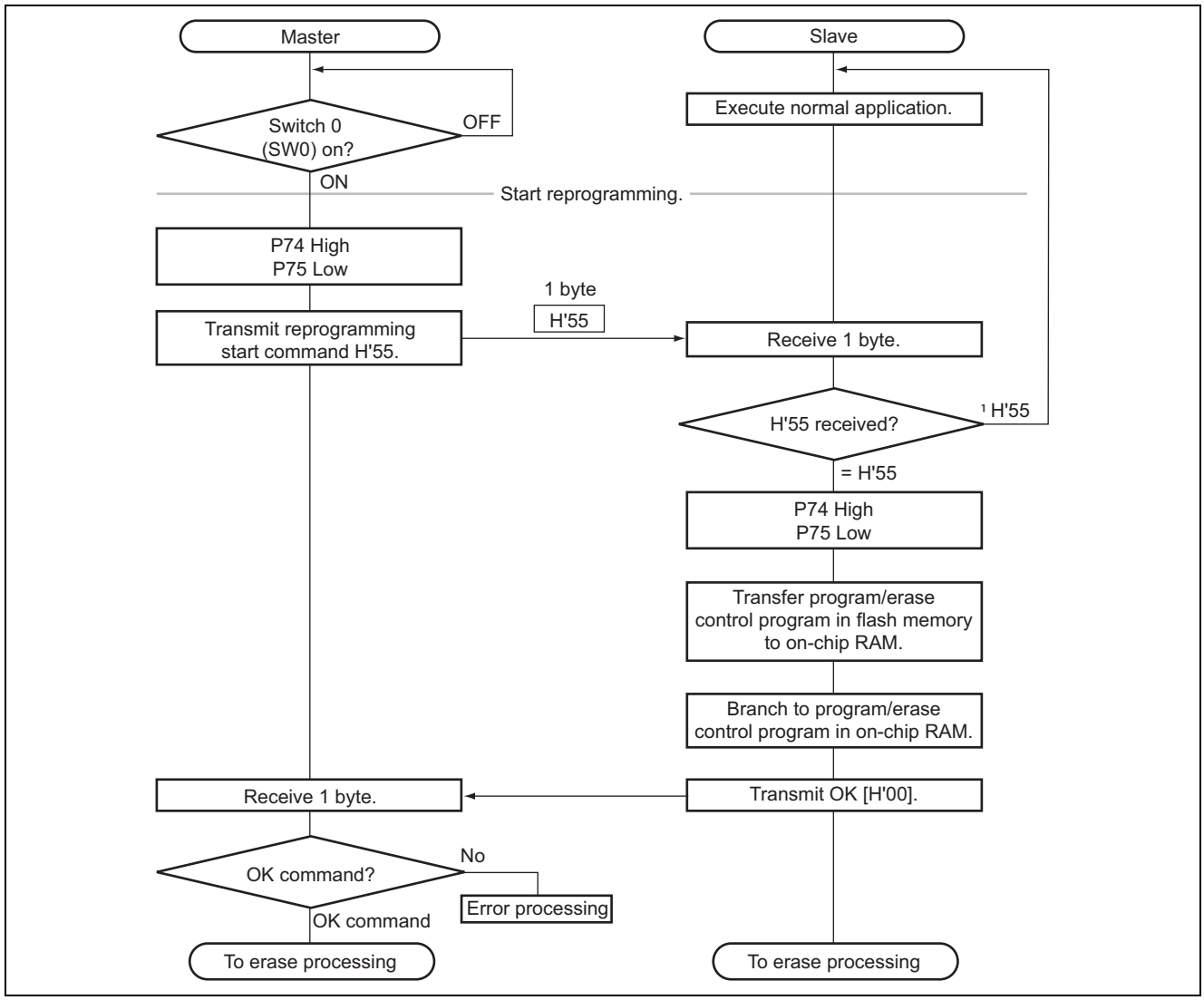



**Figure 12 Normal Operation**

## 4.2 Erase Processing

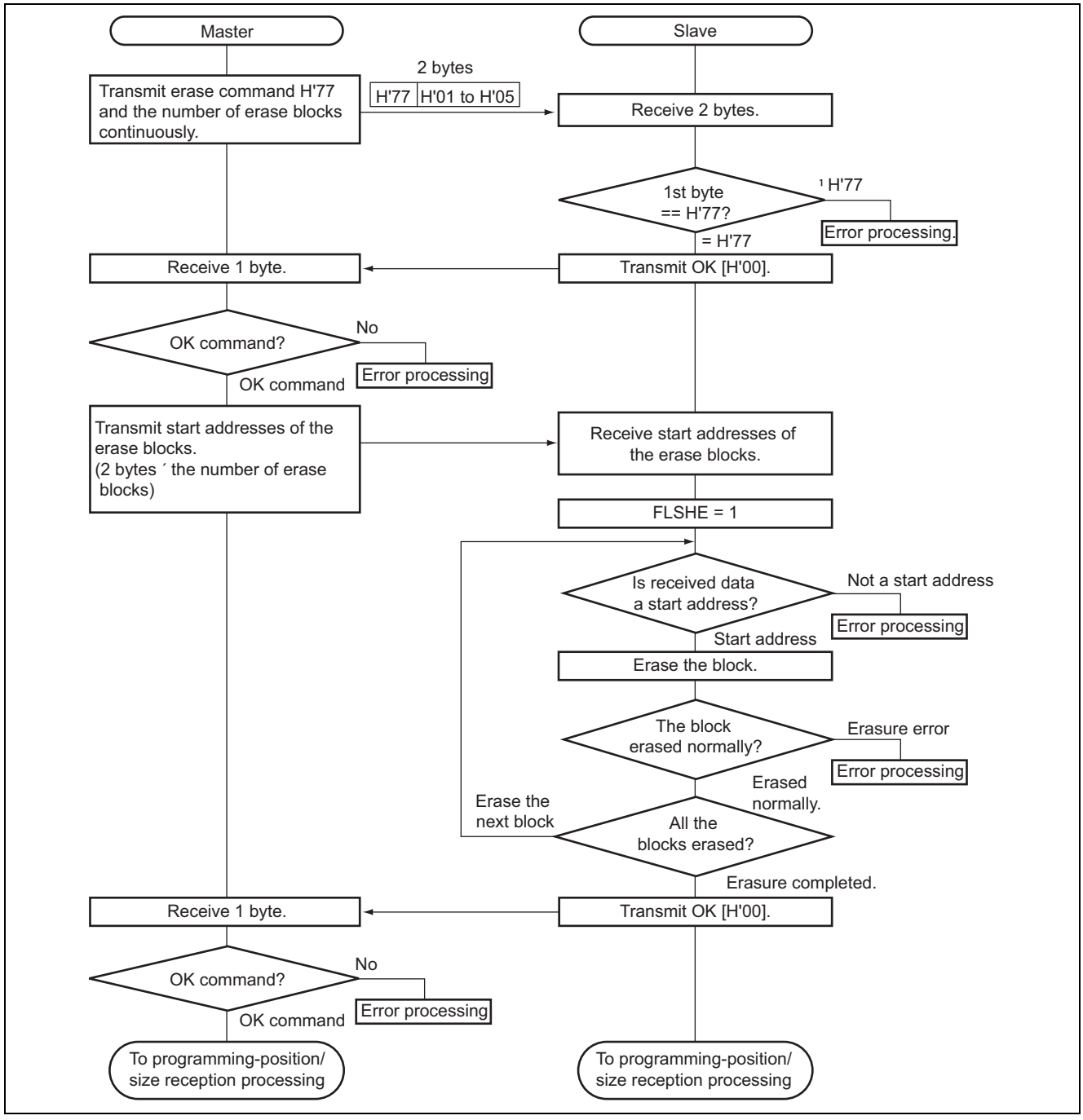



**Figure 13 Erase Processing**

## 4.3 Programming Position/Size Reception Processing

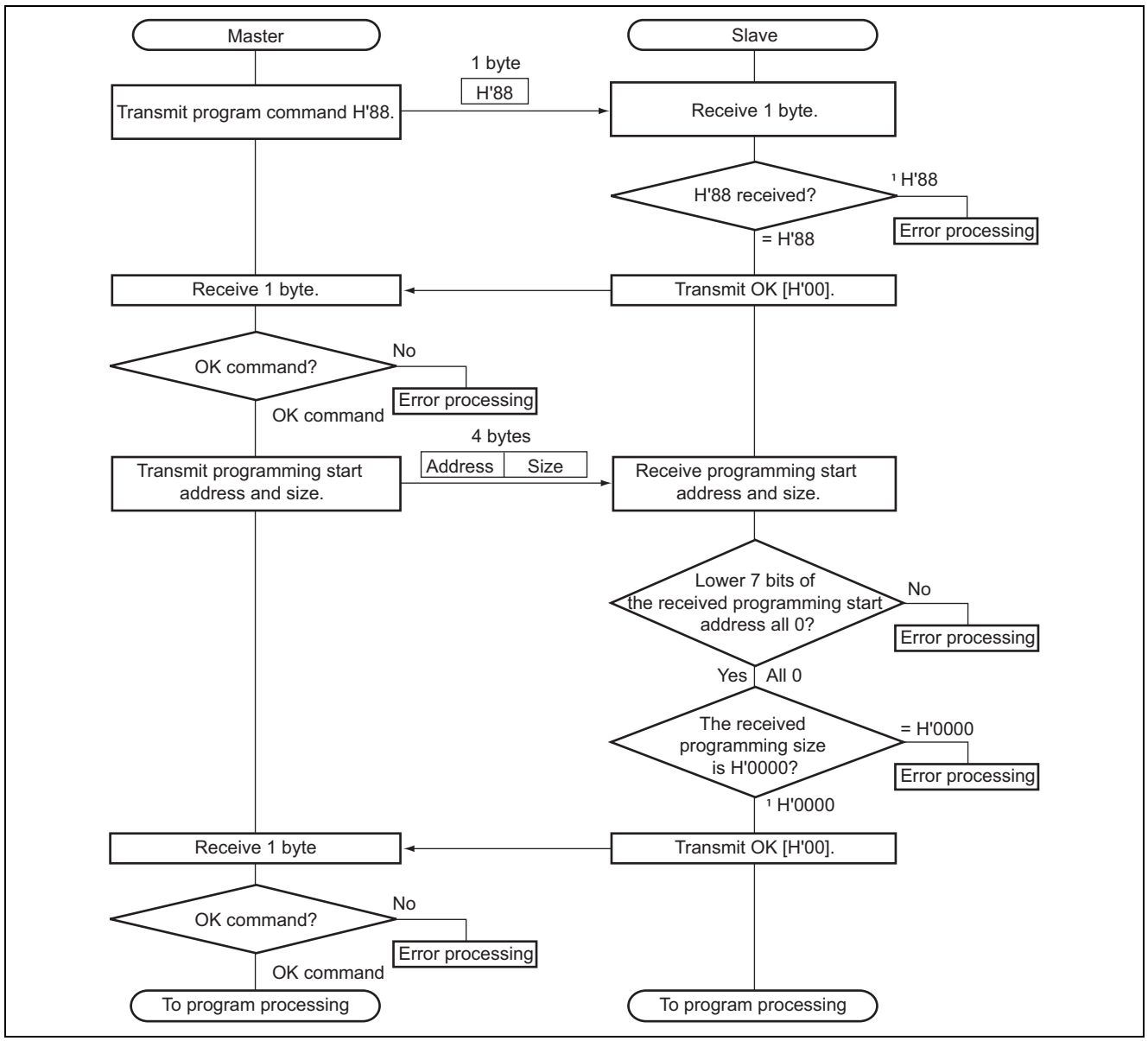



**Figure 14 Programming Position/Size Reception Processing**

## 4.4 Program Processing

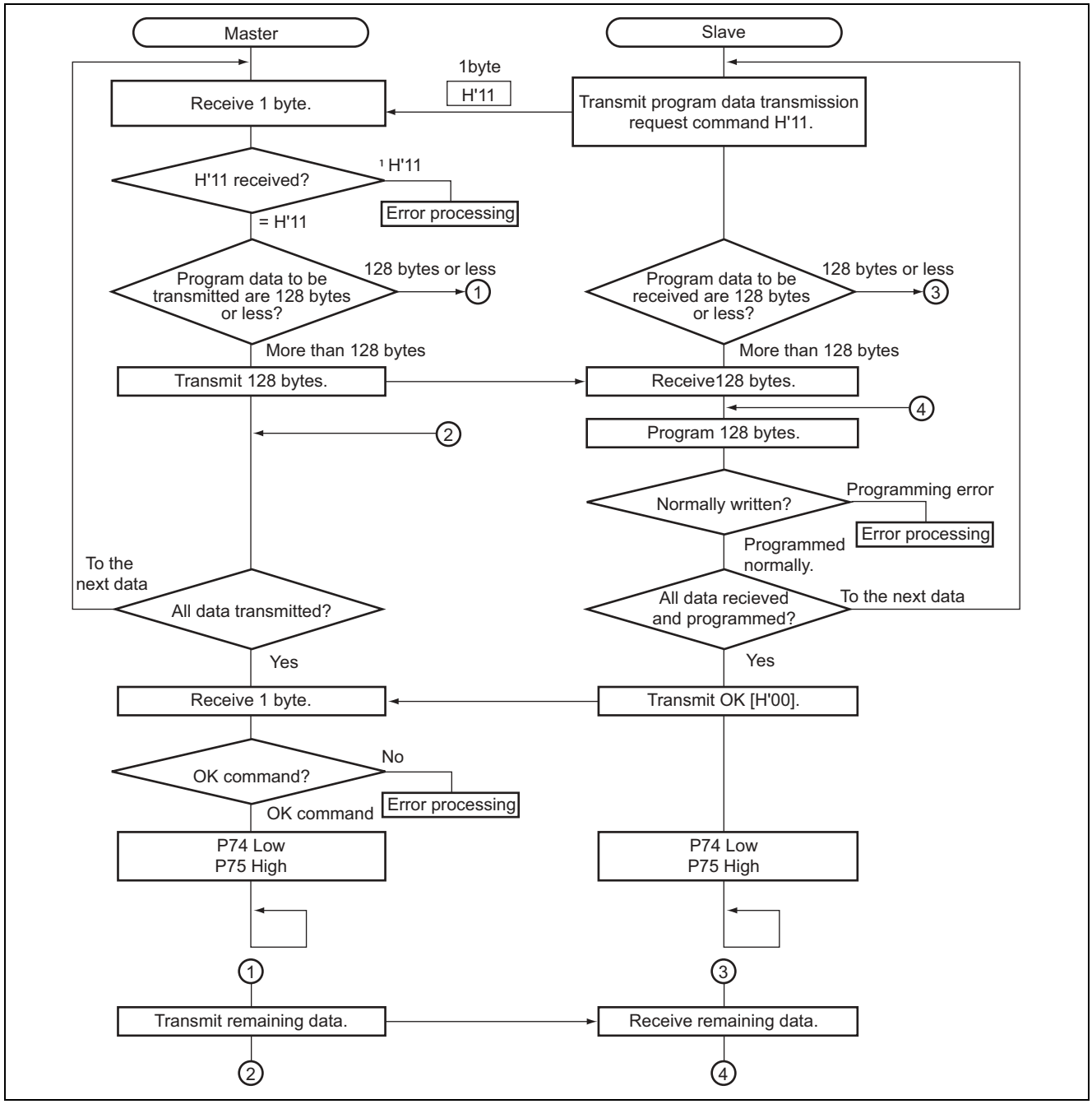



**Figure 15 Program Processing**

## 4.5 Error Processing

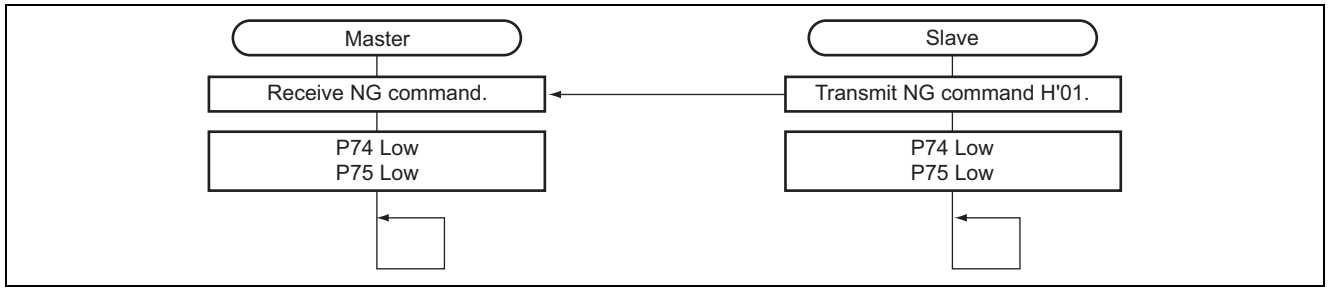
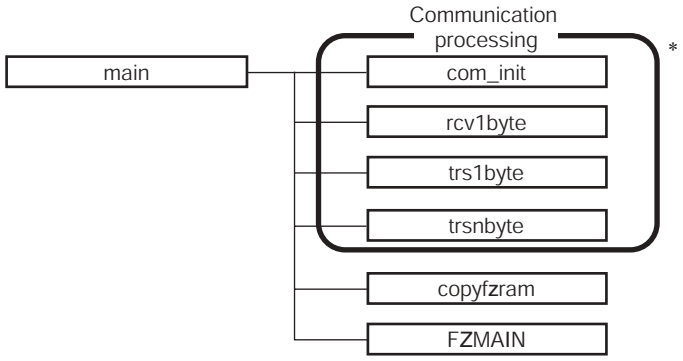



**Figure 16 Error Processing**

# 5. Slave-Side Normal Program

## 5.1 Hierarchical Structure

The slave-side normal program that runs in flash memory (slave-side normal program) performs processing to execute a user application program (normal application), receive the reprogramming start command, and transfer the program/erase control program in flash memory to on-chip RAM. The hierarchical structure of routines used in the slave-side normal program is shown in figure 17.




Note: * Called from the asynchronous serial communication program.

**Figure 17 Slave-Side Normal Program**

## 5.2 List of Functions

**Table 7 Functions of Slave-Side Normal Program**

| Function Name | Description |
|---|---|
| main | Executes the normal application, receives a reprogramming start command, and transfers the program/erase control program in flash memory to on-chip RAM. |
| copyfzram | Transfers the flash memory program/erase control program to RAM. |
| FZMAIN | Flash memory program/erase control program |

## 5.3 Modules

### 5.3.1 main() Function

(1) **Module Specifications**

- Functional summary
  - — Executes the user application program (normal application)
  - — Performs reprogramming-start command reception processing
  - — Branches to the program/erase control program

**Table 8 Module Specifications**

| | Type | Variable Name | Description |
|---|---|---|---|
| Argument | None | None | None |

(2) **Internal Registers**

The following describes internal registers used in this sample task

- SSR: Serial status register (Address: H'FFAC)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 6 | RDRF | Undefined | Receive Data Register Full<br>0: No receive data is stored in RDR.<br>1: Receive data is stored in RDR. |

- PDR7: Port data register 7 (Address: H'FFDA)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 5 | P75 | 0 | Port 75<br>0: The P75 pin is low.<br>1: The P75 pin is high. |
| 4 | P74 | Undefined | Port 74<br>0: The P74 pin is low.<br>1: The P74 pin is high. |

- PMR1: Port mode register 1 (Address: H'FFE0)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 5 | IRQ1 | 1 | P15/$\overline{\text{IRQ1}}$ Pin Function Switching<br>0: P15/$\overline{\text{IRQ1}}$ pin functions as a general I/O port.<br>1: P15/$\overline{\text{IRQ1}}$ pin functions as the $\overline{\text{IRQ1}}$ input pin. |

- PCR7: Port control register 7 (Address: H'FFEA)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 5 | PCR75 | 0 | 0: The P75 pin is set up as an input pin.<br>1: The P75 pin is set up as an output pin. |
| 4 | PCR74 | 0 | 0: The P74 pin is set up as an input pin.<br>1: The P74 pin is set up as an output pin. |

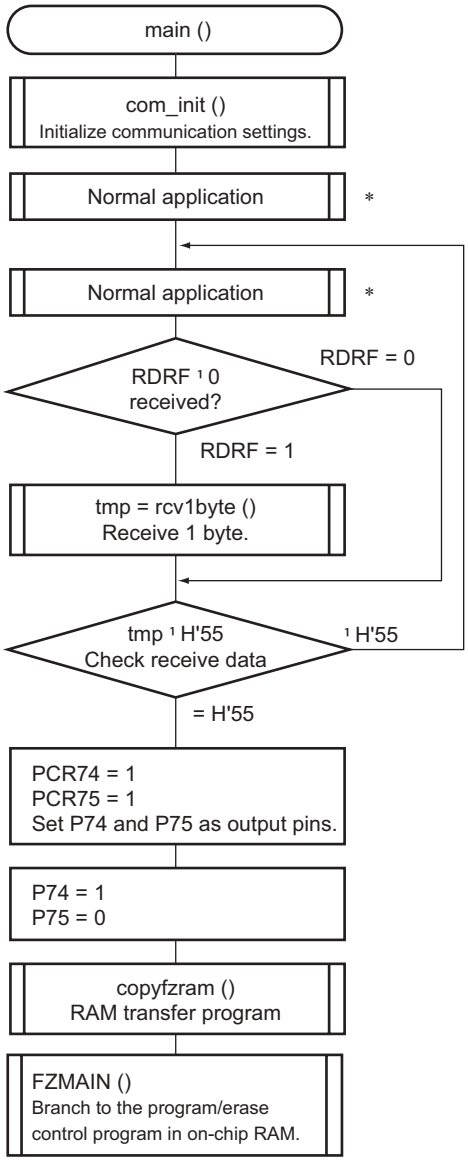
- IEGR1: Interrupt edge select register 1 (Address: H'FFF2)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 1 | IEG1 | 1 | IRQ1 Edge Select<br>0: The falling edge on the $\overline{\text{IRQ1}}$ pin input is detected.<br>1: The rising edge on the $\overline{\text{IRQ1}}$ pin input is detected. |

- IRR1: Interrupt flag register 1 (Address: H'FFF6)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 1 | IRRI1 | 0 | IRQ1 Interrupt Request Flag<br>0: An interrupt of the $\overline{\text{IRQ1}}$ pin has not been requested.<br>1: An interrupt of the $\overline{\text{IRQ1}}$ pin has been requested. |

(3) **Flowchart**




Notes: * The normal application is a sample program which is unrelated to flash-memory reprogram processing. In this example, it appears two times.

### 5.3.2 copyfzram() Function

(1) **Module Specifications**

- Functional summary
  — Transfers the flash memory program/erase control program to RAM.

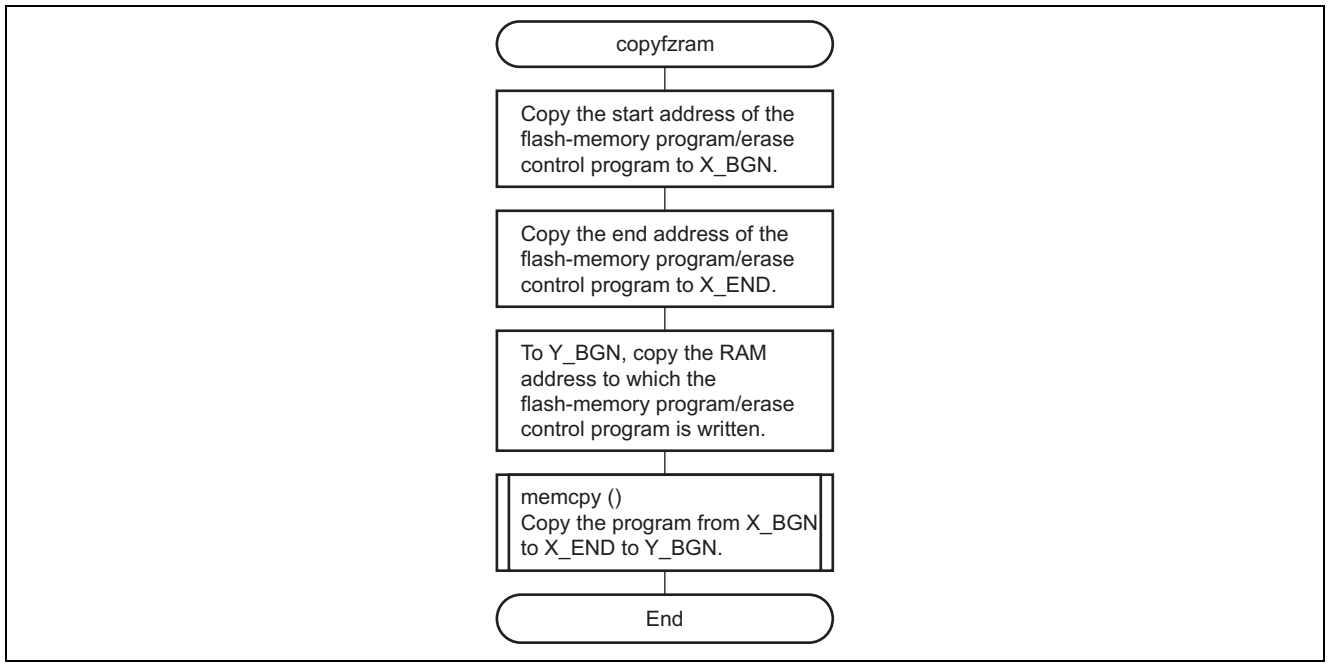
**Table 9 Module Specifications**

| | Type | Variable Name | Description |
|---|---|---|---|
| Argument | None | None | None |

(2) **Internal Registers**

None

(3) **Flowchart**




### 5.3.3 FZMAIN() Function

(1) **Module Specifications**

- Functional summary
  — Calls the main routine of the program/erase control program.

# 6. Slave-Side Program/Erase Control Program

## 6.1 Hierarchical Structure

The program/erase control program performs erasure in erase block units, receives data to be programmed into flash memory, and programs them into flash memory. Figure 18 shows the hierarchical structure of the routines used in the program/erase control program. Subroutines other than the FZMAIN() function are divided into communication processing and flash memory program/erase processing.

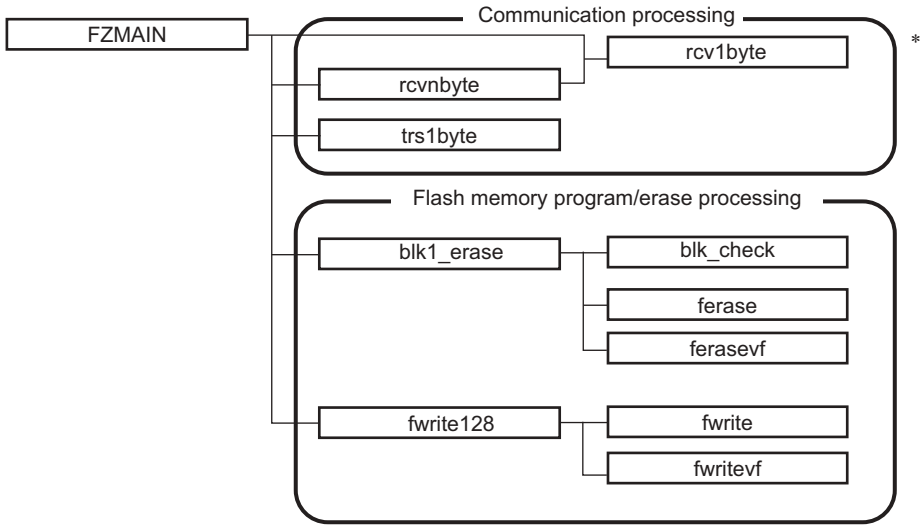



**Figure 18 Program/Erase Control Program**

## 6.2 List of Functions

**Table 10 Functions of Program/Erase Control Program**

| Function Name | Description |
|---|---|
| FZMAIN | Main routine of the program/erase control program |
| blk1_erase | Erases the specified block in the flash memory. |
| blk_check | Determines the block number of the erase block from the erase start address. |
| ferase | Erases the specified block. |
| ferasevf | Erases the specified block and performs verification. |
| fwrite128 | Programs and verifies 128 bytes. |
| fwrite | Programs data to the specified address. |
| fwritevf | Verifies data for the specified address and creates reprogramming data. |

## 6.3 List of Constants

**Table 11 List of Constants**

| Constant Name | Value | Description |
|---|---|---|
| OK | H'00 | Return value when normal |
| NG | H'01 | Return value when an error has occurred |
| WNG | H'02 | Programming error |
| MAXBLK | H'05 | Total number of blocks in flash memory |
| OW_COUNT | H'06 | Number of reprogramming operations |
| WDT_ERASE | H'A0 | Watchdog timer setting value for erasure |
| WDT_WRITE | H'F0 | Watchdog timer setting value for programming |
| WLOOP1 | 1*MHZ/KEISU+1 = 2 (H'02) | Number of WAIT statement executions, 1-µs wait |
| WLOOP2 | 2*MHZ/KEISU+1 = 4 (H'04) | Number of WAIT statement executions, 2-µs wait |
| WLOOP4 | 4*MHZ/KEISU+1 = 7 (H'07) | Number of WAIT statement executions, 4-µs wait |
| WLOOP5 | 5*MHZ/KEISU+1 = 9 (H'09) | Number of WAIT statement executions, 5-µs wait |
| WLOOP10 | 10*MHZ/KEISU+1 = 17 (H'11) | Number of WAIT statement executions, 10-µs wait |
| WLOOP20 | 20*MHZ/KEISU+1 = 33 (H'21) | Number of WAIT statement executions, 20-µs wait |
| WLOOP50 | 50*MHZ/KEISU+1 = 81 (H'51) | Number of WAIT statement executions, 50-µs wait |
| WLOOP100 | 100*MHZ/KEISU+1 = 161 (H'A1) | Number of WAIT statement executions, 100-µs wait |
| TIME10 | 10*MHZ/KEISU+1 = 17 (H'11) | Number of WAIT statement executions, 10-µs wait |
| TIME30 | 30*MHZ/KEISU+1 = 49 (H'31) | Number of WAIT statement executions, 30-µs wait |
| TIME200 | 200*MHZ/KEISU+1 = 321 (H'141) | Number of WAIT statement executions, 200-µs wait |
| TIME10000 | 10000*MHZ/KEISU+1 = 16001 (H'3E81) | Number of WAIT statement executions, 10-ms wait |

Note: * MHZ: 16 (Operating frequency is 16 MHz.)
KEISU: 10 (The number of steps in one loop when the loop is repeated by a 'for' statement for an unsigned short variable.

## 6.4 RAM Usage

During execution of the FZMAIN function, the stack memory of table 12 is used. Stack memory is used for program execution, but the amount used varies depending on the compiler version and option settings.

**Table 12 RAM Used Mainly**

| Data Name | Amount Used in Stack Memory |
|---|---|
| Programming data | 128 bytes |
| Reprogramming data | 128 bytes |
| Additional programming data | 128 bytes |

## 6.5 Modules

### 6.5.1 FZMAIN() Function

(1) **Module Specifications**

- Functional summary
  - Erases flash memory.
  - Receives data to be programmed into flash memory.
  - Programs data into flash memory.
  - Executes a reset-start after programming is complete.

**Table 12 Module Specifications**

| | Type | Variable Name | Description |
|---|---|---|---|
| Argument | None | None | None |

(2) **Internal Registers**

- FENR: Flash memory enable register 7 (Address: H'FF9B)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 7 | FLSHE | 1 | Flash Memory Control Register Enable<br>0: Disables access to the flash memory control register.<br>1: Enables access to the flash memory control register. |

- TCSRWD: Timer control/status register WD (Address: H'FFC0)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 7 | B6WI | 0 | Bit 6 Write Disable<br>0: Enables writing to bit 6 in TCSRWD.<br>1: Disables writing to bit 6 in TCSRWD. |
| 6 | TCWE | 1 | Timer Counter WD Write Enable<br>0: Disables writing 8-bit data to TCWD.<br>1: Enables writing 8-bit data to TCWD. |
| 5 | B4WI | 0 | Bit 4 Write Disable<br>0: Enables writing to bit 4 in TCSRWD.<br>1: Disables writing to bit 4 in TCSRWD. |
| 4 | TCSRWE | 1 | Timer Control/Status Register WD Write Enable<br>0: Disables writing to bit 2 and bit 0 in TCSRWD.<br>1: Enables writing to bit 2 and bit 0 in TCSRWD. |
| 3 | B2WI | 0 | Bit 2 Write Disable<br>0: Enables writing to bit 2 in TCSRWD.<br>1: Disables writing to bit 2 in TCSRWD. |
| 2 | WDON | 0 | Watchdog Timer On<br>0: Disables watchdog timer operation.<br>1: Enables watchdog timer operation. |
| 1 | B0WI | 0 | Bit 0 Write Disable<br>0: Enables writing to bit 0 in TCSRWD.<br>1: Disables writing to bit 0 in TCSRWD. |

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 0 | WRST | 0 | Watchdog Timer Reset<br>0: Indicates that TCWD has not overflowed and an internal reset signal has not been generated.<br>1: Indicates that TCWD has overflowed and an internal reset signal has been generated. |

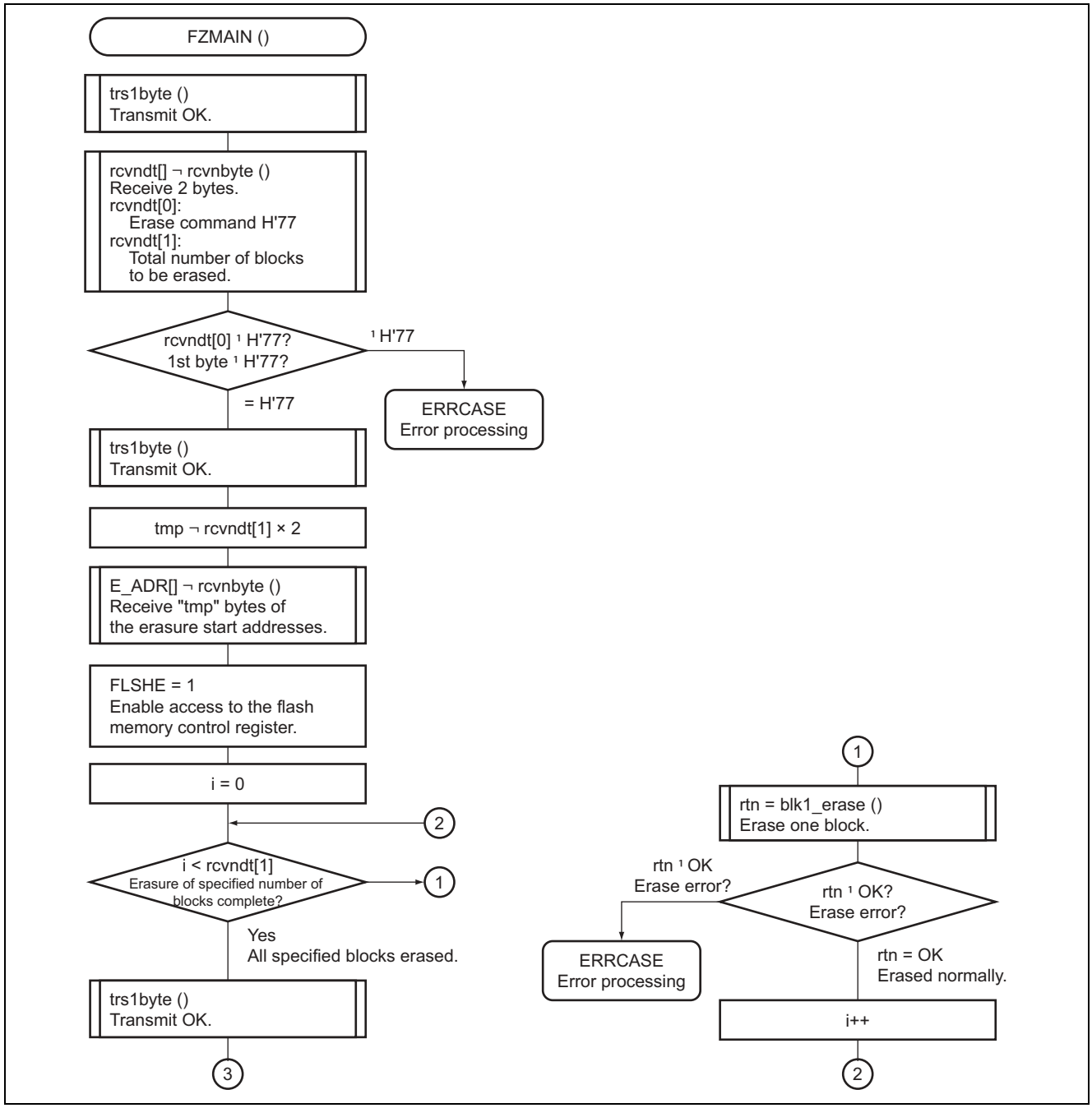
- TCWD: Timer counter WD (Address: H'FFC1)
  - Function: 8-bit counter which takes the φ/8192 clock as input
  - Setting value: H'FF
- TMWD: Timer mode register WD (Address H'FFC2)

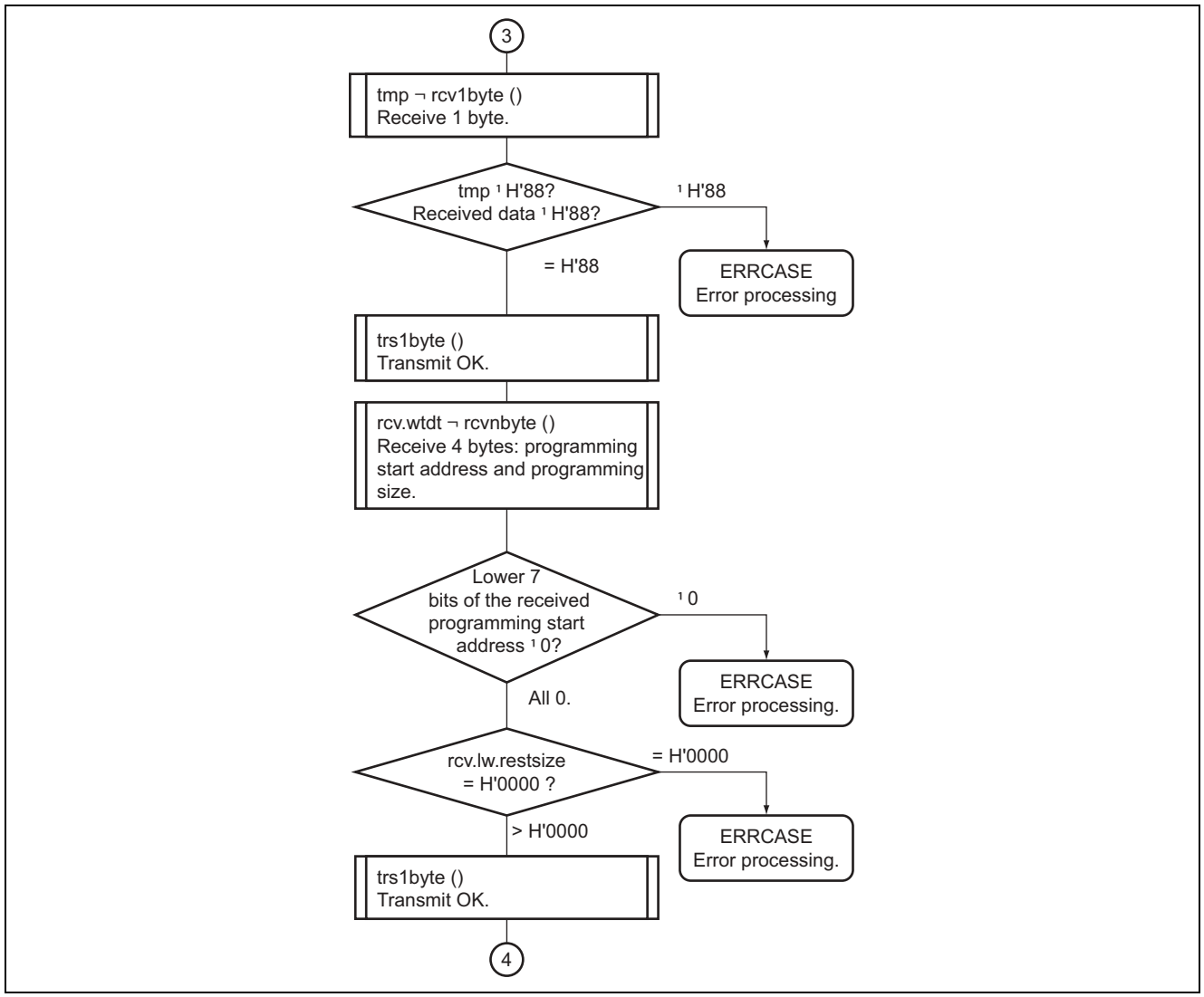
| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 3 | CKS3 | 1 | Clock Select 3 to 0<br>When CKS3 to CKS0 are set to B'1111, φ/8192 is selected as the input clock for TCWD. |
| 2 | CKS2 | 1 | |
| 1 | CKS1 | 1 | |
| 0 | CKS0 | 1 | |

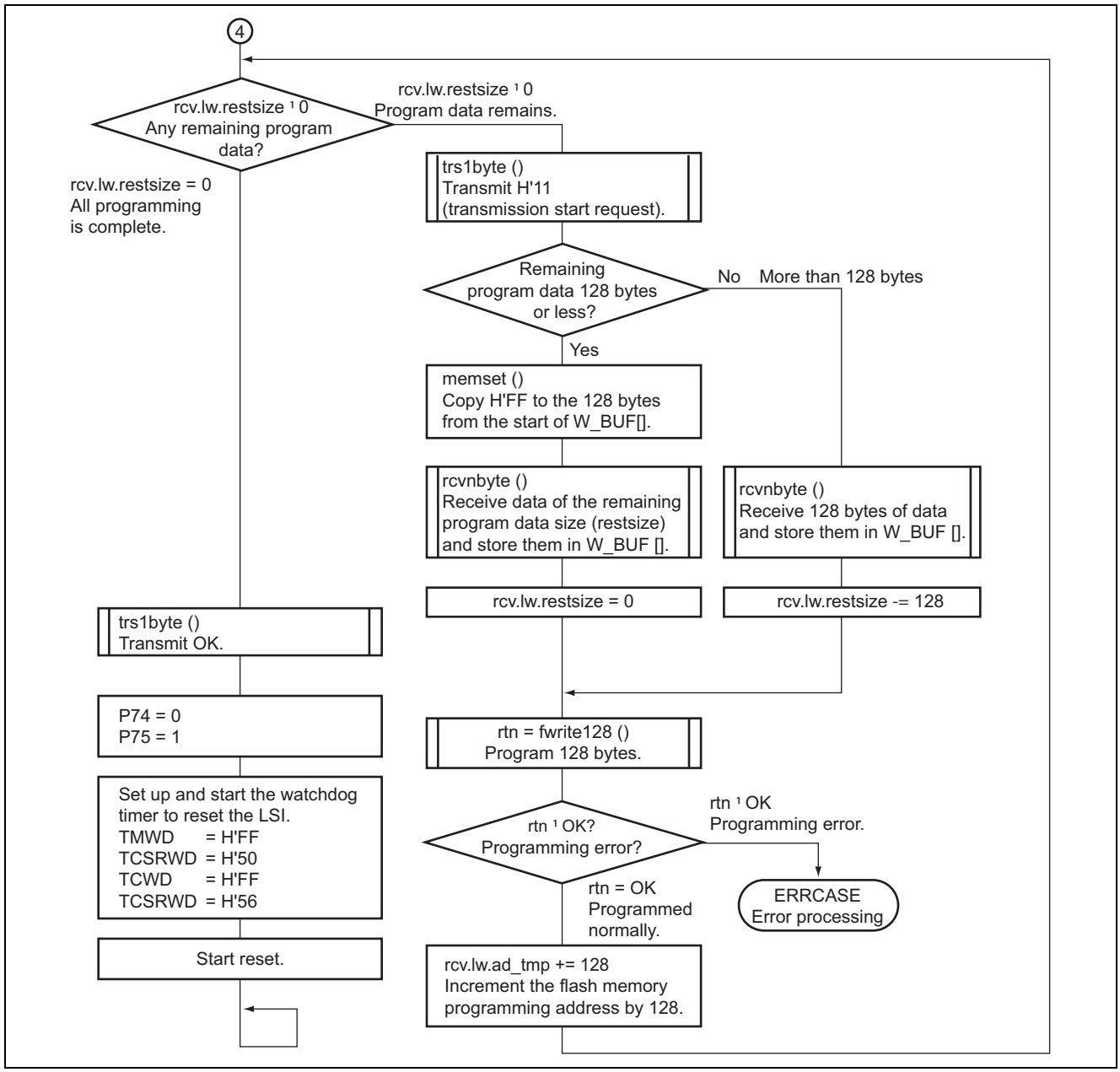
- PDR7: Port data register 7 (Address: H'FFDA)

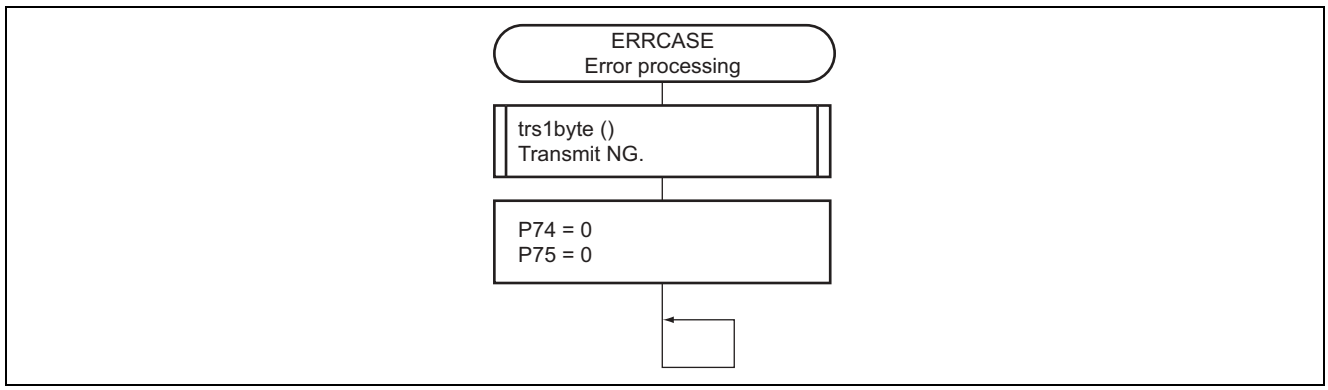
| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 5 | P75 | Undefined | Port 75<br>0: The P75 pin is low.<br>1: The P75 pin is high. |
| 4 | P74 | 0 | Port 74<br>0: The P74 pin is low.<br>1: The P74 pin is high. |

(3) **Flowchart**


FZMAIN ()
trs1byte ()
Transmit OK.
rcvndt[] ¬ rcvnbyte ()
Receive 2 bytes.
rcvndt[0]:
Erase command H'77
rcvndt[1]:
Total number of blocks
to be erased.
rcvndt[0] ¹ H'77?
1st byte ¹ H'77?
¹ H'77
ERRCASE
Error processing
= H'77
trs1byte ()
Transmit OK.
tmp ¬ rcvndt[1] × 2
E_ADR[] ¬ rcvnbyte ()
Receive "tmp" bytes of
the erasure start addresses.
FLSHE = 1
Enable access to the flash
memory control register.
i = 0
2
i < rcvndt[1]
Erasure of specified number of
blocks complete?
1
Yes
All specified blocks erased.
trs1byte ()
Transmit OK.
3
1
rtn = blk1_erase ()
Erase one block.
rtn ¹ OK
Erase error?
rtn ¹ OK?
Erase error?
ERRCASE
Error processing
rtn = OK
Erased normally.
i++
2


3
tmp ¬ rcv1byte ()
Receive 1 byte.
tmp ¹ H'88?
Received data ¹ H'88?
¹ H'88
ERRCASE
Error processing
= H'88
trs1byte ()
Transmit OK.
rcv.wtdt ¬ rcvnbyte ()
Receive 4 bytes: programming start address and programming size.
Lower 7 bits of the received programming start address ¹ 0?
¹ 0
ERRCASE
Error processing.
All 0.
rcv.lw.restsize = H'0000 ?
= H'0000
ERRCASE
Error processing.
> H'0000
trs1byte ()
Transmit OK.
4


4
rcv.lw.restsize ¹ 0
Any remaining program data?
rcv.lw.restsize ¹ 0
Program data remains.
rcv.lw.restsize = 0
All programming is complete.
trs1byte ()
Transmit H'11
(transmission start request).
Remaining program data 128 bytes or less?
No
More than 128 bytes
Yes
memset ()
Copy H'FF to the 128 bytes from the start of W_BUF[].
rcvnbyte ()
Receive data of the remaining program data size (restsize) and store them in W_BUF [].
rcvnbyte ()
Receive 128 bytes of data and store them in W_BUF [].
rcv.lw.restsize = 0
rcv.lw.restsize -= 128
trs1byte ()
Transmit OK.
P74 = 0
P75 = 1
Set up and start the watchdog timer to reset the LSI.
TMWD = H'FF
TCSRWD = H'50
TCWD = H'FF
TCSRWD = H'56
Start reset.
rtn = fwrite128 ()
Program 128 bytes.
rtn ¹ OK?
Programming error?
rtn ¹ OK
Programming error.
rtn = OK
Programmed normally.
ERRCASE
Error processing
rcv.lw.ad_tmp += 128
Increment the flash memory programming address by 128.


ERRCASE
Error processing
trs1byte ()
Transmit NG.
P74 = 0
P75 = 0

### 6.5.2 blk1_erase() Function

(1) **Module Specifications**

- Functional summary
  - — Determines the target erase block number from the erase start address.
  - — Erases the specified block of flash memory.

**Table 13 Module Specifications**

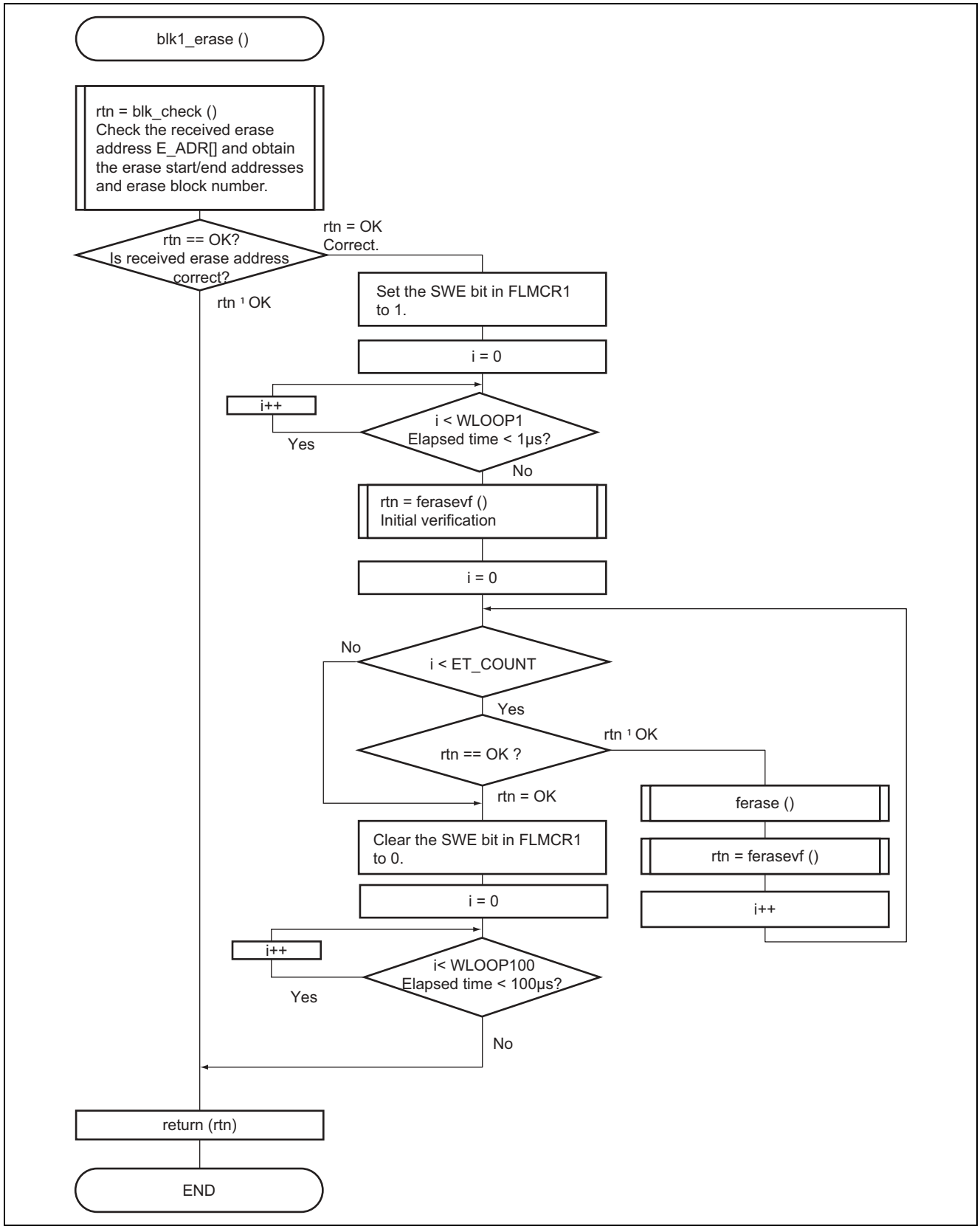
| | Type | Variable Name | Description |
|---|---|---|---|
| Argument | unsigned short | ers_ad | Erase start address |
| | unsigned char | ET_COUNT | Maximum number of erasures |
| Return value | char | — | Result flag (OK = H'00, NG = H'01) |

(2) **Internal Register**

- FLMCR1: Flash memory control register 1 (Address: H'FF90)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 6 | SWE | 1 | Software Write Enable<br>0: Disables flash memory programming/erasure.<br>1: Enables flash memory programming/erasure. |

(3) **Flowchart**


blk1_erase ()
rtn = blk_check ()
Check the received erase address E_ADR[] and obtain the erase start/end addresses and erase block number.
rtn == OK?
Is received erase address correct?
rtn = OK
Correct.
rtn ¹ OK
Set the SWE bit in FLMCR1 to 1.
i = 0
i++
i < WLOOP1
Elapsed time < 1µs?
Yes
No
rtn = ferasevf ()
Initial verification
i = 0
No
i < ET_COUNT
Yes
rtn == OK ?
rtn ¹ OK
rtn = OK
ferase ()
Clear the SWE bit in FLMCR1 to 0.
rtn = ferasevf ()
i = 0
i++
i< WLOOP100
Elapsed time < 100µs?
Yes
No
return (rtn)
END

### 6.5.3 blk_check() Function

(1) **Module Specifications**

- Functional summary
  - — Determines the target erase block number from the erase start address.
  - — Performs comparison with BLOCKADR[] to find out if the received erase start address value is correct, and returns the result flag, erase start address, erase end address, and block number.

**Table 14 Module Specifications**

| | Type | Variable Name | Contents |
|---|---|---|---|
| Argument | unsigned short | eck_ad | Erase start address |
| | unsigned short | *eck_st | Erase start address after verification |
| | unsigned short | *eck_ed | Erase end address after verification |
| | unsigned char | *blk_no | Target erase block number |
| Return value | char | — | Result flag (OK = H'00; NG = H'01) |

(2) **Internal Registers**

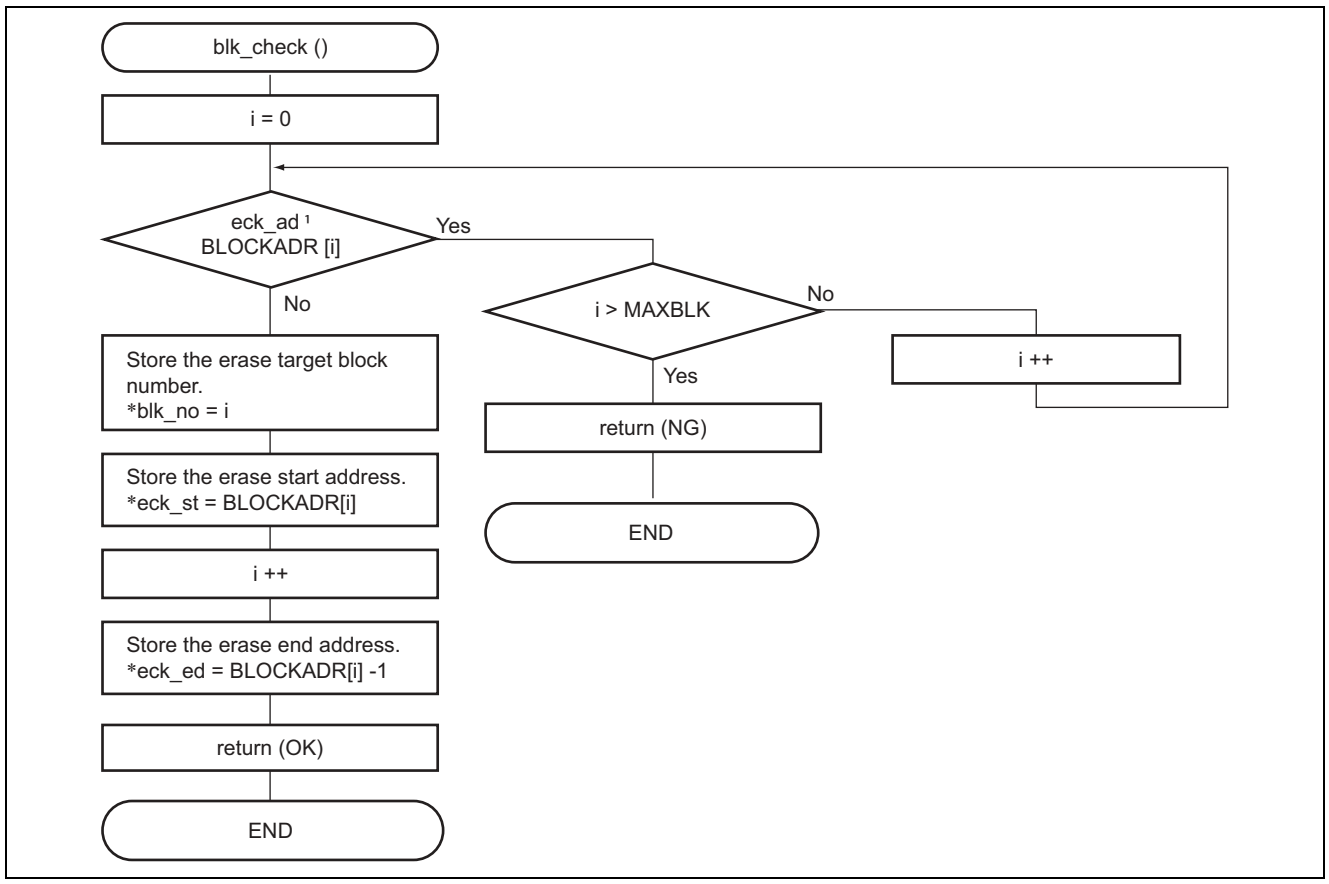
None

(3) **Global Variables**

- BLOCKADR[]: Stores the start addresses of blocks in flash memory.

```
unsigned short BLOCKADR[6] = {     /* Erase Block Address   */
   0x0000,                         /* EB0    1KBYTE         */
   0x0400,                         /* EB1    1KBYTE         */
   0x0800,                         /* EB2    1KBYTE         */
   0x0c00,                         /* EB3    1KBYTE         */
   0x1000,                         /* EB4    28KBYTE        */
   0x8000,                         /* End Block Address     */
};
```

(4) **Flowchart**


blk_check ()
i = 0
eck_ad [1]
BLOCKADR [i]
Yes
No
i > MAXBLK
No
Yes
i ++
Store the erase target block number.
*blk_no = i
return (NG)
END
Store the erase start address.
*eck_st = BLOCKADR[i]
i ++
Store the erase end address.
*eck_ed = BLOCKADR[i] -1
return (OK)
END

### 6.5.4 ferase() Function

(1) **Module Specifications**

- Functional summary
  — Erases the specified block of flash memory

**Table 15 Module Specifications**

| | Type | Variable Name | Description |
|---|---|---|---|
| Argument | unsigned char | e_blk_no | Target erase block number |

(2) **Internal Registers**

- FLMCR1: Flash memory control register 1 (Address: H'FF90)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 5 | ESU | 1 | Erase Setup<br>0: Cancels an erase setup state.<br>1: Enters an erase setup state. |
| 1 | E | 1 | Erase<br>0: Cancels erase mode.<br>1: Erase mode is entered if SWE = 1 and ESU = 1. |

- EBR1: Block specification register 1 (Address: H'FF93)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 4 | EB4 | 0 | When this bit is set to 1, the 28 kbytes from H'1000 to H'7FFF are specified to be erased. |
| 3 | EB3 | 0 | When this bit is set to 1, the 1 kbytes from H'0C00 to H'0FFF are specified to be erased. |
| 2 | EB2 | 0 | When this bit is set to 1, the 1 kbytes from H'0800 to H'0BFF are specified to be erased. |
| 1 | EB1 | 0 | When this bit is set to 1, the 1 kbytes from H'0400 to H'07FF are specified to be erased. |
| 0 | EB0 | 0 | When this bit is set to 1, the 1 kbytes from H'0000 to H'03FF are specified to be erased. |

Note: This register specifies the erase block of flash memory. When the SWE bit in FLMCR1 is 0, EBR1 is initialized to H'00. Multiple bits of this register should not be set to 1 at a time. When set, EBR1 is cleared to 0 automatically.

- TCSRWD: Timer control/status register WD (Address: H'FFC0)

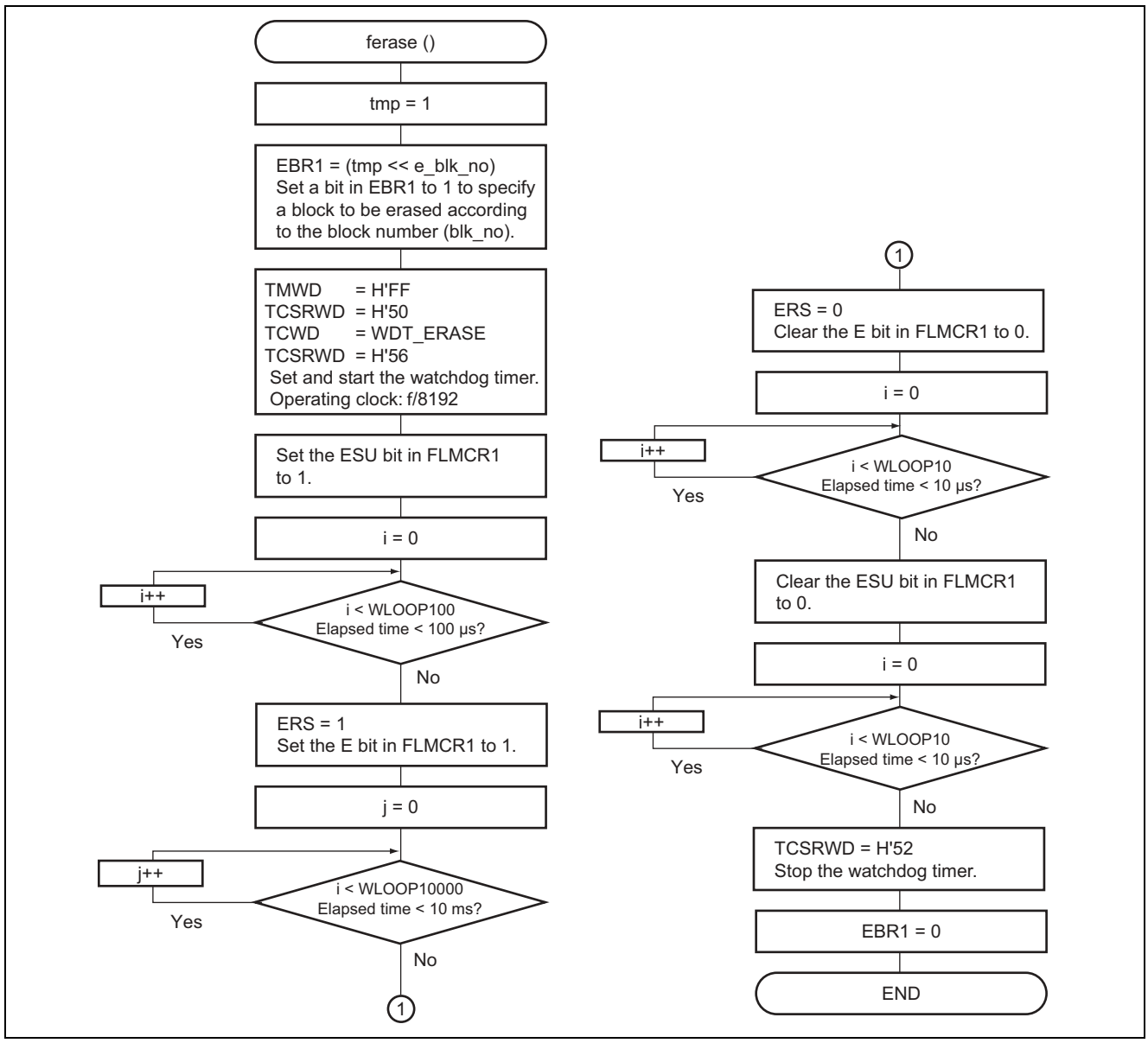
| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 7 | B6WI | 0 | Bit 6 Write Disable<br>0: Enables writing to bit 6 in TCSRWD.<br>1: Disables writing to bit 6 in TCSRWD. |
| 6 | TCWE | 1 | Timer Counter WD Write Enable<br>0: Disables writing 8-bit data to TCWD.<br>1: Enables writing 8-bit data to TCWD. |
| 5 | B4WI | 0 | Bit 4 Write Disable<br>0: Enables writing to bit 4 in TCSRWD.<br>1: Disables writing to bit 4 in TCSRWD. |
| 4 | TCSRWE | 1 | Timer Control/Status Register WD Write Enable<br>0: Disables writing to bit 2 and bit 0 in TCSRWD.<br>1: Enables writing to bit 2 and bit 0 in TCSRWD. |
| 3 | B2WI | 0 | Bit 2 Write Disable<br>0: Enables writing to bit 2 in TCSRWD.<br>1: Disables writing to bit 2 in TCSRWD. |
| 2 | WDON | 0 | Watchdog Timer On<br>0: Disables watchdog timer operation.<br>1: Enables watchdog timer operation. |
| 1 | B0WI | 0 | Bit 0 Write Disable<br>0: Enables writing to bit 0 in TCSRWD.<br>1: Disables writing to bit 0 in TCSRWD. |
| 0 | WRST | 0 | Watchdog Timer Reset<br>0: Indicates that TCWD has not overflowed and an internal reset signal has not been generated.<br>1: Indicates that TCWD has overflowed and an internal reset signal has been generated. |

- TCWD: Timer counter WD (Address: H'FFC1)
  - Function: 8-bit counter which takes the $\phi$/8192 clock as input
  - Setting value: WDT_ERASE (H'A0)

- TMWD: Timer mode register WD (Address H'FFC2)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 3 | CKS3 | 1 | Clock Select 3 to 0 |
| 2 | CKS2 | 1 | When CKS3 to CKS0 are set to B'1111, $\phi$/8192 is selected as the input clock for TCWD. |
| 1 | CKS1 | 1 | |
| 0 | CKS0 | 1 | |

(3) **Flowchart**


ferase ()
tmp = 1
EBR1 = (tmp << e_blk_no)
Set a bit in EBR1 to 1 to specify a block to be erased according to the block number (blk_no).
TMWD = H'FF
TCSRWD = H'50
TCWD = WDT_ERASE
TCSRWD = H'56
Set and start the watchdog timer.
Operating clock: f/8192
Set the ESU bit in FLMCR1 to 1.
i = 0
i++
i < WLOOP100
Elapsed time < 100 µs?
Yes
No
ERS = 1
Set the E bit in FLMCR1 to 1.
j = 0
j++
i < WLOOP10000
Elapsed time < 10 ms?
Yes
No
1
1
ERS = 0
Clear the E bit in FLMCR1 to 0.
i = 0
i++
i < WLOOP10
Elapsed time < 10 µs?
Yes
No
Clear the ESU bit in FLMCR1 to 0.
i = 0
i++
i < WLOOP10
Elapsed time < 10 µs?
Yes
No
TCSRWD = H'52
Stop the watchdog timer.
EBR1 = 0
END

### 6.5.5 ferasevf() Function

(1) **Module Specifications**

- Functional summary:
  — Verifies that the specified block in the flash memory has been erased.

**Table 16 Module Specifications**

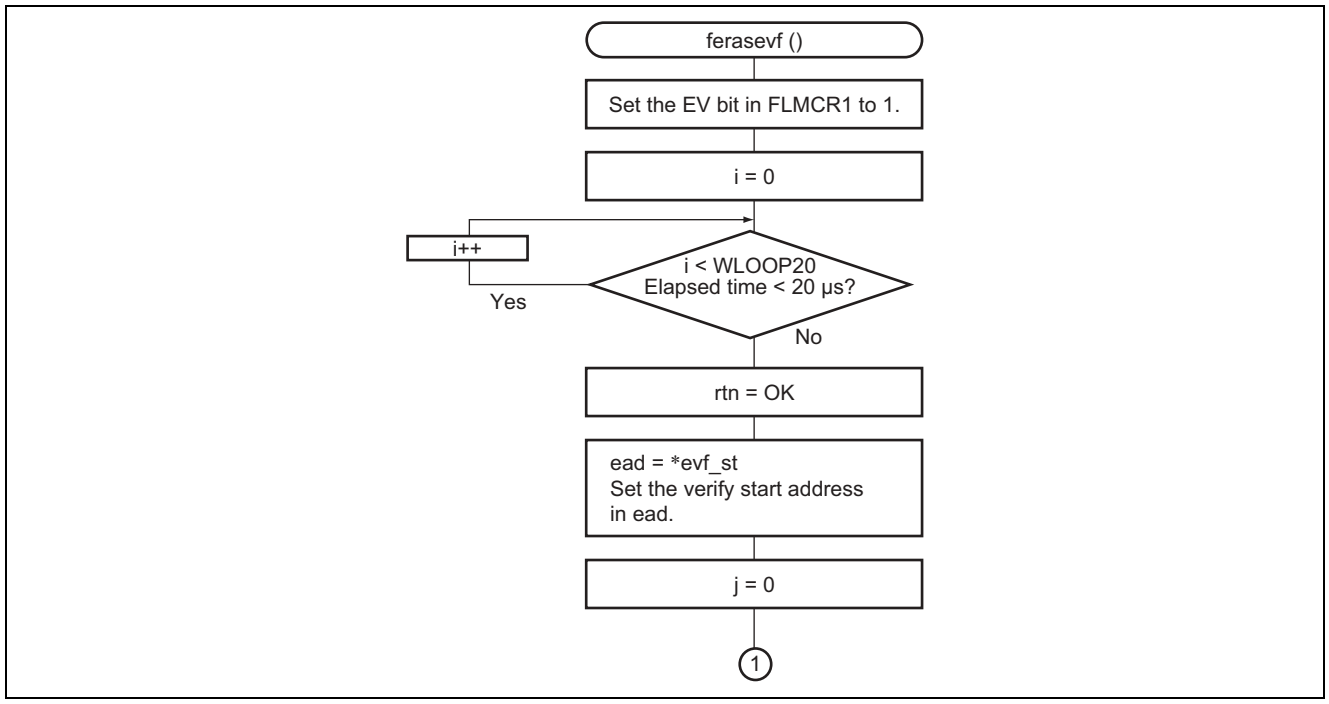
| | Type | Variable Name | Description |
|---|---|---|---|
| Argument | unsigned short | evf_st | Erase start address |
| | unsigned short | evf_ed | Erase end address |
| Return value | char | — | Result flag (OK = H'00; NG = H'01) |

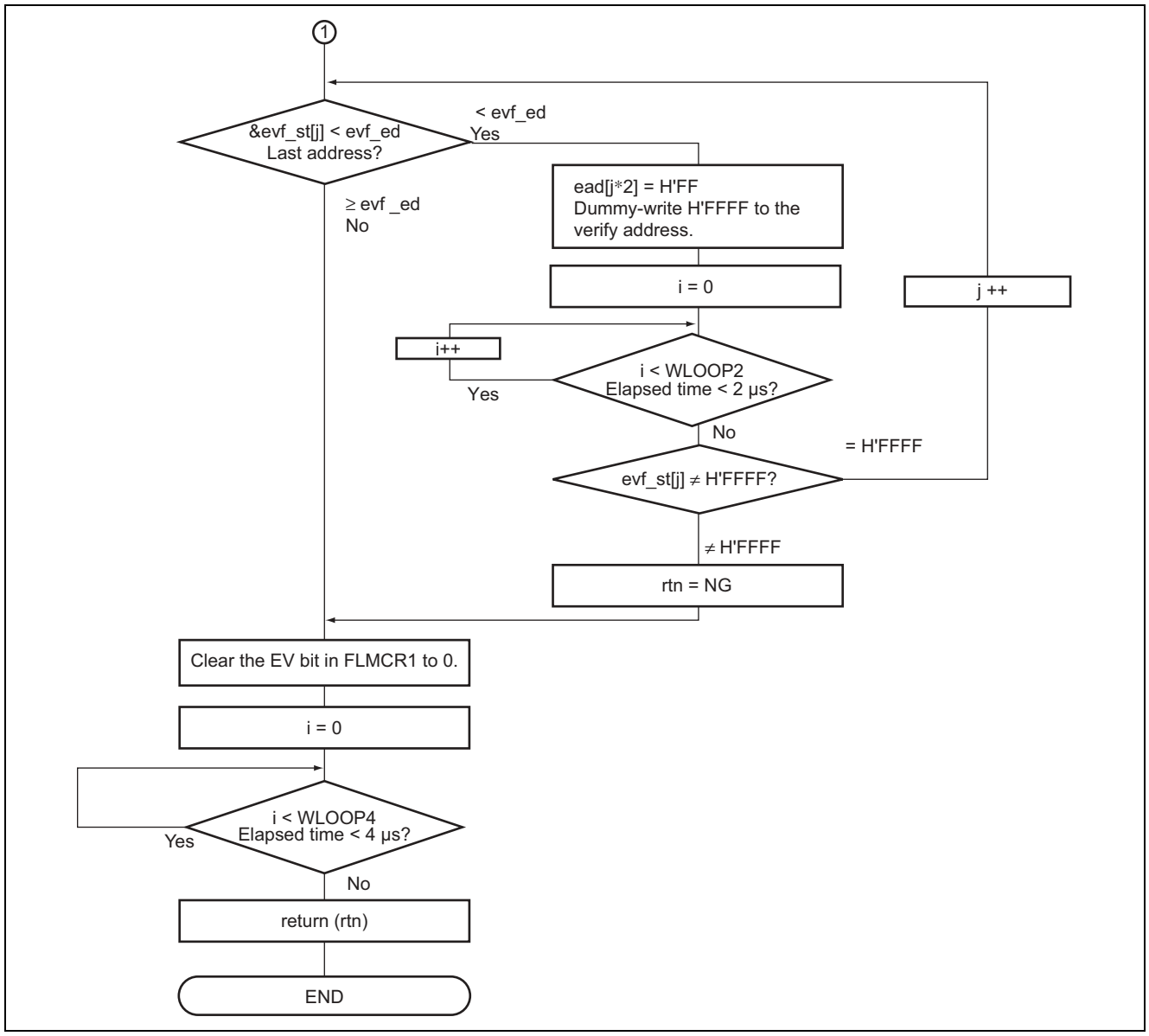
(2) **Internal Registers**

- FLMCR1: Flash memory control register 1 (Address: H'FF90)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 3 | EV | 1 | Erase Verify<br>0: Cancels erase-verify mode.<br>1: Enters erase-verify mode. |

(3) **Flowchart**




1
&evf_st[j] < evf_ed
Last address?
< evf_ed
Yes
≥ evf _ed
No
ead[j*2] = H'FF
Dummy-write H'FFFF to the
verify address.
i = 0
j ++
i++
i < WLOOP2
Elapsed time < 2 µs?
Yes
No
= H'FFFF
evf_st[j] ≠ H'FFFF?
≠ H'FFFF
rtn = NG
Clear the EV bit in FLMCR1 to 0.
i = 0
i < WLOOP4
Elapsed time < 4 µs?
Yes
No
return (rtn)
END

### 6.5.6 fwrite128() Function

(1) **Module Specifications**

- Functional summary
  — Programs 128 bytes and performs verification.

**Table 17 Module Specifications**

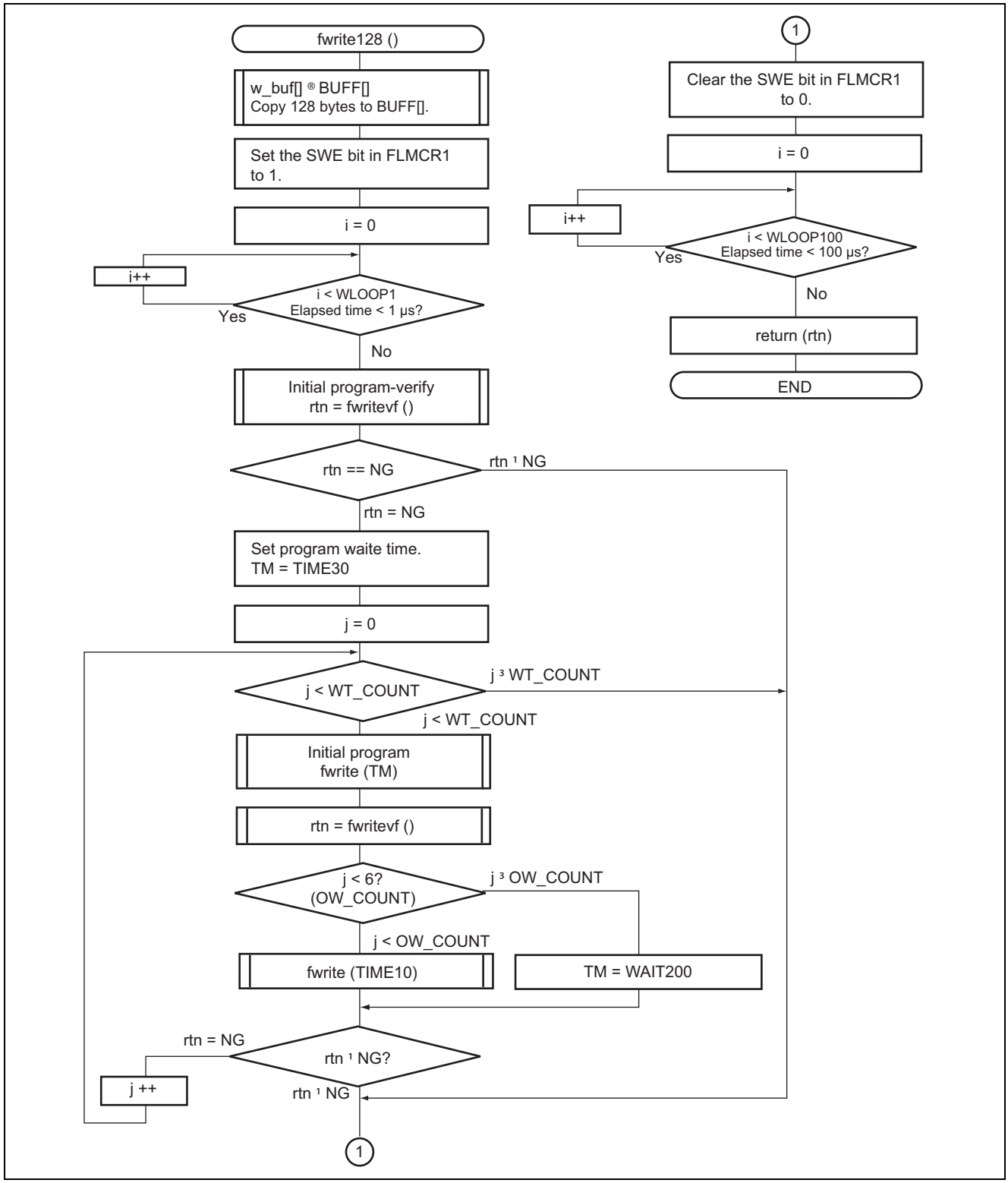
| | Type | Variable Name | Description |
|---|---|---|---|
| Argument | unsigned char | *wt_buf | 128 bytes of program data |
| | unsigned char | *wt_adr | Programming address |
| | unsigned short | WT_COUNT | Maximum number of programming operations |
| Return value | char | — | Result flag (OK = H'00; NG = H'01) |

(2) **Internal Register**

- FLMCR1: Flash memory control register 1 (Address: H'FF90)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 6 | SWE | 1 | Software Write Enable<br>0: Disables flash memory programming/erasure.<br>1: Enables flash memory programming/erasure. |

(3) **Flowchart**


fwrite128 ()
w_buf[] ® BUFF[]
Copy 128 bytes to BUFF[].
Set the SWE bit in FLMCR1 to 1.
i = 0
i++
i < WLOOP1
Elapsed time < 1 µs?
Yes
No
Initial program-verify
rtn = fwritevf ()
rtn == NG
rtn ¹ NG
rtn = NG
Set program waite time.
TM = TIME30
j = 0
j < WT_COUNT
j ³ WT_COUNT
j < WT_COUNT
Initial program
fwrite (TM)
rtn = fwritevf ()
j < 6?
(OW_COUNT)
j ³ OW_COUNT
j < OW_COUNT
fwrite (TIME10)
TM = WAIT200
rtn = NG
rtn ¹ NG?
j ++
rtn ¹ NG
1
1
Clear the SWE bit in FLMCR1 to 0.
i = 0
i++
i < WLOOP100
Elapsed time < 100 µs?
Yes
No
return (rtn)
END

### 6.5.7 fwrite() Function

(1) **Module Specifications**

- Functional summary
  — Writes addresses for programming.

**Table 18 Module Specifications**

| | Type | Variable Name | Description |
|---|---|---|---|
| Argument | unsigned char | *buf | Program start address (reprogramming data or additional programming data) |
| | unsigned char | *w_adr | Program address |
| | unsigned short | ptime | P-bit setting time (10 μs, 30 μs, or 2000 μs) |

(2) **Internal Registers**

- FLMCR1: Flash memory control register 1 (Address: H'FF90)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 4 | PSU | 1 | Program Setup<br>0: Cancels program setup state.<br>1: Enters program setup state. |
| 0 | P | 1 | Program<br>0: Cancels program mode.<br>1: Program mode is entered if SWE = 1 and PSU = 1. |

- TCSRWD: Timer control/status register WD (Address: H'FFC0)

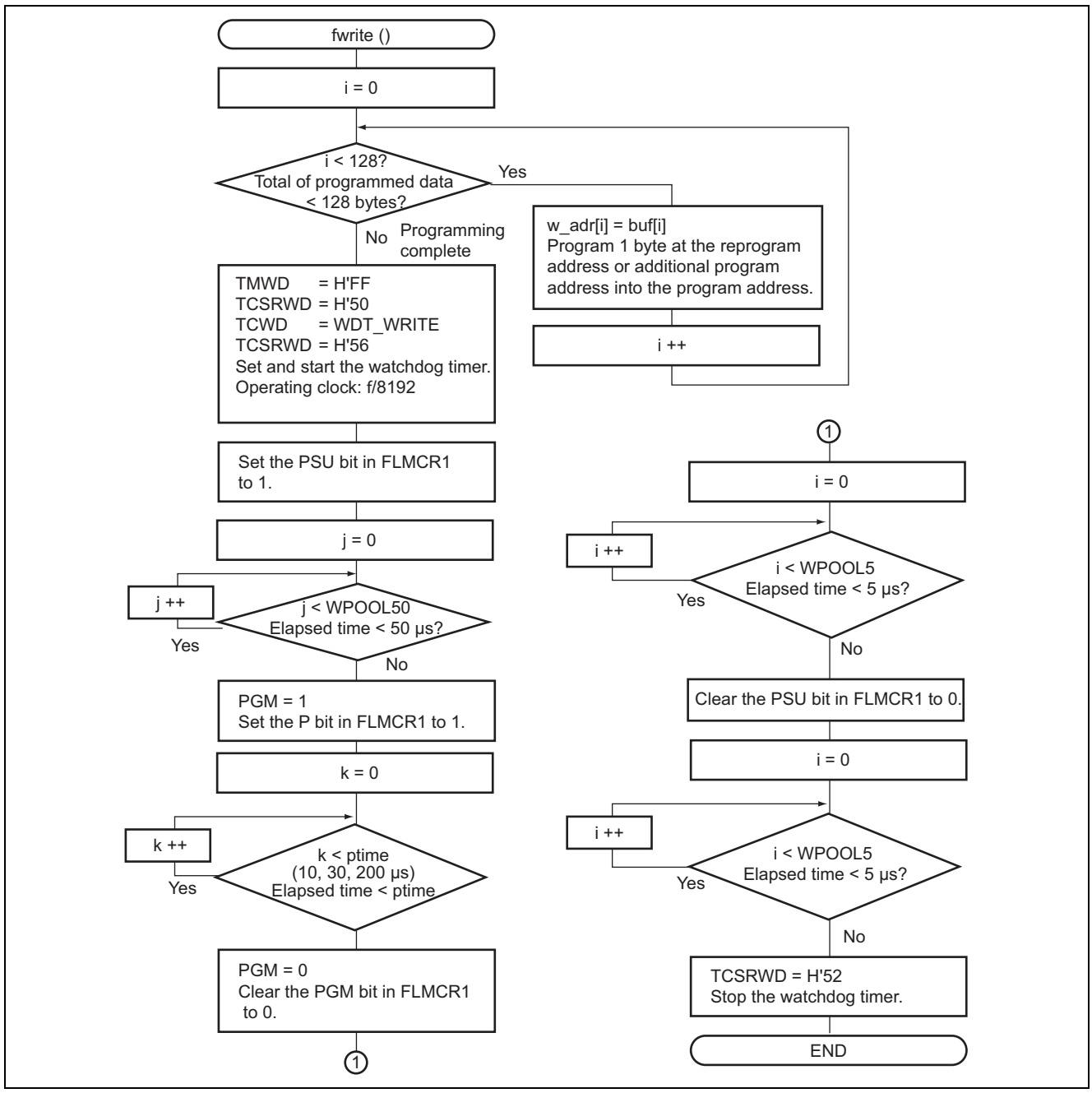
| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 7 | B6WI | 0 | Bit 6 Write Disable<br>0: Enables writing to bit 6 in TCSRWD.<br>1: Disables writing to bit 6 in TCSRWD. |
| 6 | TCWE | 1 | Timer Counter WD Write Enable<br>0: Disables writing 8-bit data to TCWD.<br>1: Enables writing 8-bit data to TCWD. |
| 5 | B4WI | 0 | Bit 4 Write Disable<br>0: Enables writing to bit 4 in TCSRWD.<br>1: Disables writing to bit 4 in TCSRWD. |
| 4 | TCSRWE | 1 | Timer Control/Status Register WD Write Enable<br>0: Disables writing to bit 2 and bit 0 in TCSRWD.<br>1: Enables writing to bit 2 and bit 0 in TCSRWD. |
| 3 | B2WI | 0 | Bit 2 Write Disable<br>0: Enables writing to bit 2 in TCSRWD.<br>1: Disables writing to bit 2 in TCSRWD. |
| 2 | WDON | 0 | Watchdog Timer On<br>0: Disables watchdog timer operation.<br>1: Enables watchdog timer operation. |

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 1 | B0WI | 0 | Bit 0 Write Disable<br>0: Enables writing to bit 0 in TCSRWD.<br>1: Disables writing to bit 0 in TCSRWD. |
| 0 | WRST | 0 | Watchdog Timer Reset<br>0: Indicates that TCWD has not overflowed and an internal reset signal has not been generated.<br>1: Indicates that TCWD has overflowed and an internal reset signal has been generated. |

- TCWD: Timer counter WD (Address: H'FFC1)
  - Function: 8-bit counter which takes the φ/8192 clock as input
  - Setting value: WDT_WRITE (H'F0)
- TMWD: Timer mode register WD (Address H'FFC2)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 3 | CKS3 | 1 | Clock Select 3 to 0 |
| 2 | CKS2 | 1 | When CKS3 to CKS0 are set to B'1111, φ/8192 is selected as the input clock for TCWD. |
| 1 | CKS1 | 1 | |
| 0 | CKS0 | 1 | |

(3) **Flowchart:**


fwrite ()
i = 0
i < 128?
Total of programmed data
< 128 bytes?
Yes
No
Programming
complete
w_adr[i] = buf[i]
Program 1 byte at the reprogram
address or additional program
address into the program address.
i ++
TMWD = H'FF
TCSRWD = H'50
TCWD = WDT_WRITE
TCSRWD = H'56
Set and start the watchdog timer.
Operating clock: f/8192
Set the PSU bit in FLMCR1
to 1.
j = 0
j ++
j < WPOOL50
Elapsed time < 50 µs?
Yes
No
PGM = 1
Set the P bit in FLMCR1 to 1.
k = 0
k ++
k < ptime
(10, 30, 200 µs)
Elapsed time < ptime
Yes
PGM = 0
Clear the PGM bit in FLMCR1
to 0.
1
1
i = 0
i ++
i < WPOOL5
Elapsed time < 5 µs?
Yes
No
Clear the PSU bit in FLMCR1 to 0.
i = 0
i ++
i < WPOOL5
Elapsed time < 5 µs?
Yes
No
TCSRWD = H'52
Stop the watchdog timer.
END

### 6.5.8 fwritevf() Function

(1) **Module Specifications**

- Functional summary
  — Verifies the specified address and creates reprogramming data.

**Table 19 Module Specifications**

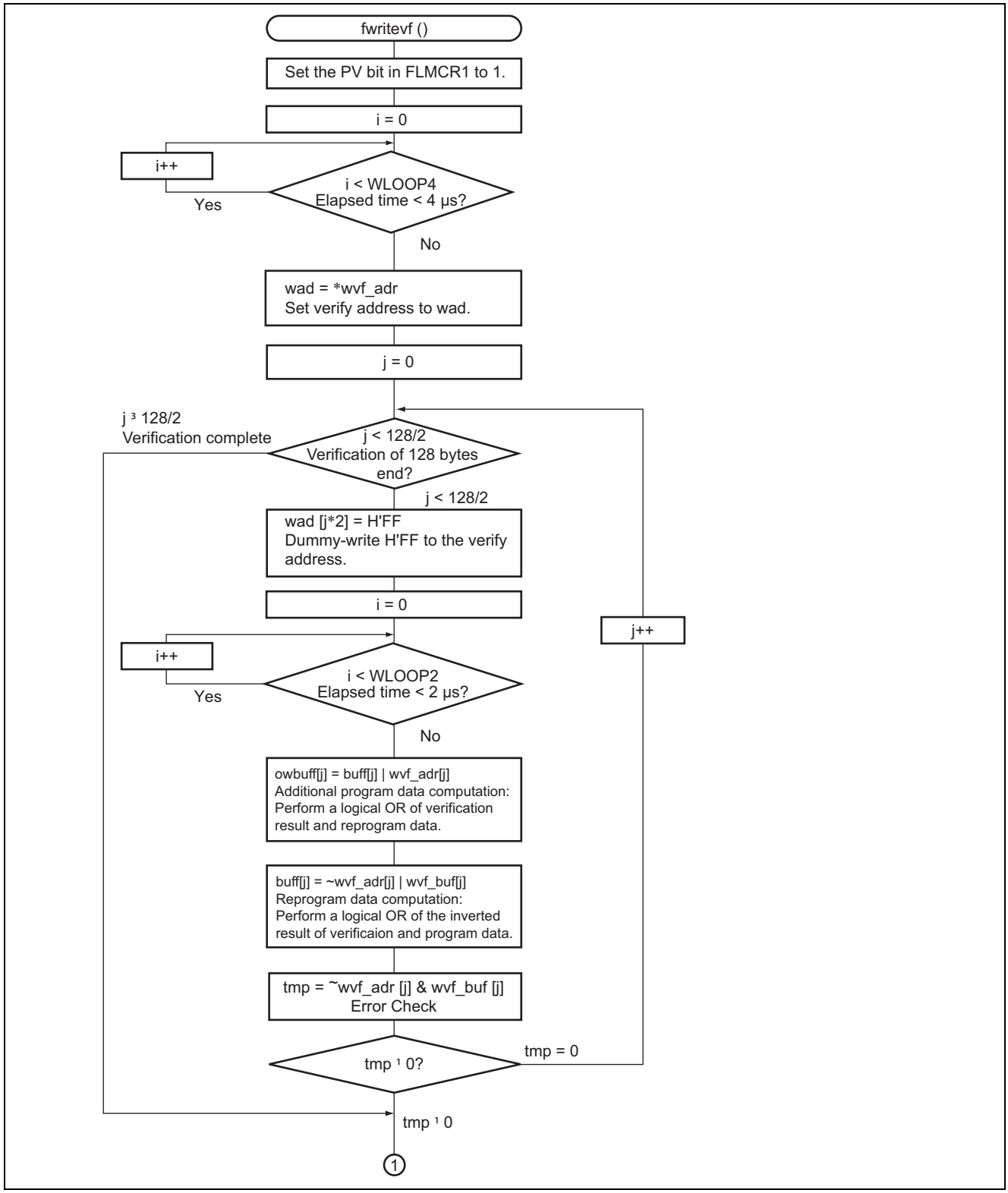
| | Type | Variable Name | Contents |
|---|---|---|---|
| Argument | unsigned short | *owbuff | 128 bytes of additional programming data |
| | unsigned short | *buff | 128 bytes of reprogramming data |
| | unsigned short | *wvf_buf | 128 bytes of programming data |
| | unsigned short | *wvf_adr | Program address |
| Return value | char | — | Result flag<br>(OK = H'00, NG = H'01, or WNG = H'02) |

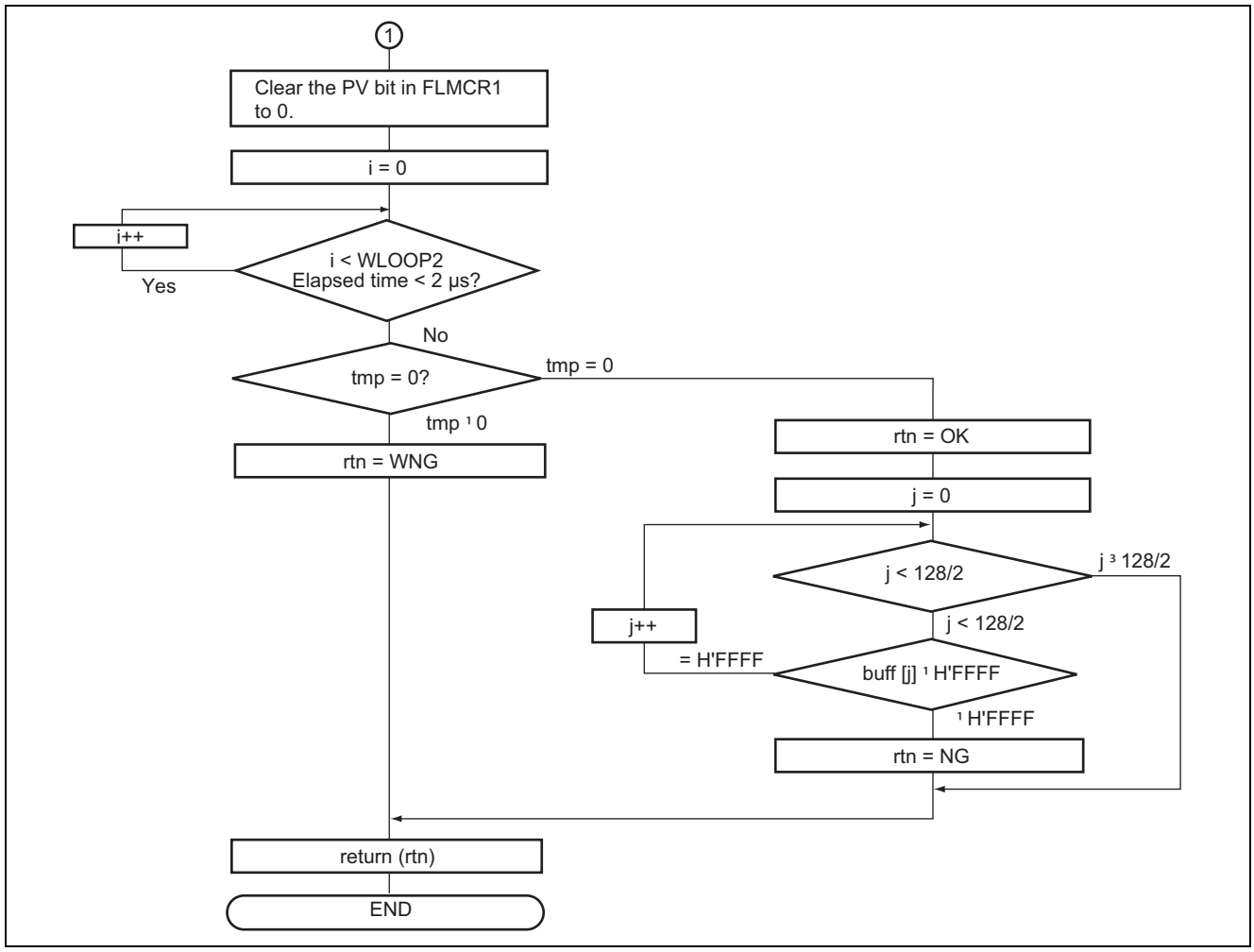
(2) **Internal Registers**

- FLMCR1: Flash memory control register 1 (Address: H'FF90)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 2 | PV | 1 | Program Verify<br>0: Cancels program-verify mode.<br>1: Enters program-verify mode. |

(3) **Flowchart**


fwritevf ()
Set the PV bit in FLMCR1 to 1.
i = 0
i++
i < WLOOP4
Elapsed time < 4 µs?
Yes
No
wad = *wvf_adr
Set verify address to wad.
j = 0
j ³ 128/2
Verification complete
j < 128/2
Verification of 128 bytes
end?
j < 128/2
wad [j*2] = H'FF
Dummy-write H'FF to the verify
address.
i = 0
j++
i++
i < WLOOP2
Elapsed time < 2 µs?
Yes
No
owbuff[j] = buff[j] | wvf_adr[j]
Additional program data computation:
Perform a logical OR of verification
result and reprogram data.
buff[j] = ~wvf_adr[j] | wvf_buf[j]
Reprogram data computation:
Perform a logical OR of the inverted
result of verificaion and program data.
tmp = ~wvf_adr [j] & wvf_buf [j]
Error Check
tmp ¹ 0?
tmp = 0
tmp ¹ 0
1


1
Clear the PV bit in FLMCR1 to 0.
i = 0
i++
i < WLOOP2
Elapsed time < 2 µs?
Yes
No
tmp = 0?
tmp = 0
tmp ¹ 0
rtn = WNG
rtn = OK
j = 0
j < 128/2
j ³ 128/2
j++
j < 128/2
= H'FFFF
buff [j] ¹ H'FFFF
¹ H'FFFF
rtn = NG
return (rtn)
END

# 7. Asynchronous Serial Communication Program

## 7.1 Hierarchical Structure

The asynchronous serial communication program performs processing to communicate with the master device.

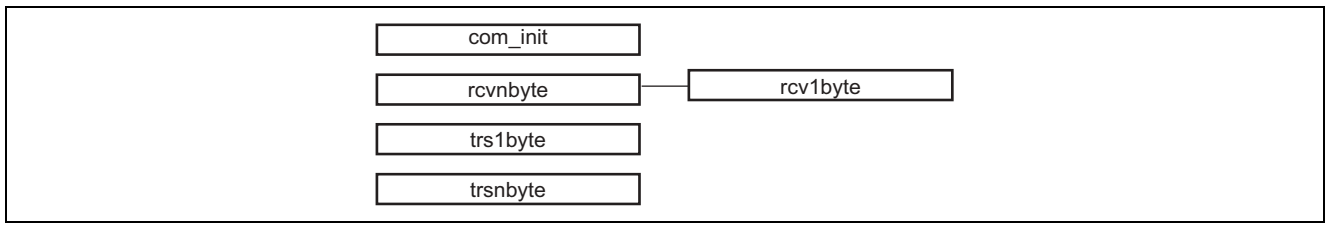



**Figure 19 Asynchronous Serial Communication Program**

## 7.2 List of Functions

**Table 20 Asynchronous Serial Communication Program Functions**

| Function Name | Summary |
|---|---|
| com_init | Initializes asynchronous serial communication settings. |
| rcv1byte | Receives one byte of data. |
| rcvnbyte | Receives n bytes of data. |
| trs1byte | Transmits one byte of data. |
| trsnbyte | Transmits n bytes of data. |

## 7.3 Modules

### 7.3.1 com_init() Function

(1) **Module Specifications**

- Functional summary
  - Initializes asynchronous serial communication settings.

**Table 21 Module Specifications**

| | Type | Variable Name | Description |
|---|---|---|---|
| Argument | None | None | None |

(2) **Internal Registers**

- SMR: Serial mode register (Address: H'FFA8)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 7 | COM | 0 | Communication Mode<br>0: Sets communication mode to asynchronous mode.<br>1: Sets communication mode to synchronous mode. |
| 6 | CHR | 0 | Character Length<br>0: Sets data length in asynchronous mode to 8 bits.<br>1: Sets data length in asynchronous mode to 7 bits. |
| 5 | PE | 0 | Parity Enable<br>0: Disables parity bit addition and check during transmission in asynchronous mode.<br>1: Enables parity bit addition and check during transmission in asynchronous mode. |
| 4 | PM | 0 | Parity Mode<br>0: Selects even parity for parity addition and check.<br>1: Selects odd parity for parity addition and check. |
| 3 | STOP | 0 | Stop Bit Length<br>0: Sets stop bit length to 1 bit in asynchronous mode.<br>1: Sets stop bit length to 2 bits in asynchronous mode. |
| 2 | MP | 0 | Multiprocessor Mode<br>0: Disables multiprocessor communication functions.<br>1: Enables multiprocessor communication functions. |
| 1 | CKS1 | 0 | Clock Select 1, 0 |
| 0 | CKS0 | 0 | When CKS1 = 0 and CKS0 = 0, clock source for the on-chip baud rate generator is set to $\phi$. |

- BRR: Bit rate register (Address: H'FFA9)
  - Functions: When BRR = H'12, the transmit bit rate that is in accordance with the baud rate generator's operating clock selected by CKS1 and CKS0 in SMR is set to 38,400 bps.
  - Setting value: H'12

- SCR3: Serial control register 3 (Address: H'FFAA)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 5 | TE | 0 | Transmit Enable<br>0: Disables transmission.<br>1: Enables transmission. |
| 4 | RE | 0 | Receive Enable<br>0: Disables reception.<br>1: Enables reception. |
| 1 | CKE1 | 0 | Clock Enable 1, 0 |
| 0 | CKE0 | 0 | When CKE1 = 0 and CKE0 = 0, internal clock is used as the clock source in asynchronous mode and the SCK32 pin functions as an I/O port. |

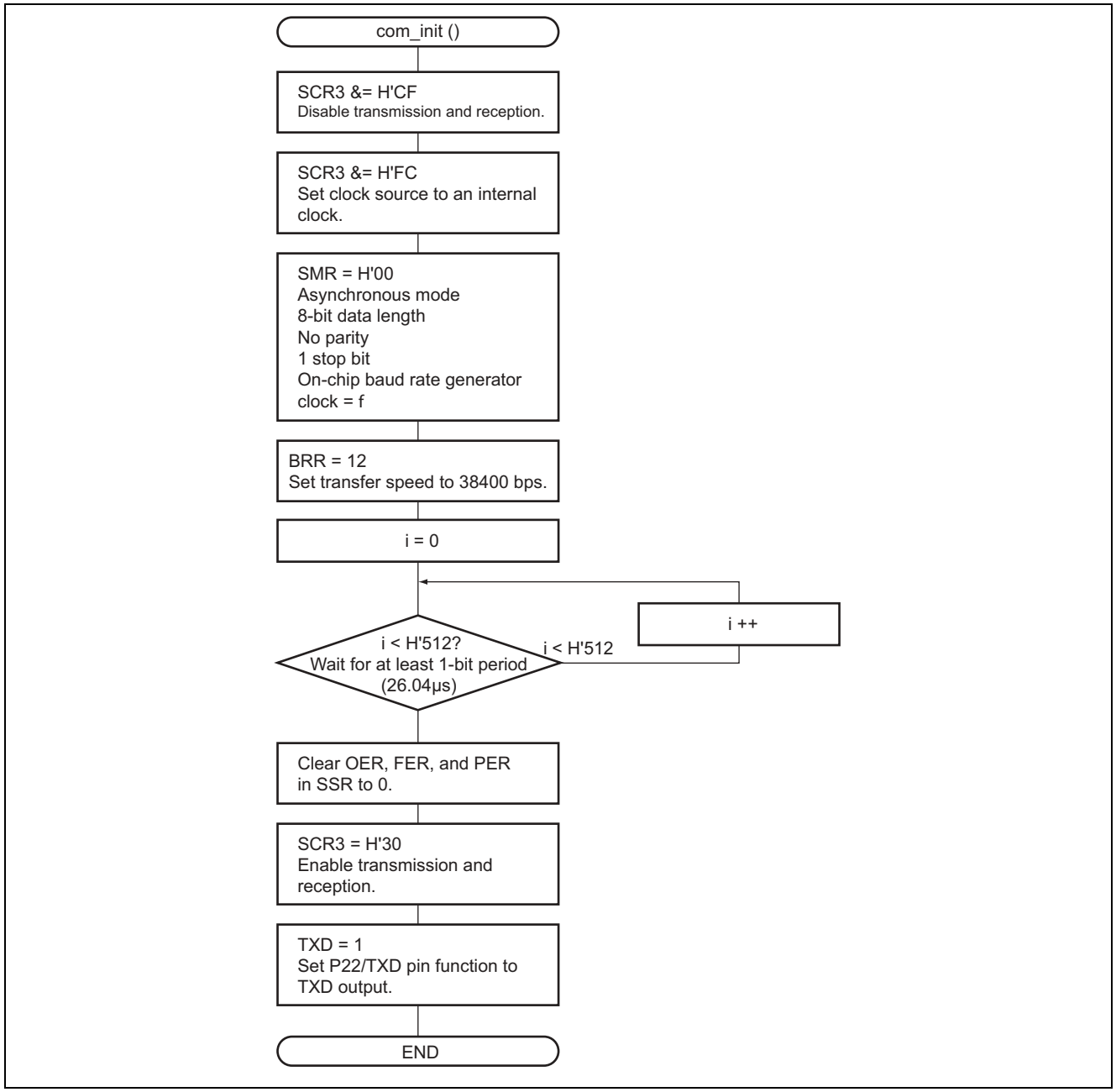
- SSR: Serial status register (Address: H'FFAC)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 7 | TDRE | Undefined | Transmit Data Register Empty<br>0: Indicates that transmit data written to TDR has not been transferred to TSR.<br>1: Indicates that transmit data has not been written to TDR or transmit data written to TDR has been transferred to TSR. |
| 6 | RDRF | Undefined | Receive Data Register Full<br>0: Indicates that receive data has not been stored in RDR.<br>1: Indicates that receive data has been stored in RDR. |
| 5 | OER | 0 | Overrun Error<br>0: Indicates that reception is in progress or complete.<br>1: Indicates that an overrun error occurred during reception. |
| 4 | FER | 0 | Framing Error<br>0: Indicates that reception is in progress or complete.<br>1: Indicates that a framing error occurred during reception. |
| 3 | PER | 0 | Parity Error<br>0: Indicates that reception is in progress or complete.<br>1: Indicates that a parity error occurred during reception. |
| 2 | TEND | Undefined | Transmit End<br>0: Indicates that transmission is in progress.<br>1: Indicates that transmission is complete. |

- PMR1: Port mode register (Address: H'FFEC)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 1 | TXD | 1 | P22/TXD Pin Function Select<br>0: The P22/TXD pin functions as the P22 pin.<br>1: The P22/TXD pin functions as the TXD output pin. |

(3) **Flowchart**


com_init ()
SCR3 &= H'CF
Disable transmission and reception.
SCR3 &= H'FC
Set clock source to an internal clock.
SMR = H'00
Asynchronous mode
8-bit data length
No parity
1 stop bit
On-chip baud rate generator clock = f
BRR = 12
Set transfer speed to 38400 bps.
i = 0
i ++
i < H'512?
Wait for at least 1-bit period (26.04µs)
i < H'512
Clear OER, FER, and PER in SSR to 0.
SCR3 = H'30
Enable transmission and reception.
TXD = 1
Set P22/TXD pin function to TXD output.
END

### 7.3.2 rcv1byte() Function

(1) **Module Specifications**

- Functional summary
  — Receives one byte of asynchronous serial data.

**Table 22 Module Specifications**

| | Type | Variable Name | Contents |
|---|---|---|---|
| Argument | None | None | None |
| Return value | unsigned char | — | One byte of received data |

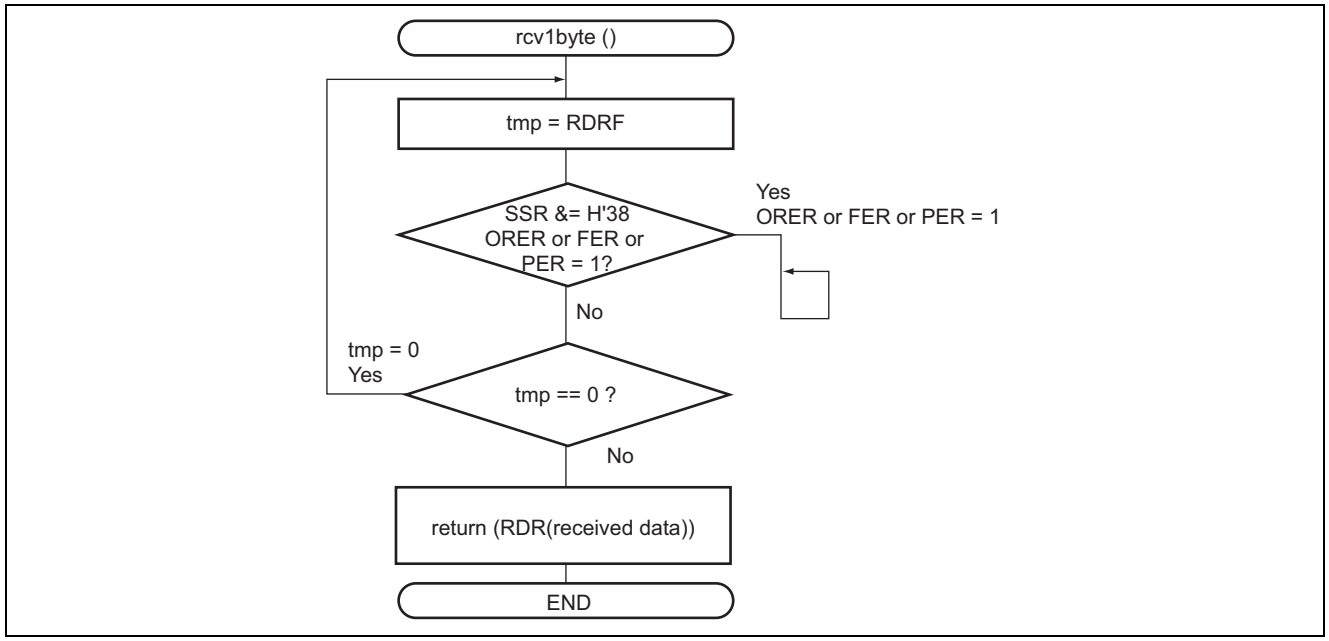
(2) **Internal Registers**

- SSR: Serial status register (Address: H'FFAC)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 6 | RDRF | Undefined | Receive Data Register Full<br>0: Indicates that receive data has not been stored in RDR.<br>1: Indicates that receive data has been stored in RDR. |
| 5 | OER | Undefined | Overrun Error<br>0: Indicates that reception is in progress or complete.<br>1: Indicates that an overrun error occurred during reception. |
| 4 | FER | Undefined | Framing Error<br>0: Indicates that reception is in progress or complete.<br>1: Indicates that a framing error occurred during reception. |
| 3 | PER | Undefined | Parity Error<br>0: Indicates that reception is in progress or complete.<br>1: Indicates that a parity error occurred during reception. |

- RDR: Receive data register (Address: H'FFAD)
  — Function: 8-bit register which stores received data.
  — Setting value: Undefined

(3) **Flowchart**


rcv1byte ()
tmp = RDRF
SSR &= H'38
ORER or FER or
PER = 1?
Yes
ORER or FER or PER = 1
No
tmp = 0
Yes
tmp == 0 ?
No
return (RDR(received data))
END

### 7.3.3 rcvnbyte() Function

(1) **Module Specifications**

- Functional summary:
  — Receives n bytes of asynchronous serial data.

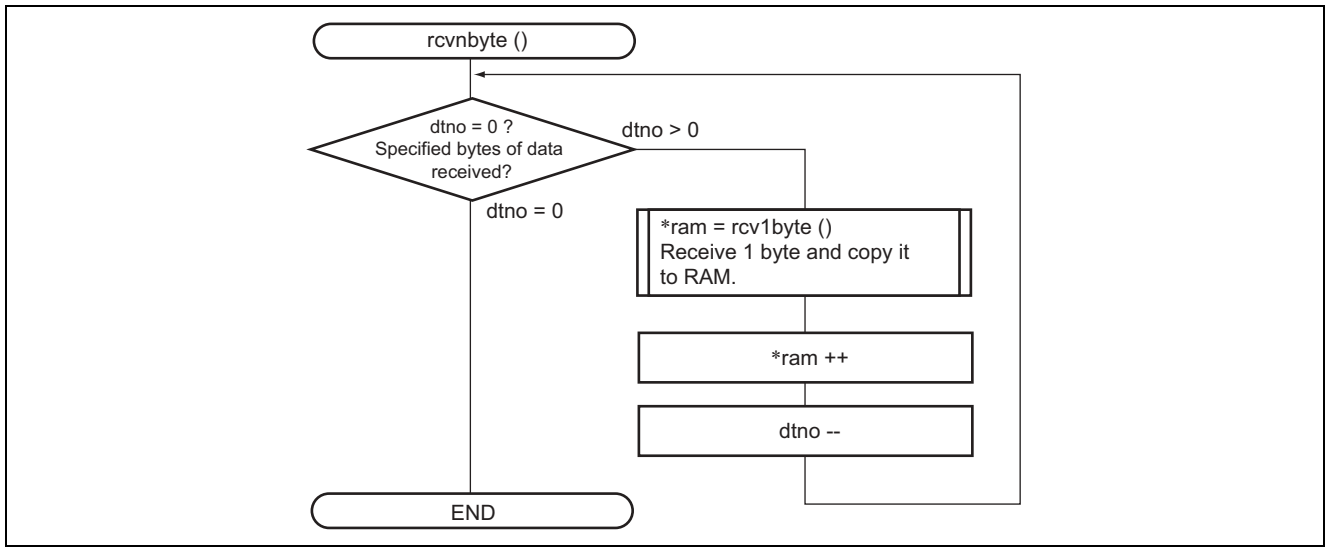
**Table 23 Module Specifications**

| | Type | Variable Name | Description |
|---|---|---|---|
| Argument | unsigned char | *ram | Start address of RAM to store receive data. |
| | unsigned char | dtno | Number of bytes to be received |

(2) **Internal Registers**

None

(3) **Flowchart**



### 7.3.4 trs1byte() Function

(1) **Module Specifications**

- Functional summary
  - Transmits 1 byte of asynchronous serial data.

**Table 24 Module Specifications**

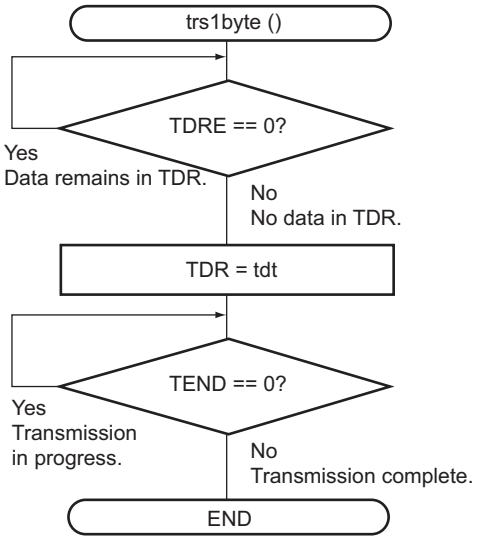
| | Type | Variable Name | Contents |
|---|---|---|---|
| Argument | unsigned char | tdt | Byte transmit data |

(2) **Internal Registers**

- TDR: Transmit data register (Address: H'FFAB)
  - Function: 8-bit register which stores transmit data.
  - Setting value: Undefined
- SSR: Serial status register (Address: H'FFAC)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 7 | TDRE | Undefined | Transmit Data Register Empty<br>0: Indicates that transmit data written to TDR has not been transferred to TSR.<br>1: Indicates that transmit data has not been written to TDR or transmit data written to TDR has been transferred to TSR. |
| 2 | TEND | Undefined | Transmit End<br>0: Indicates that transmission is in progress.<br>1: Indicates that transmission is complete. |

(3) **Flowchart**


trs1byte ()
TDRE == 0?
Yes
Data remains in TDR.
No
No data in TDR.
TDR = tdt
TEND == 0?
Yes
Transmission
in progress.
No
Transmission complete.
END

### 7.3.5 trsnbyte() Function

(1) **Module Specifications**

- Functional summary:
  — Transmits n bytes of asynchronous serial data.

**Table 25 Module Specifications**

| | Type | Variable Name | Contents |
|---|---|---|---|
| Argument | unsigned char | *tdt | Start address of transmit data |
| | unsigned char | dtno | Transmit data size |

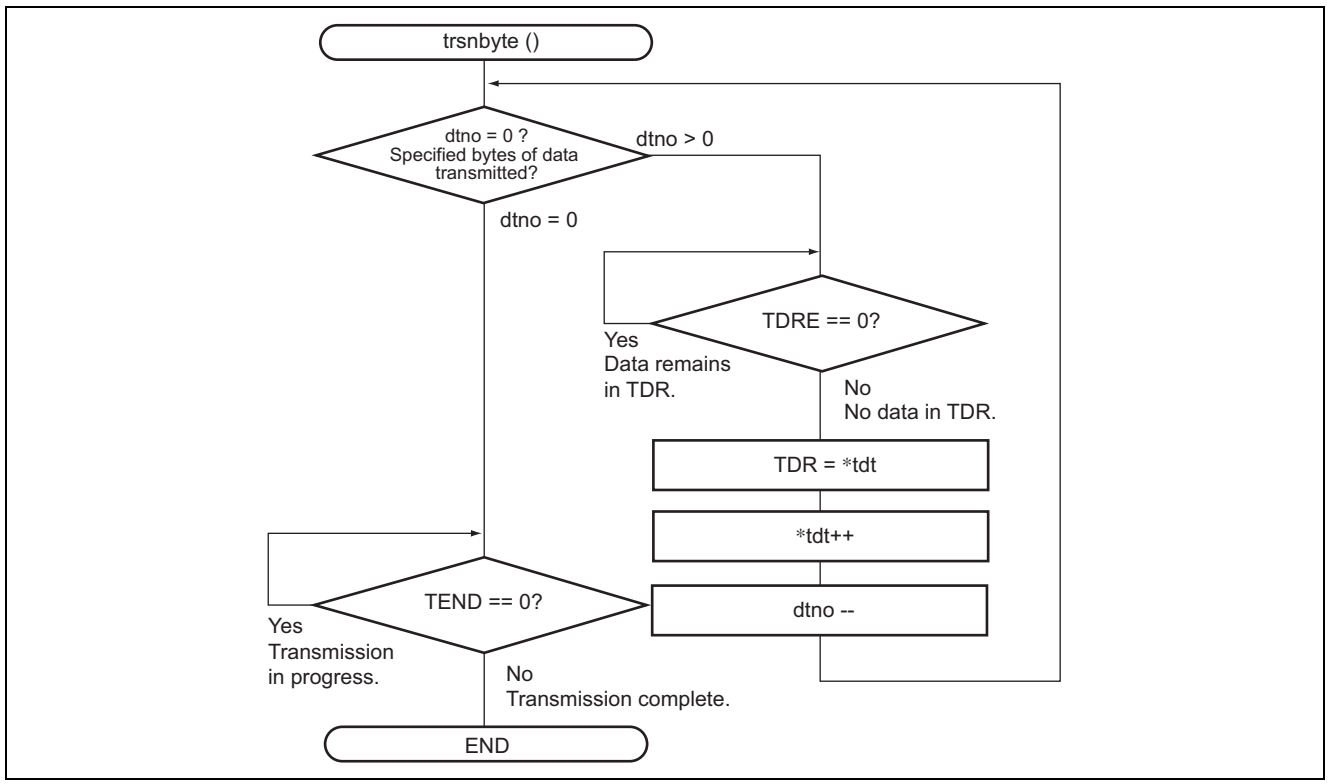
(2) **Internal Registers**

- TDR: Transmit data register (Address: H'FFAB)
  — Function: 8-bit register which stores transmit data.
  — Setting value: Undefined

- SSR: Serial status register (Address: H'FFAC)

| Bit | Bit Name | Setting | Description |
|---|---|---|---|
| 7 | TDRE | Undefined | Transmit Data Register Empty<br>0: Indicates that transmit data written to TDR has not been transferred to TSR.<br>1: Indicates that transmit data has not been written to TDR or transmit data written to TDR has been transferred to TSR. |
| 2 | TEND | Undefined | Transmit End<br>0: Indicates that transmission is in progress.<br>1: Indicates that transmission is complete. |

(3) **Flowchart**


trsnbyte ()
dtno = 0 ?
Specified bytes of data transmitted?
dtno > 0
dtno = 0
TDRE == 0?
Yes
Data remains in TDR.
No
No data in TDR.
TDR = *tdt
*tdt++
dtno --
TEND == 0?
Yes
Transmission in progress.
No
Transmission complete.
END

## 8. Link Address Specifications

| Section Name | Address |
|---|---|
| CV1 | H'0000 |
| P | H'0100 |
| PASSCI | H'0400 |
| DSMPL1 | H'0800 |
| DSMPL2 | H'0C00 |
| DSMPL3 | H'0FF6 |
| FZTAT, PFZTAT, DFZTAT, FZEND | H'1000 |
| RAM, PRAM, DRAM | H'F780 |

## Revision Record

| | | Description | |
|---|---|---|---|
| Rev. | Date | Page | Summary |
| 1.00 | Jul.28.04 | — | First edition issued |
| | | | |
| | | | |
| | | | |
| | | | |

## Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

## Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.