# RL78/F23, F24

## Safety Function

R01AN6624EJ0110
Rev.1.10
2023.12.31

### Introduction

This application note describes the safety functions implemented on the RL78/F23, RL78/F24 microcontrollers (MCUs).

### Target Device

- RL78/F23, RL78/F24

### Contents

# 1. Overview of the Safety Functions

To detect any errors and failures with the built-in self-test function, the RL78/F23, RL78/F24 MCUs have the following safety functions.

**(1) CRC (cyclic redundancy check) operation functions (High-speed CRC operation & general-purpose CRC operation)**

**High-speed CRC operation:** This check is executed on the entire code flash memory area after stopping the CPU (making the CPU transition to HALT mode).

**General-purpose CRC operation:** The general-purpose CRC can be used in the code flash memory area. Also, it can be used for multi-purpose data check such as serial communication.

**(2) Internal RAM ECC function**

This function detects and corrects data corruption (bit errors) during a read access to the internal RAM and notifies the error detected/corrected by generating an interrupt.

**(3) CAN-RAM ECC function**

This function detects and corrects data corruption (bit errors) during a read and write access to the CAN-RAM and notifies the error detected/corrected by generating an interrupt.

**(4) Code Flash-ECC function**

This function detects and corrects data corruption (bit errors) during a read access to the code flash memory and notifies the error detected/corrected by generating an interrupt.

**(5) CPU stack pointer monitor function**

This function detects an overflow and underflow of the stack pointer (SP) and generates an interrupt in response.

**(6) Clock monitor function**

This function detects an oscillation stop of the main system clock ($f_{MAIN}$) and main/PLL selection clock ($f_{MP}$) using the low-speed on-chip oscillation clock ($f_{IL}$) and accordingly generates a reset signal or interrupt.

**(7) RAM guard function**

This function protects data in RAM that is to be guarded from any erroneous writing when a CPU runaway etc., occurs.

**(8) SFR guard function**

This function protects the SFRs (special function registers for port functions, interrupts, clock control, and voltage detector control) that is to be guarded from any erroneous writing when a CPU runaway or any problem occurs.

**(9) Invalid memory access detection function**

This function detects any invalid access to the memory area when a CPU runaway or any problem occurs and generates a reset signal.

**(10) Frequency detection function**

The function detects whether or not the clock is operating on an abnormal frequency by comparing the high-speed on-chip oscillator clock ($f_{IH}$), external X1 oscillation clock ($f_{MX}$), or PLL clock ($f_{PLL}$) with the low-speed on-chip oscillator clock ($f_{IL}$).

**(11) A/D test function**

This function supports a self-check of A/D conversion, and the build-in pre-charge and dis-charge function to be able to assist disconnection detection of the wire which is connected to analog inputs.

**(12) Digital output signal level detection function for I/O ports**

This function detects any output abnormality by reading the digital output level (high or low) of the pin when the port is set to output mode.

**(13) Watchdog timer function**

This function detects an inadvertent program loop, and assert internal reset signal to CPU.

# 2. Flash Memory CRC Operation Function (High-speed CRC)

## 2.1 Overview of High-speed CRC Operation

The high-speed CRC operation is a function to perform a high-speed check on the entire code flash memory area by stopping the CPU (by making the CPU enter HALT mode). Any failure in the code flash memory can be detected by comparing the expected value of the CRC function which is calculated beforehand with the result of the high-speed CRC operation.

The CRC generator polynomial used complies with "$X^{16} + X^{12} + X^5 + 1$" of CRC-16-CCITT. The high-speed CRC operates in MSB first order from bit 31 to bit 0.

Since the CPU is stopped during the high-speed CRC operation, it is impossible to run the user software. Confirm the processing time for the high-speed CRC operation function listed in Table 2-1 and use this function according to the specifications of your system.

**Table 2-1 Processing Time of High-speed (HS) CRC Operation**

| Range of HS CRC operation Note | Processing time ($f_{CLK}$=40MHz) | Register setting |
|---|---|---|
| 16KB (00000H - 03FFBH) | 4095 clocks (approx.102 μs) | CRC0CTL.FEA[5:0] = 000000B |
| 32KB (00000H - 07FFBH) | 8191clocks (approx.205 μs) | CRC0CTL.FEA[5:0] = 000001B |
| 48KB (00000H - 0BFFBH) | 12287 clocks (approx.307 μs) | CRC0CTL.FEA[5:0] = 000010B |
| 64KB (00000H - 0FFFBH) | 16383 clocks (approx.410 μs) | CRC0CTL.FEA[5:0] = 000011B |
| 80KB (00000H - 13FFBH) | 20479 clocks (approx.512 μs) | CRC0CTL.FEA[5:0] = 000100B |
| 96KB (00000H - 17FFBH) | 24575 clocks (approx.614 μs) | CRC0CTL.FEA[5:0] = 000101B |
| 112KB (00000H - 1BFFBH) | 28671 clocks (approx.717 μs) | CRC0CTL.FEA[5:0] = 000110B |
| 128KB (00000H - 1FFFBH) | 32767 clocks (approx.819 μs) | CRC0CTL.FEA[5:0] = 000111B |
| 144KB (00000H - 23FFBH) | 36863 clocks (approx.922 μs) | CRC0CTL.FEA[5:0] = 001000B |
| 160KB (00000H - 27FFBH) | 40959 clocks (approx.1024 μs) | CRC0CTL.FEA[5:0] = 001001B |
| 176KB (00000H - 2BFFBH) | 45055 clocks (approx.1126 μs) | CRC0CTL.FEA[5:0] = 001010B |
| 192KB (00000H - 2FFFBH) | 49151 clocks (approx.1229 μs) | CRC0CTL.FEA[5:0] = 001011B |
| 208KB (00000H - 33FFBH) | 53247 clocks (approx.1331 μs) | CRC0CTL.FEA[5:0] = 001100B |
| 224KB (00000H - 37FFBH) | 57343 clocks (approx.1434 μs) | CRC0CTL.FEA[5:0] = 001101B |
| 240KB (00000H - 3BFFBH) | 61439 clocks (approx.1536 μs) | CRC0CTL.FEA[5:0] = 001110B |
| 256KB (00000H - 3FFFBH) | 65535 clocks (approx.1638 μs) | CRC0CTL.FEA[5:0] = 001111B |

**Note** The last four bytes of the flash memory (e.g., an area of 003FFCH-003FFFH of a 16-KB memory) are not included in the range of high-speed CRC operation.

## 2.2 Registers used for High-speed CRC Operation

The registers used for the high-speed CRC operation are described below.

(1) **Flash memory CRC control register (CRC0CTL)**

This register enables/disables the high-speed CRC operation and specifies the calculation range. This CRC0CTL register can be accessed by a 1-bit memory manipulation instruction (CRC0EN) or an 8-bit memory manipulation instruction.

Address: F02F0H After reset: 00H R/W

| Symbol | <7> | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CRC0CTL | CRC0EN | 0 | FEA5 | FEA4 | FEA3 | FEA2 | FEA1 | FEA0 |

| Bit Name | Description |
|---|---|
| CRC0EN | 0：Stops the high-speed CRC arithmetic unit.<br>1：Starts the high-speed CRC operation upon execution of the HALT instruction. |
| FEA[5:0] | Specify the high-speed CRC operation range. Note<br>000000B: 16KB 000001B: 32KB 000010B: 48KB 000011B: 64KB<br>000100B: 80KB 000101B: 96KB 000110B: 112KB 000111B: 128KB<br>001000B: 144KB 001001B: 160KB 001010B: 176KB 001011B: 192KB<br>001100B: 208KB 001101B: 224KB 001110B: 240KB 001111B: 256KB<br>Other than the above ranges: Setting prohibited |

**Note** Be sure to set the calculation range to be within the memory range of the product used.

(2) **Flash memory CRC operation result register (PGCRCL)**

This register stores the results of high-speed CRC operation. This register can be accessed by a 16-bit memory manipulation instruction.

Address: F02F2H After reset: 0000H R/W

| Symbol | 15 | 0 |
|---|---|---|
| PGCRCL | PGCRC[15:0] | |

| Bit Name | Description |
|---|---|
| PGCRC[15:0] | Stores the results of high-speed CRC operation. Note |

**Note** This register is writable only when the value of the CRC0EN bit is 1.

## 2.3 Flow Chart of High-speed CRC operation

Figure 2-1 is a flow chart of the high-speed CRC operation.

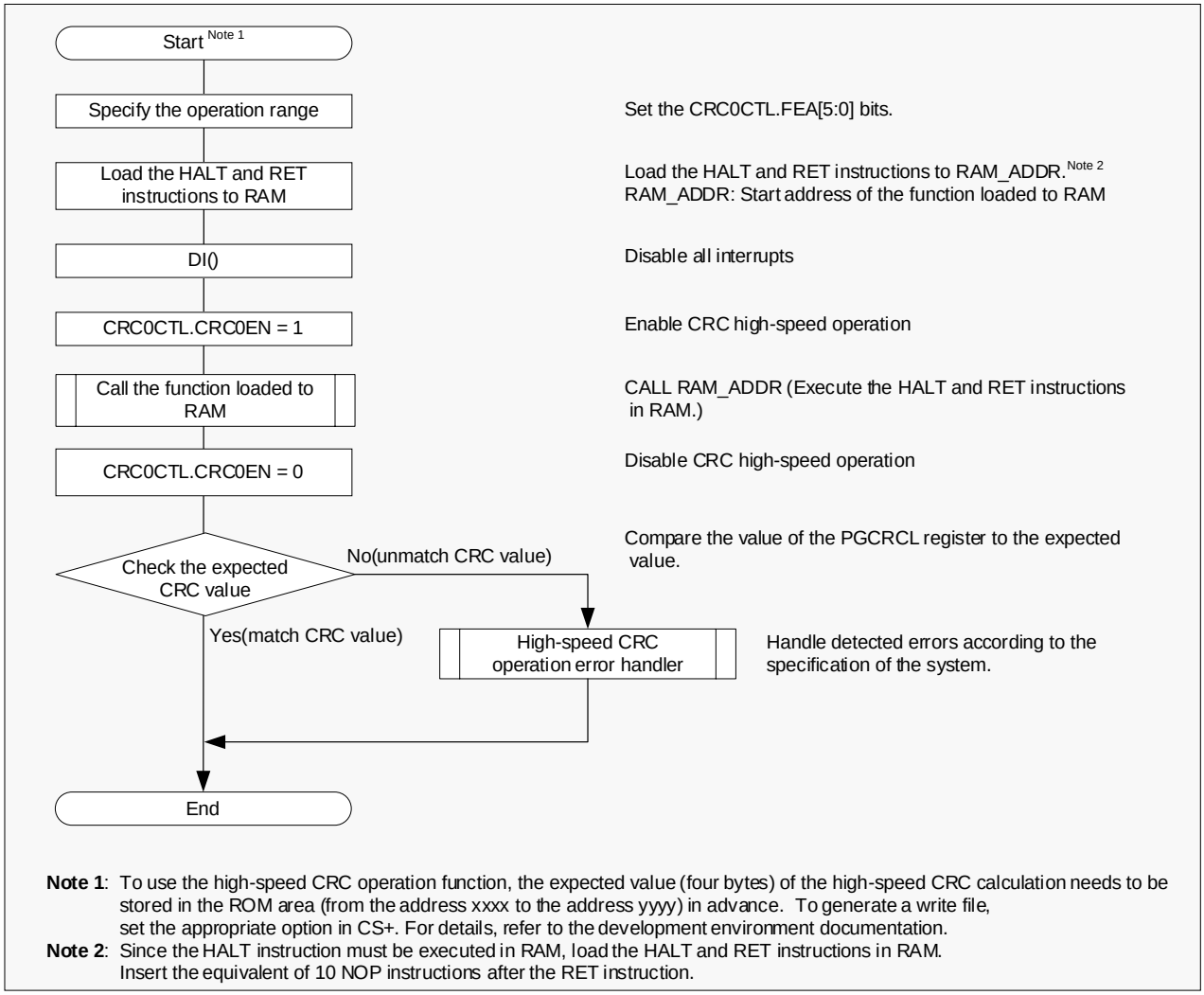
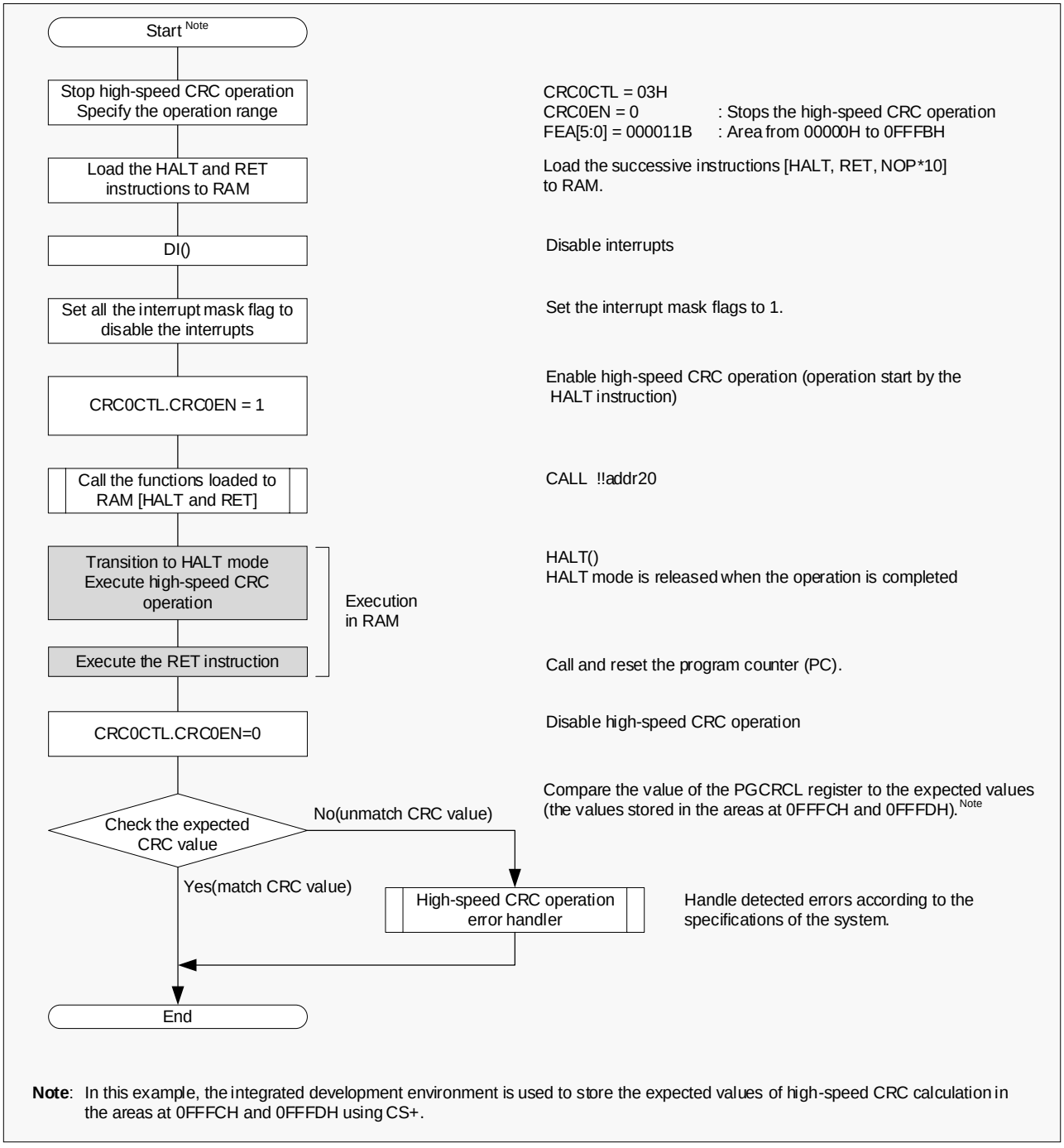



**Figure 2-1 Flow Chart of High-speed CRC operation**

## 2.4 Example of High-speed CRC operation

Figure 2-2 is an example of the high-speed CRC operation function for a product whose ROM size is 64KB.




**Note**: In this example, the integrated development environment is used to store the expected values of high-speed CRC calculation in the areas at 0FFFCH and 0FFFDH using CS+.

**Figure 2-2 Example of High-speed CRC operation**

## 2.5 Cautions when Using High-speed CRC operation

The following are the cautions when using the high-speed CRC operation function.

(1) The high-speed CRC operation starts upon execution of the HALT instruction in RAM. HALT mode is released when the calculation is finished. Therefore, before executing the HALT instruction, be sure to disable the interrupts (DI) and also to set all the interrupt mask flags to 1 (interrupt processing disabled).

(2) Since the CPU is stopped during high-speed CRC operation, it is impossible to run the user software. When using this function, confirm that the processing time of the high-speed CRC operation function will not lead to problems. (See Table 2-1.)

(3) The RL78 CPU core performs pre-reading when an instruction code is fetched. Therefore, to execute the instruction in the RAM area, the subsequent addresses (after the instruction) up to a size of 10 bytes need to be initialized.

(4) When the expected value of the high-speed CRC operation function is calculated using the integrated development environment, the result can be represented in a HEX file, S-record file. However, it will not be represented in a load module file.

(5) The monitor program is allocated to the code flash memory area. Accordingly, high-speed CRC calculation result will not match its expected value during on-chip debugging.

(6) When the IDRDEN bit in security option byte (000C4H) is "0", reading on-chip debug security ID (000C6H to 000D5H) and flash serial programming security ID (000D6H to 000E5H) during high-speed CRC operation will read the actual set value instead of 00H for the read protection function.

# 3. CRC Operation Function (General-purpose CRC)

## 3.1 Overview of General-purpose CRC operation

The function of general-purpose CRC is to write calculation data to the CRC input register (CRCIN) and to store the calculation result in the CRC data register (CRCD) while the CPU is operating. This function can be used for a wide variety of purposes, such as serial communication or other applications.

The CRC generator polynomial supports "$X^{16} + X^{12} + X^5 + 1$" of CRC-16-CCITT and "$X^4 + X^3 + X^2 + 1$" of SENT compliant.

## 3.2 General-purpose CRC Operation Registers

The registers used for the general-purpose CRC operation are described below.

(1) **CRC input register (CRCIN)**

This register is used to set data used for CRC operation. This register can be accessed by an 8-bit memory manipulation instruction.

Address: FFFACH After reset: 00H R/W

| Symbol | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|
| CRCIN | CRCIN[7:0] | | | | | | | |

| Bit Name | Description |
|---|---|
| CRCIN[7:0] | Specifies the range of input data for CRC operation.<br>When supporting CRC-CCITT: 00H-FFH<br>When conforming to SENT: 00H-0FH |

(2) **CRC operation mode control register (CRCMD)**

This register selects a calculation mode (CRC generator polynomial) for the general-purpose CRC arithmetic unit. This register can be accessed by an 8-bit memory manipulation instruction.

Address: F02F9H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CRCMD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | POLYSEL |

| Bit Name | Description |
|---|---|
| POLYSEL | 0: CRC-CCITT ($X^{16}+X^{12}+X^5+1$)<br>1: Conforms to SENT ($X^4+X^3+X^2+1$) |

(3) **CRC data register (CRCD)**

This register stores the results of general-purpose CRC operation. This register can be accessed by a 16-bit memory manipulation instruction.

Address: F02FAH After reset: 0000H R/W

| **Symbol** | 15 | 0 |
|---|---|---|
| CRCD | CRCD[15:0] | |

| Bit Name | Description |
|---|---|
| CRCD[15:0] | Stores the result of CRC operation.<br>When supporting CRC-CCITT: 0000H-FFFFH<br>When conforming to SENT: 0000H-000FH |

## 3.3 Flow Chart of General-purpose CRC Operation

**Figure 3-1** is a flow chart of the general-purpose CRC operation function.

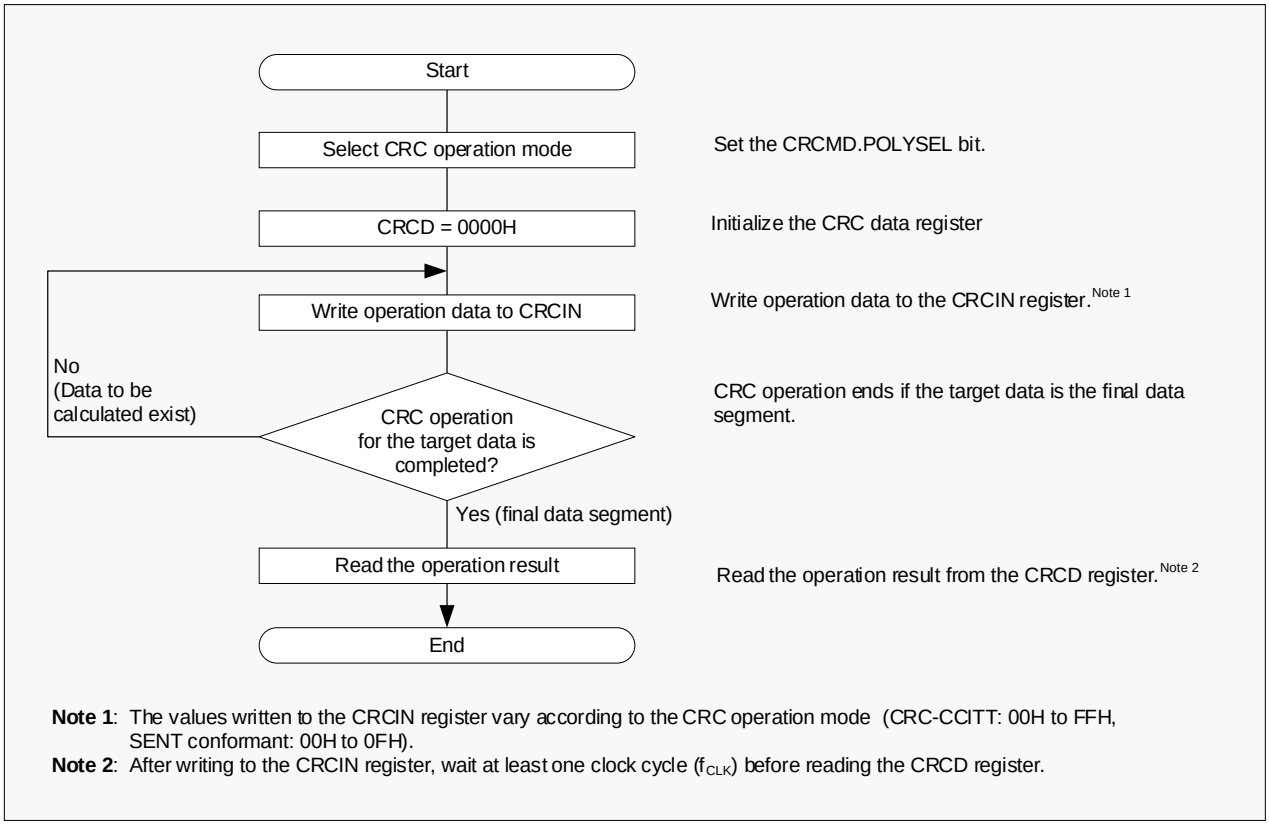



**Figure 3-1 Flow Chart of General-purpose CRC Operation**

## 3.4 Cautions when Using General-purpose CRC Operation

The following are the cautions when using the general-purpose CRC operation function.

(1) After writing to the CRCIN register, wait at least one clock cycle ($f_{CLK}$) before reading the CRCD register.

(2) Do not set any software break in the target area of CRC operation. Setting a software break in that area will alter the CRC operation result. This is because the debugger changes the row where the software break is to be set into a break instruction.

(3) In case IDRDEN bit in security option byte (000C4H) is "0", reading on-chip debug security ID (000C6H to 000D5H) and flash serial programming security ID (000D6H to 000E5H) are 00H of read-protected.

# 4. Internal RAM ECC Function

## 4.1 Overview of the Internal RAM ECC Function

The Internal RAM ECC function detects any data corruption (bit error) and accordingly generates an interrupt. Also, if the error detected is a 1-bit error, this function corrects the corruption data.

For the write access to the internal RAM, this function generates a 4-bit ECC code and a 1-bit parity bit for 8-bit data written to RAM. For the read access to RAM, this function checks the ECC code and parity bits and outputs the bit error detection interrupt request (INTRAM) if a bit error is detected.

**Table 4-1 Operation of the Internal RAM ECC function**

| Bit corruption (bit error) | | | Interrupt notification (INTRAM) | ECCER register | ERADR register | Read value |
|---|---|---|---|---|---|---|
| Data bit | ECC code | Parity bit | | DBERR bit | | |
| No bit error | | | – | – | – | Expected value |
| 1-bit error | – | – | Request generation Note 1 | 0 Note 1 | Address storage Note 1 | Expected value |
| – | 1-bit error | – | Request generation Note 1 | 0 Note 1 | Address storage Note 1 | Expected value |
| – | – | 1-bit error | – | – | – | Expected value |
| 2-bit error | | | Request generation Note 2 | 1 | Address storage | Indefinite Note 2 |
| 3-bit or more error | | | Indefinite Note 3 | Indefinite Note 3 | Indefinite Note 3 | Indefinite Note 3 |

**Remark** In the table above, "–" means "no bit error" and "no update" for the item "Bit corruption (bit error) and other items (Input notification, ECCER register, ERADR register and Read value), respectively.

**Notes 1.** When the value of the IEN bit in the ECCIER register is 1 (Interrupt enabled), the interrupt request signal (INTRAM) is generated. Also, in this case, the ERADR register and the DBERR bit will be updated.

**2.** An interrupt request signal will be generated regardless of the setting of the IEN bit. In this case, the ERADR register and the DBERR bit will be updated. Since the error detected is a multiple-bits (two or more) error, the expected data correction will not be performed.

**3.** Since the error detected is a multiple-bits (two or more) error, the expected data correction will not be performed. In addition, error detection will not be checked correctly.

## 4.2 Registers used for The Internal RAM ECC Function

The registers used for the internal RAM ECC function are described below.

(1) **Error address store register (ERADR)**

This register stores the address corresponding to a bit error detected. This register can be read by a 16-bit memory manipulation instruction.

Address: F0200H After reset: 0000H R

| Symbol | 15 | 0 |
|---|---|---|
| ERADR | ERAD[15:0] | |

| Bit Name | Description |
|---|---|
| ERAD[15:0] | Stores the address of a bit error detected. Note |

**Note** The register value is updated every time a bit error interrupt request is generated.

(2) **1-bit error detection interrupt enable register (ECCIER)**

This register enables/disables the interrupt when a 1-bit error is detected. This register can be accessed by an 8-bit memory manipulation instruction.

Address: F0202H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ECCIER | 0 | 0 | 0 | 0 | 0 | 0 | 0 | IEN |

| Bit Name | Description |
|---|---|
| IEN | 0: Disables the interrupt generation for a 1-bit error detected. Note<br>1: Enables the interrupt generation for a 1-bit error detected. |

**Note** When a 2-bit error is detected, a bit-error detection interrupt request (INTRAM) is generated regardless of the setting of the IEN bit.

(3) **Bit error detection register (ECCER)**

This register checks whether the bit error detected is a 1-bit error (correction of errors detected) or a 2-bit error. This register is accessed by an 8-bit memory manipulation instruction.

Address: F0203H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ECCER | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DBERR |

| Bit Name | Description |
|---|---|
| DBERR Note | 0: A 1-bit error detected (Error correction)<br>1: A 2-bit error detected |

**Note** If the bit error interrupt request (INTRAM) has not been generated, the value of the DBERR bit is invalid.

(4) **ECC test protect register (ECCTPR)**

This register enables/disables the access to the ECC test mode register (ECCTMDR). This register can be accessed by an 8-bit memory manipulation instruction.

Address: F0204H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ECCTPR | 0 | 0 | 0 | 0 | 0 | TPR[2:0] | | |

| Bit Name | Description |
|---|---|
| TPR[2:0] | 111B: Enables the access to the ECCTMDR register.<br>Other than 111B: Disables the access to the ECCTMDR register. |

(5) **ECC test mode register (ECCTMDR)**

This register selects an ECC test mode. This register can be accessed by an 8-bit memory manipulation instruction.

Before accessing this register, write 07H to the ECCTPR register.

Address: F0205H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ECCTMDR | 0 | 0 | 0 | 0 | 0 | TMD[2:0] | | |

| Bit Name | Description |
|---|---|
| TMD[2:0] | 000B: Normal operating mode<br>001B: ECC test mode<br>Other than above: Setting prohibited |

(6) **Write data inversion register (ECCDWRVR)**

This register is used to confirm that the ECC is operating correctly by inverting the parity bits of write data and ECC code in ECC test mode. This register can be accessed by a 16-bit memory manipulation instruction.

Address: F0206H After reset: 0000H R/W

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| ECCDWRVR | 0 | 0 | 0 | PRTYRV | ECCRV[3:0] | | | |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DWRV[7:0] | | | | | | | |

| Bit Name | Description |
|---|---|
| PRTYRV | 0: Parity bit not inverted<br>1: Parity bit inverted |
| ECCRV[3:0] | 0: Bit (i) of ECC code not inverted<br>1: Bit (i) of ECC code inverted<br>(i: 3-0) |
| DWRV[7:0] | 0: Bit (j) of internal RAM write data not inverted<br>1: Bit (j) of internal RAM write data inverted<br>(j: 7-0) |

## 4.3 Flow Chart of Internal RAM ECC Function

Figure 4-1 is a flow chart of the internal RAM ECC function.

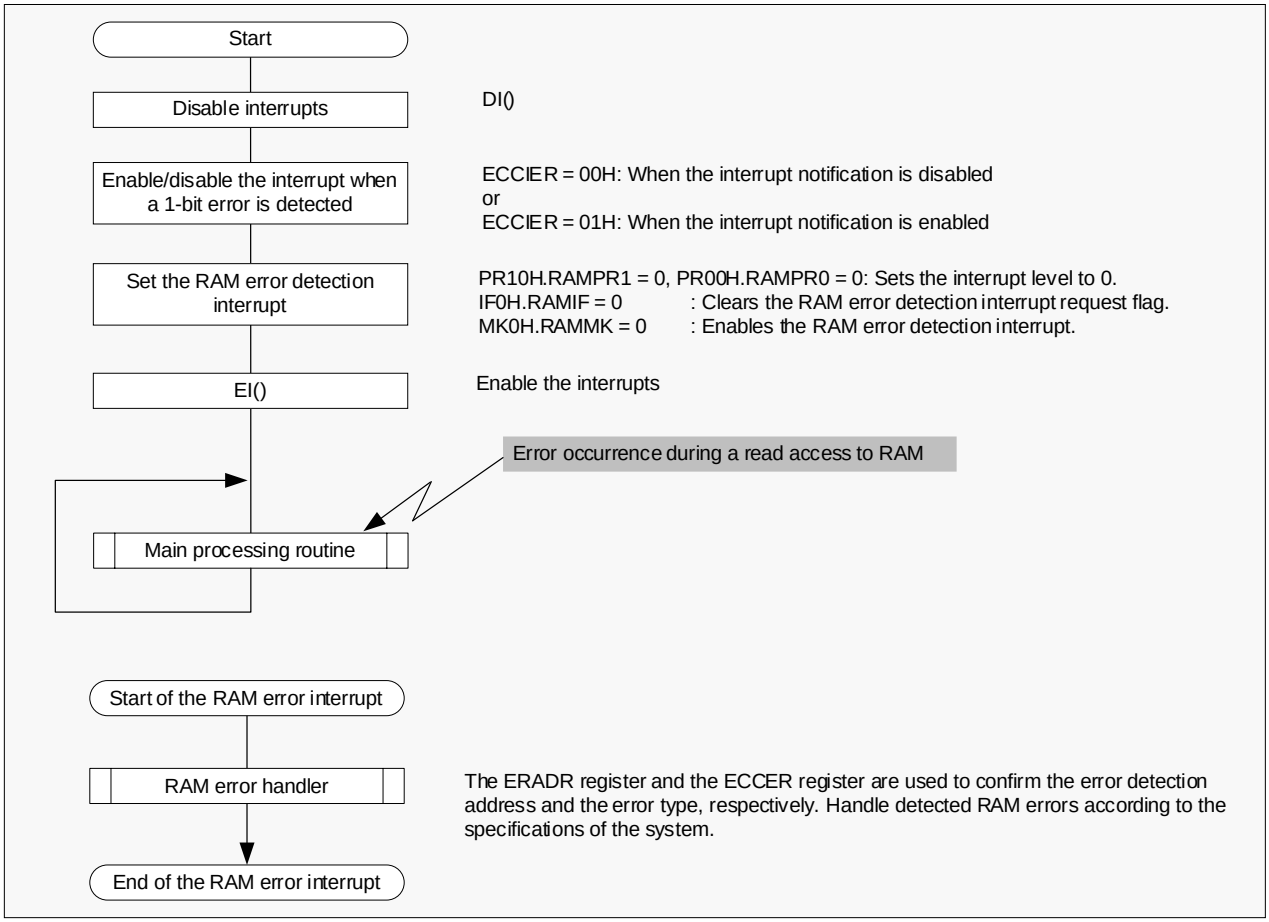



**Figure 4-1 Flow Chart of the internal RAM ECC Function**

## 4.4 ECC Test Mode

In ECC test mode, operations of the internal RAM ECC function can be checked by writing a bit-inverted value to write data/ECC code/parity bits and by reading the target RAM. To enable this mode, the internal RAM must not be accessed. (Enable this mode when, for example, initialization of internal RAM is being executed.)

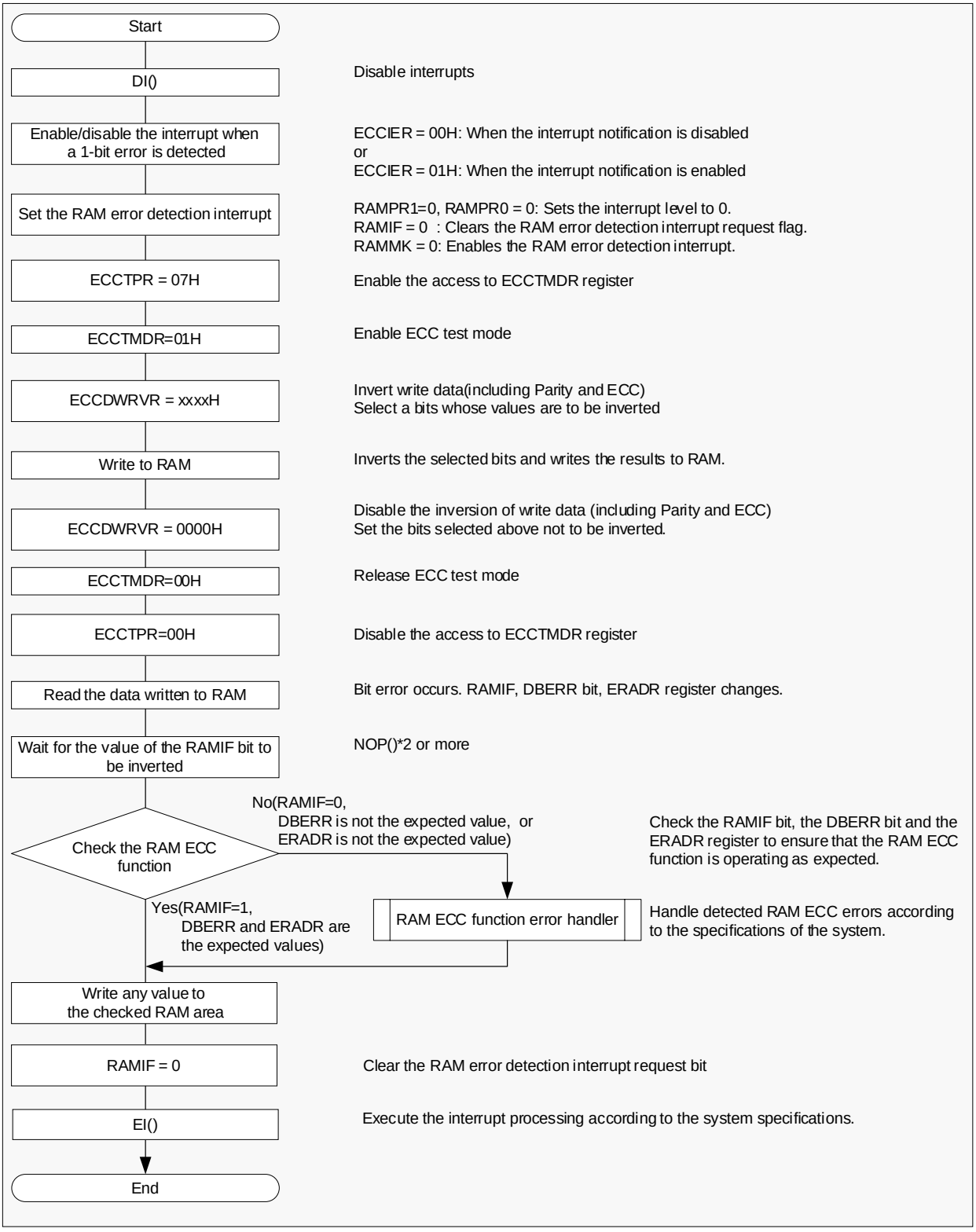



**Figure 4-2 Flow Chart of ECC Test Function**

Table 4-2 **is an example of the setting of the ECC test mode.**

**Table 4-2 Setting Example of ECC test mode**

| ECCDWRVR register | | | Interrupt notification (INTRAM) | ECCER register | ERADR register | Read value |
|---|---|---|---|---|---|---|
| DWRV[7:0] | ECCRV[3:0] | PRTYRV | | DBERR bit | | |
| No bit inversion | | | – | – | – | Expected value |
| 1-bit inversion | – | – | Request generation Note 1 | 0 Note 1 | Address storage Note 1 | Expected value |
| – | 1-bit inversion | – | Request generation Note 1 | 0 Note 1 | Address storage Note 1 | Expected value |
| – | – | 1-bit inversion | – | – | – | Expected value |
| 2-bit inversion | – | – | Request generation | 1 | Address storage | Indefinite Note 2 |
| 1-bit inversion | 1-bit inversion | – | Request generation | 1 | Address storage | Indefinite Note 2 |
| 1-bit inversion | – | 1-bit inversion | Request generation | 1 | Address storage | Expected value |
| – | 2-bit inversion | – | Request generation | 1 | Address storage | Indefinite Note 2 |
| – | 1-bit inversion | 1-bit inversion | Request generation | 1 | Address storage | Expected value |
| 3-bit or more inversion | | | Indefinite Note 3 | Indefinite Note 3 | Indefinite Note 3 | Indefinite Note 3 |

**Remark** In the table above, "–" means "no bit inversion" and "no update" for the item "ECCDWRVR register" and other items (Input notification, ECCER register, ERADR register and Read value), respectively.

**Notes 1.** When the value of the IEN bit in the ECCIER register is 1 (Interrupt enabled), the interrupt request signal (INTRAM) is generated. In this case, the ERADR register and the DBERR bit will be updated.

**2.** Since the error detected is a multiple-bit (two or more) error, the expected data correction will not be performed.

**3.** Since the error detected is a multiple-bit (three or more) error, the expected data correction will not be performed. In this case, error detection will not be checked correctly.

## 4.5 Cautions when Using The Internal RAM ECC Function

The following are the cautions when using the internal RAM ECC function.

(1) When a 1-bit error is detected, the expected value (a value written) can be read since the error detected is to be corrected. However, since the RAM value will not be rewritten, when the 1-bit error detection interrupt enable bit is set to 1 (Interrupt enabled), the interrupt request (INTRAM) is generated every time the address where this error has been detected is read.

(2) Since the internal RAM ECC function is not executed during on-chip debugging, do not use the ECC test mode.

(3) When a 2-bit error is detected, the bit error detection interrupt (INTRAM) is generated regardless of the setting of the IEN bit (enables/disables the interrupt when a 1-bit error is detected) in the ECCIER register.

# 5. CAN-RAM ECC Function (RL78/F24 only)

## 5.1 Overview of CAN-RAM ECC Function

The CAN-RAM area (F0420H to F067FH) on RS-CANFD lite supports ECC function. CAN-RAM ECC function is used to detect erroneous data (bit errors), generate interrupt requests. If only one bit is in error, the data are corrected.

## 5.2 Registers used for CAN-RAM ECC Function

The register used for the CAN-RAM ECC function are described below.

(1) **CAN-RAM ECC control register (CFDECCTL)**

This register is used to manage the status of CAN-RAM ECC function. This register can be accessed by the 8-bit or 16-bit memory manipulation instruction.

Address: F07C0H After reset: 0010H R/W

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| CFDECCTL | EMCA[1:0] | | 0 | 0 | ECOVFF | ECER2C | ECER1C | 0 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | ECERVF | EC1ECP | EC2EDIC | EC1EDIC | ECER2F | ECER1F | 0 |

| Bit Name | Description |
|---|---|
| EMCA[1:0] | Access control of ECERVF.<br>01B: Write access enabled.<br>Other than above: Write access disabled. |
| ECOVFF | ECC overflow detection.<br>0: Overflow is not occurred after reset.<br>1: Error address register overflowed. |
| ECER2C | Flag clear bit of ECC 2-bit error detection.<br>0: No operation.<br>1: ECER2F bit clear. |
| ECER1C | Flag clear bit of ECC 1-bit error detection.<br>0: No operation.<br>1: ECER1F bit clear. |
| ECERVF | Control ECC error judgement.<br>0: Error judgement disabled.<br>1: Error judgement enabled. |
| EC1ECP | Control ECC 1-bit error correction.<br>0: Enable 1-bit error correction upon error detection.<br>1: Disable 1-bit error correction upon error detection. |
| EC2EDIC | Control ECC 2-bit error detection.<br>0: When a 2-bit error is detected interrupt disabled.<br>1: When a 2-bit error is detected interrupt enable. |
| EC1EDIC | Control interrupt at ECC 1-bit error detection.<br>0: When a 1-bit error is detected, interrupt disabled.<br>1: When a 1-bit error is detected, interrupt enabled. |
| ECER2F | Flag of ECC 2-bit error detection.<br>0: A 2-bit error has not occurred. (Not occurred after this bit was cleared.)<br>1: A 2-bit error has occurred. |
| ECER1F | Flag of ECC 1-bit error detection.<br>0: A 1-bit error has not occurred. (Not occurred after this bit was cleared.)<br>1: A 1-bit error has occurred. |

(2) **CAN-RAM ECC test mode control register (CFDECTMC)**

This register is used to control the test function of CAN-RAM ECC function. This register can be accessed by the 8-bit or 16-bit memory manipulation instruction.

Address: F07C4H After reset: 0000H R/W

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| CFDECTMC | ETMA[1:0] | | 0 | 0 | 0 | 0 | 0 | 0 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ECTMEC | 0 | 0 | ECTRRS | 0 | 0 | ECDCS | ECREIS |

| Bit Name | Description |
|---|---|
| ETMA[1:0] | Access control of ECTMCE bit.<br>10B: Write access enabled.<br>Other than above: Write access disabled. |
| ECTMCE | Enable the ECC test mode control.<br>0: Disable access to the test registers and test control bits.<br>1: Enable access to the test registers and test control bits. |
| ECTRRS | Select the test register read data.<br>0: When CFDECTEDH / CFDECTEDL is read, register value is read.<br>When CFDECERDB is read, register value is read.<br>1: When CFDECTEDH / CFDECTEDL is read, CAN-RAM output value is read.<br>When CFDECERDB is read, CAN-RAM ECC input value is read. |
| ECDCS | Select the ECC decoder input data. (lower 32-bit data)<br>0: Select a CAN-RAM output data as the decoder input.<br>1: Select the value set in CFDECTEDH / CFDECTEDL as the decoder input. |
| ECREIS | Select the ECC decoder input ECC redundant bits. (upper 7-bit data)<br>0: Select a CAN-RAM output data as decoder input.<br>1: Select the value set in CFDECERDB as decoder input. |

(3) **CAN-RAM ECC decoder input data replacement test register L (CFDECTEDL)**

This register is used to ECC function test. This register can be accessed by the 16-bit memory manipulation instruction.

Address: F07CCH After reset: 0000H R/W

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| CFDECTEDL | ECEDBL[15:8] | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ECEDBL[7:0] | | | | | | | |

| Bit Name | Description |
|---|---|
| ECEDBL[15:0] | Decoder input data for test mode.<br>0000H to FFFFH |

(4) **CAN-RAM ECC decoder input data replacement test register H (CFDECTEDH)**

This register is used to ECC function test. This register can be accessed by the 16-bit memory manipulation instruction.

Address: F07CEH After reset: 0000H R/W

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| CFDECTEDH | ECEDBH[15:8] | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ECEDBH[7:0] | | | | | | | |

| Bit Name | Description |
|---|---|
| ECEDBH[15:0] | Decoder input data for test mode.<br>0000H to FFFFH |

(5) **CAN-RAM ECC syndrome test register (CFDECSYND)**

This register is used to confirm the syndrome code generated in decode circuit when ECC test mode. This register can be read by an 8-bit memory manipulation instruction.

Address: F07CBH After reset: 00H R

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CFDECSYND | 0 | SYND[6:0] | | | | | | |

| Bit Name | Description |
|---|---|
| SYND[6:0] | ECC decode syndrome data for ECC test mode.<br>00H to 7FH |

(6) **CAN-RAM ECC redundant bit test register (CFDECHORD)**

This register holds the redundant bits output to the CAN-RAM when ECC test mode is enabled. This register can be read by 8-bit memory manipulation instruction.

Address: F07CAH After reset: 00H R

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CFDECHORD | 0 | HORD[6:0] | | | | | | |

| Bit Name | Description |
|---|---|
| HORD[6:0] | ECC redundant bits for ECC test mode. |

(7) **CAN-RAM ECC decoder input ECC bit replacement test register (CFDECERDB)**

This register holds the ECC redundant bit error injection data output to the CAN-RAM when ECC test mode is enabled. This register can be accessed by the 8-bit memory manipulation instruction.

Address: F07C8H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CFDECERDB | 0 | ERDB[6:0] | | | | | | |

| Bit Name | Description |
|---|---|
| ERDB[6:0] | ECC redundant bit error injection data for ECC test mode.<br>00H to 7FH |

(8) **CAN-RAM ECC error address register (CFDECEAD)**

This register used to confirm the address at which an ECC error has occurred. This register can be read by 8-bit memory manipulation instruction.

Address: F07D0H After reset: 0000H R

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| CFDECEAD | 0 | 0 | 0 | 0 | 0 | 0 | ECEAD[9:8] | |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ECEAD[7:0] | | | | | | | |

| Bit Name | Description |
|---|---|
| ECEAD[9:0] | ECC error address.<br>000H to 3FFH |

## 5.3 Flow Chart of CAN-RAM ECC Function

Figure 5-1 is a flow chart of the CAN-RAM ECC function.

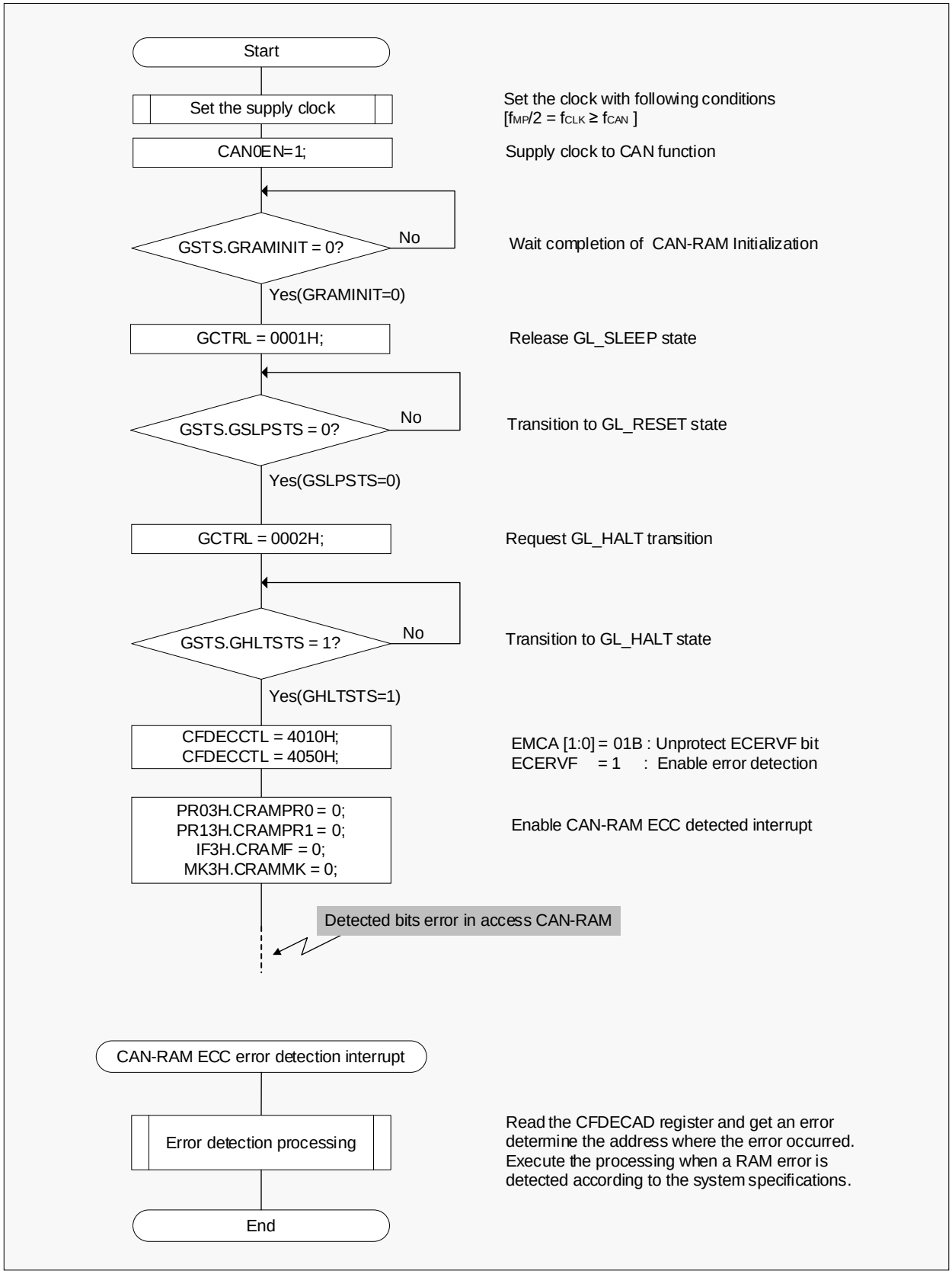



**Figure 5-1 Flow Chart of CAN-RAM ECC Function**

## 5.4 Cautions when Using CAN-RAM ECC Function

The following are the cautions when using CAN-RAM ECC function.

(1) It is recommended to clear the bit error flag (ECER1F, ECER2F) of the CFDECCTL register after initializing the CAN-RAM.

(2) If CAN-RAM is in the initial state, do not perform error judgment by ECC. Set the error judgment to disabled once in the CFDECCTL register, and allow the judgment after initializing the CAN-RAM.

(3) The CAN-RAM error address is stored to the CAN-RAM ECC error address register (CFDECEAD) upon detection of the first ECC error while no error status is set. However, only when 2-bit error is detected after 1-bit error detection, the address of second 2-bit error will be overwritten to this register regardless of the address is same or not. Only one address can be held in this register.

# 6. Code Flash Memory ECC Function

## 6.1 Overview of Code Flash Memory ECC Function

The Code Flash Memory ECC function has the following functions for 1-bit error correction and identification of code flash memory data.

(1) Single-bit error correction (SEC).

(2) Accumulated error detection (AED).

(3) Brand-new error detection (BED).

(4) Brand-new error address and syndrome code capturing.

(5) Error address overflow (OVF).

(6) Maskable interrupt for code flash memory ECC (INTROM).

## 6.2 Registers used for Code Flash Memory ECC Function

The registers used for the Code Flash Memory ECC function are described below.

(1) **Code flash bit error detection function control register (CFERRCTLR)**

This register is used to manage error detection for the code flash memory. This register can be accessed by the 8-bit memory manipulation instruction.

Address: F00B8H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CFERRCTLR | RES | AEDEN | BEDEN | OVFEN | DIAG | 0 | PAGE[1:0] | |

| Bit Name | Description |
|---|---|
| RES | Reset the code flash memory error detection function.<br>0: No reset request<br>1: Reset request for the code flash memory error detection function registers |
| AEDEN | Enable AED detection interrupt and AED status flag.<br>0: Set disable<br>1: Set enable |
| BEDEN | Enable BED detection interrupt and BED status flag.<br>0: Set disable<br>1: Set enable |
| OVFEN | Enable OVF detection interrupt and OVF status flag.<br>0: Set disable<br>1: Set enable |
| DIAG | Enable the self-diagnosis.<br>0: Set disable<br>1: Set enable Note |
| PAGE[1:0] | Select capture registers.<br>00B: Select ERRADRL, ERRADRH (including ERRDAT bits)<br>01B: Select ERRADR1L, ERRADR1H (including ERRDAT1 bits)<br>10B: Select ERRADR2L, ERRADR2H (including ERRDAT2 bits)<br>11B: Select ERRADR3L, ERRADR3H (including ERRDAT3 bits) |

**Note** Setting DIAG=1 during flash memory programming is prohibited.

(2) **Code flash bit error detection function status register (CFERRSTR)**

This register indicates error status for the code flash memory ECC. This register can be accessed by the 8-bit memory manipulation instruction.

Address: F00B9H After reset: 00H R/W

| Symbol | 7 Note | 6 | 5 | 4 | 3 Note | 2 Note | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CFERRSTR | 0 | AEDST | BEDST | OVFST | 0 | 0 | CAPST[1:0] | |

| Bit Name | Description |
|---|---|
| AEDST | AED detection status and clear bit.<br>0: When reading 0, no AED detected.<br>1: When reading 1, AED detected. And write to 1, AEDST is cleared. |
| BEDST | BED detection status and clear bit.<br>0: When reading 0, no BED detected.<br>1: When reading 1, BED detected. And write to 1, BEDST is cleared. |
| OVFST | OVF detection status and clear bit.<br>0: When reading 0, no OVF detected.<br>1: When reading 1, OVF detected. And write to 1, OVFST is cleared. |
| CAPST[1:0] | Capture register status.<br>00B: AED has not been detected.<br>01B: AED has been detected. (detected by the address of ERRADR1H/L)<br>10B: AED has been detected. (detected by the address of ERRADR2H/L)<br>11B: AED has been detected. (detected by the address of ERRADR3H/L) |

**Note** Bits 7, 3 and 2 are read only. The read value is fixed to 0. Writing to these bits are ignored.

(3) **Code flash bit error detection address register L (ERRADRL)**

This register holds the error address (bits 15 to 2) when ECC syndrome error is detected. This register can be accessed by the 16-bit manipulation instruction.

Address: F00BAH (CFERRCTLR.PAGE[1:0] = 00B) After reset: FFFCH
R (CFERRCTLR.DIAG=0), R/W (CFERRCTLR.DIAG=1)

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| ERRADRL | ERRADR[15:8] | | | | | | | |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ERRADR[7:2] | | | | | | 0 | 0 |

| Bit Name | Description |
|---|---|
| ERRADR[15:2] | Latest error detection address.<br>0000H to FFFCH |

(4) **Code flash bit error detection address register H (ERRADRH)**

This register holds the error address (bits 19 to 16) when ECC syndrome error is detected. This register can be accessed by the 16-bit manipulation instruction.

Address: F00BCH (CFERRCTLR.PAGE[1:0] = 00B) After reset: 3F0FH
R (CFERRCTLR.DIAG=0), R/W (CFERRCTLR.DIAG=1)

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| ERRADRH | 0 | 0 | ERRDAT[5:0] | | | | | |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | ERRADR[19:16] | | | |

| Bit Name | Description |
|---|---|
| ERRDAT[5:0] | Latest error detection syndrome code.<br>000000B to 111111B |
| ERRADR[19:16] | Latest error detection address.<br>0000B to 1111B |

(5) **Code flash bit error detection address register nL (ERRADRnL) (n = 1 to 3)**

This register hold error addresses (bits 15 to 2) when ECC syndrome error is detected. (Up to 3 addresses) This register can be accessed by the 16-bit manipulation.

Address: F00BAH (CFERRCTLR.PAGE[1:0] = 01B or 10B or 11B) After reset: FFFCH
R(CFERRCTLR.DIAG=0), R/W (CFERRCTLR.DIAG=1)

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| ERRADRnL | ERRADRn[15:8] | | | | | | | |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ERRADRn[7:2] | | | | | | 0 | 0 |

| Bit Name | Description |
|---|---|
| ERRADRn[15:2] | Error detection address n.<br>0000H to FFFCH |

(6) **Code flash bit error detection address register nH (ERRADRnH) (n = 1 to 3)**

This register hold error addresses (bits 19 to 16) when ECC syndrome error is detected. (Up to 3 addresses) This register can be accessed by the 16-bit manipulation.

Address: F00BCH (CFERRCTLR.PAGE[1:0] = 01B or 10B or 11B) After reset: 3F0FH
R(CFERRCTLR.DIAG=0), R/W (CFERRCTLR.DIAG=1)

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| ERRADRnH | 0 | 0 | ERRDATn[5:0] | | | | | |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | ERRADRn[19:16] | | | |

| Bit Name | Description |
|---|---|
| ERRDATn[5:0] | Error detection syndrome code n.<br>000000B to 111111B |
| ERRADRn[19:16] | Error detection address n.<br>0000B to 0011B |

(7) **Flash write buffer register L (FLWL)**

When the CFERRCTLR.DIAG=0, this register used to the flash write data register (bit 15 to 0) for the flash programming. When the CFERRCTLR.DIAG=1, this register used to the flash data register (bit 15 to 0) for the code flash memory ECC self-test. This register can be read and written by the 16-bit manipulation instruction.

Address: F02BCH After reset: 0000H R/W

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| FLWL | FLW[15:8] | | | | | | | |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | FLW[7:0] | | | | | | | |

| Bit Name | Description |
|---|---|
| FLW[15:0] | When CFERRCTLR.DIAG = 1, errors can be intentionally injected as ECC self-test data (bits 15 - 0).<br>0000H to FFFFH |

(8) **Flash write buffer register H (FLWH)**

When the CFERRCTLR.DIAG=0, this register used to the flash write data register (bit 31 to 16) for the flash programming. When the CFERRCTLR.DIAG=1, this register used to the flash data register (bit 31 to 16) for the code flash memory ECC self-test. This register can be read and written by the 16-bit manipulation instruction.

Address: F02BEH After reset: 0000H R/W

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| FLWH | FLW[31:24] | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| FLW[23:16] | | | | | | | |

| Bit Name | Description |
|---|---|
| FLW[31:16] | When CFERRCTLR.DIAG = 1, errors can be intentionally injected as ECC self-test data (bits 31 - 16).<br>0000H to FFFFH |

(9) **Flash ECC write buffer register (FLWE)**

When the CFERRCTLR.DIAG=0, this register used to the flash write data register (bit 31 to 16) for the flash programming. When the CFERRCTLR.DIAG=1, this register used to the flash data register (bit 31 to 16) for the code flash memory ECC self-test. This register can be read and written by the 8-bit manipulation instruction.

Address: FFFC6H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FLWE | 0 | 0 | FLWE[5:0] | | | | | |

| Bit Name | Description |
|---|---|
| FLWE[5:0] | When CFERRCTLR.DIAG = 1, errors can be intentionally injected as ECC code of ECC self-test data.<br>000000B to 111111B |

## 6.3 Flow Chart of Code Flash Memory ECC Function

Figure 6-1 is a flow chart of code flash memory ECC function.

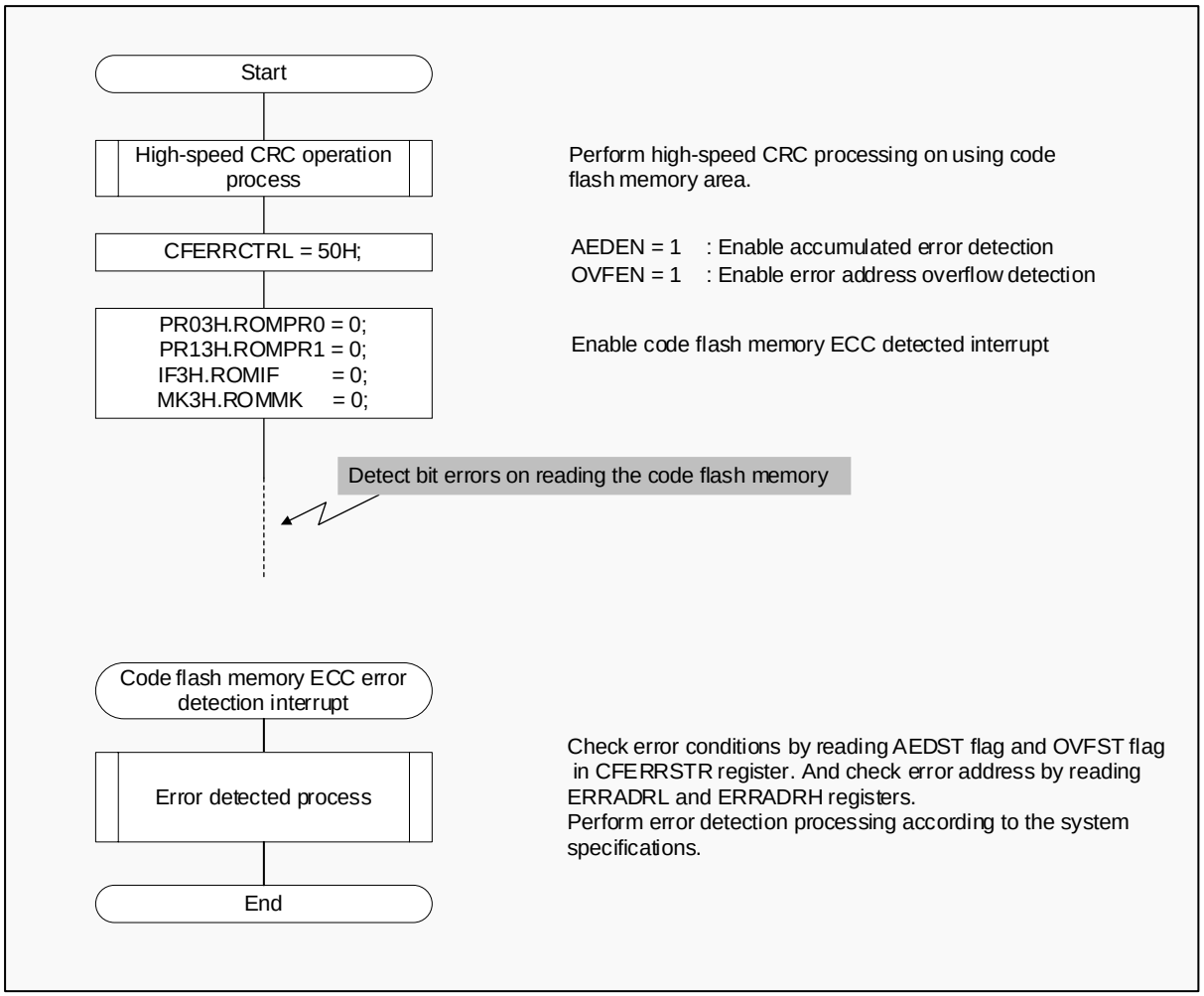



**Figure 6-1 Flow Chart of Code Flash Memory ECC Function**

## 6.4 Cautions when Using Code Flash Memory ECC Function

The following are the cautions when using code flash memory ECC function.

(1) If BED (Brand-new error detection) is enabled and 1-bit error is detected, the expected value is read due to bit correction.

(2) For effective error detection, perform a high-speed CRC test (or a read test by the CPU) on valid code flash memory area after reset release.

(3) When rewriting the code flash memory by self-programing, the code flash memory ECC circuit must be initialized after writing is completed. Initialize with a reset (external or internal).

# 7. CPU Stack Pointer Monitor Function

## 7.1 Overview of CPU Stack Pointer Monitor Function

This function monitors the address pointed to by a stack pointer to check that the address is within the stack area. When the stack pointer moves to an address beyond the area, the interrupt request (INTSPM) is generated.

## 7.2 Registers used for CPU Stack Pointer Monitor Function

The registers used for the CPU stack pointer monitor function are described below.

(1) **SPM control register (SPMCTRL)**

This register enables/disables the CPU stack pointer monitor function. This register can be accessed by an 8-bit memory manipulation instruction.

Address: F00D8H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SPMCTRL | SPMEN | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit Name | Description |
|---|---|
| SPMEN Note | 0: Disables the stack pointer monitor function.<br>1: Enables the stack pointer monitor function. Disables writing to the SPOFR and SPUFR registers. |

**Note** Writing 1 to the SPMEN bit is only valid and writing 0 after setting the SPMEN bit to 1 is invalid.

(2) **SP overflow address setting register (SPOFR)**

This register specifies a stack pointer overflow address (upper limit). This register can be accessed by a 16-bit memory manipulation instruction.

Address: F00DAH After reset: FFFEH R/W

| Symbol | 15 | | | | | | | 8 | 7 | | | | | | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPOFR | − | − | − | − | − | − | − | − | − | − | − | − | − | − | − | 0 |

| Bit Name | Description |
|---|---|
| 15-1 Note | Sets a stack pointer overflow address.<br>The lowest bit (0) is fixed to 0. When writing, write 0. |

**Note** When SPMEN = 1, writing to the SPOFR register is invalid.

(3) **SP underflow address setting register (SPUFR)**

This register specifies a CPU stack pointer underflow address (lower value). This register can be accessed by a 16-bit memory manipulation instruction.

Address: F00DCH After reset: 0000H R/W

| Symbol | 15 | | | | | | | 8 | 7 | | | | | | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPUFR | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | 0 |

| Bit Name | Description |
|---|---|
| 15-1 Note | Sets a stack pointer underflow address.<br>The lowest bit (0) is fixed to 0. When writing, write 0. |

**Note** When SPMEN = 1, writing to the SPUFR register is invalid.

## 7.3 Flow Chart of CPU Stack Pointer Monitor Function

Figure 7-1 is a flow chart of the CPU stack pointer monitor function.

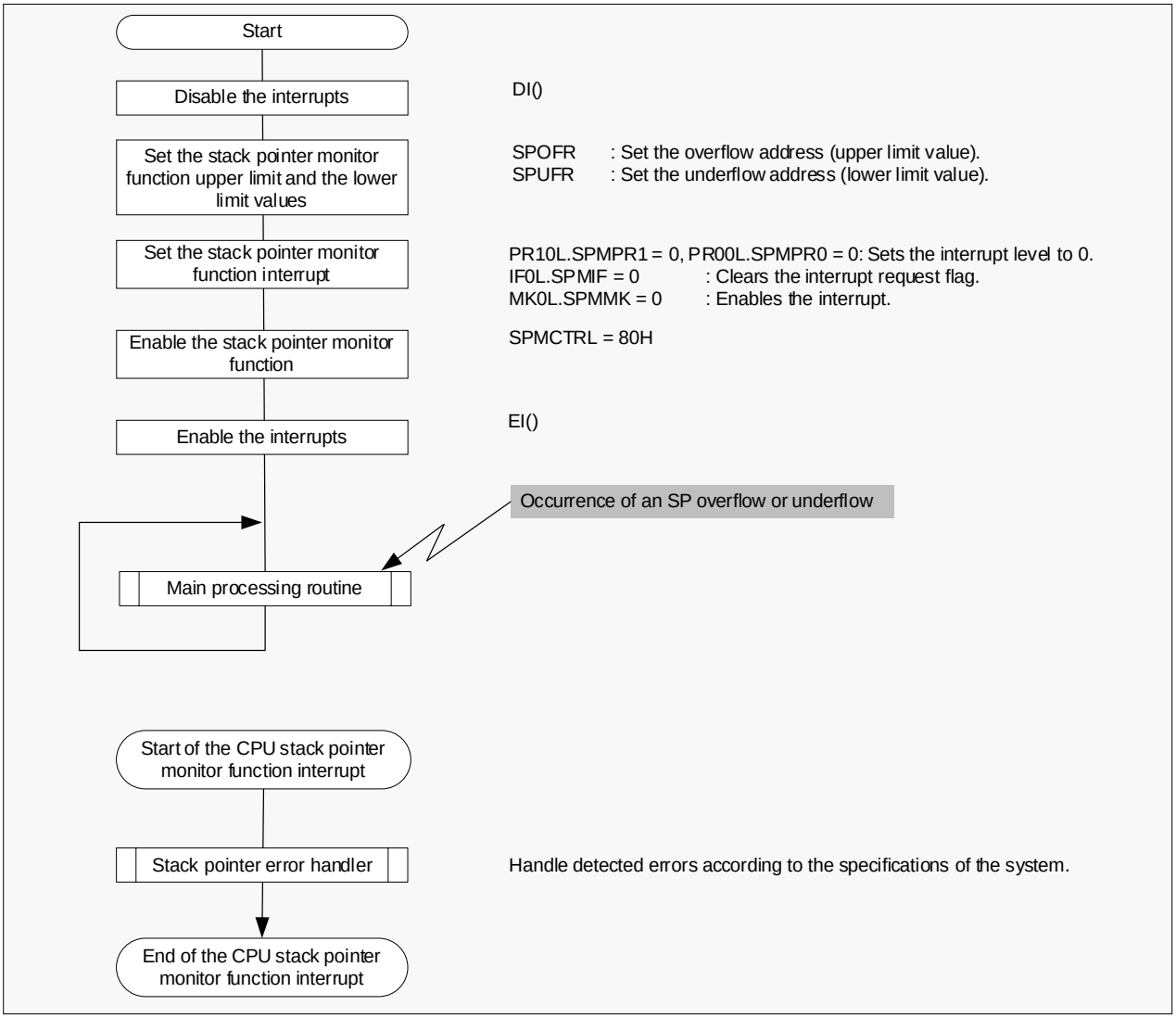



**Figure 7-1 Flow Chart of Stack Pointer Monitor Function**

## 7.4 Interrupt Determination of CPU Stack Pointer Monitor Function

The interrupt generated by the CPU stack pointer monitor function shares the same vector table address with the INTP4 interrupt. When the two interrupts are both used, the source of the interrupt needs to be determined by software.

Figure 7-2 is a flow chart of the CPU stack pointer monitor function.

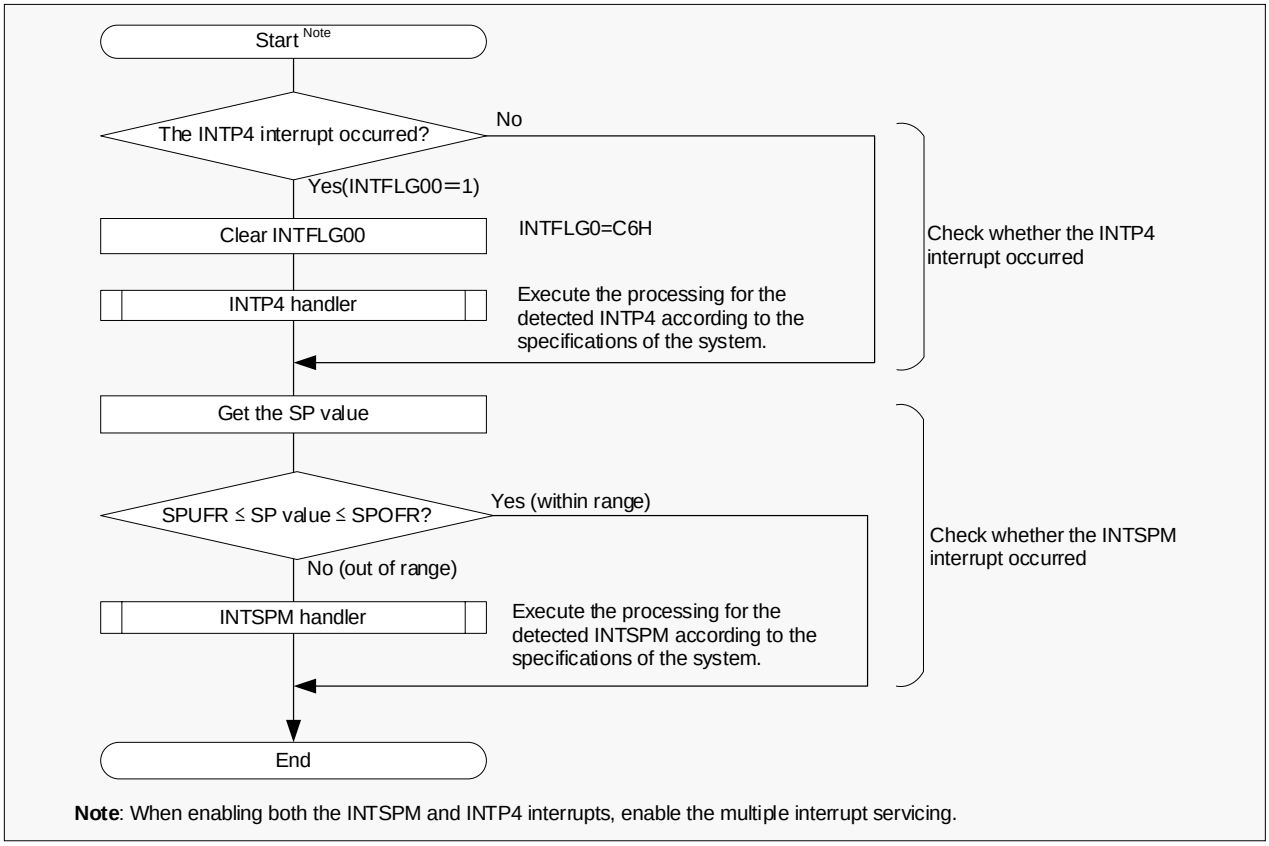



**Figure 7-2 Determination of INTSPM and INTP4 interrupts**

## 7.5 Cautions when Using CPU Stack Pointer Monitor Function

The following are the cautions when using the CPU stack pointer monitor function.

(1) If the value of the stack pointer remains out of ranges of the SPOFR and SPUFR registers after an SP overflow/underflow is detected, SP overflow/underflow will no longer be detected. To keep the monitor function enabled, set the stack pointer address value again to be within the monitorable range.

(2) The same vector table address is used by the INTSPM and INTP4 interrupts. To use both interrupt together, the source of the interrupt needs to be determined by SP overflow or underflow interrupt.

# 8. Clock Monitor Function

## 8.1 Overview of Clock Monitor Function

The clock monitor function monitors the statuses of the main system clock ($f_{MAIN}$) and main system /PLL select clock ($f_{MP}$) by using the low-speed on-chip oscillator clock ($f_{IL}$).

When this function detects that the oscillation of the main system clock ($f_{MAIN}$) has stopped, it generates a reset signal.

When this function detects that the oscillation of the PLL select clock ($f_{MP}$) has stopped, clock through mode is forcibly selected (PLLSTS.SELPLLS is set to 0), and the INTCLM interrupt request is generated. However, the value of the SELPLL bit in the PLLCTL register will not change from “1”.

If the sampling clock (low-speed on-chip oscillator clock) is stopped, the clock monitor will not operate.

## 8.2 Registers used for Clock Monitor Function

The registers used for the clock monitor function are described below.

(1) **User option byte (000C1H/040C1H)**

Setting for the clock monitor function.

Address: 000C1H/040C1H After reset: - (User setting value)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 000C1H/040C1H | VPOC[2:0] | | | CLKMB | LVIS[1:0] | | LVIMDS[1:0] | |

| Bit Name | Description |
|---|---|
| CLKMB | 0: Enables the clock monitor function.<br>1: Disables the clock monitor function. |

(2) **Clock monitor test register (CLMTES)**

This register enables/disables the clock monitor function.

Address: F02CCH After reset: 00H R/W

| Symbol | <7> | 6 | 5 | 4 | <3> | 2 | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| CLMTES | TESEN | 0 | 0 | 0 | CLMTEN | 0 | CK2MSK | CK1MSK |

| Bit Name | Description |
|---|---|
| TESEN | 0: Test function setting disabled.<br>1: Test function setting enabled. |
| CLMTEN | 0: Clock monitor test disabled.<br>1: Clock monitor test enabled. |
| CK2MSK | 0: Stop the monitoring clock ($f_{MP}$) at low level.<br>1: Does not stop the monitoring clock ($f_{MP}$). |
| CK1MSK | 0: Stop the monitoring clock ($f_{MAIN}$ or $f_{MP}$) at low level.<br>1: Does not stop the monitoring clock ($f_{MAIN}$ or $f_{MP}$). |

## 8.3 Flow Chart of Clock Monitor Function

Figure 8-1 is a flow chart of the clock monitor function.

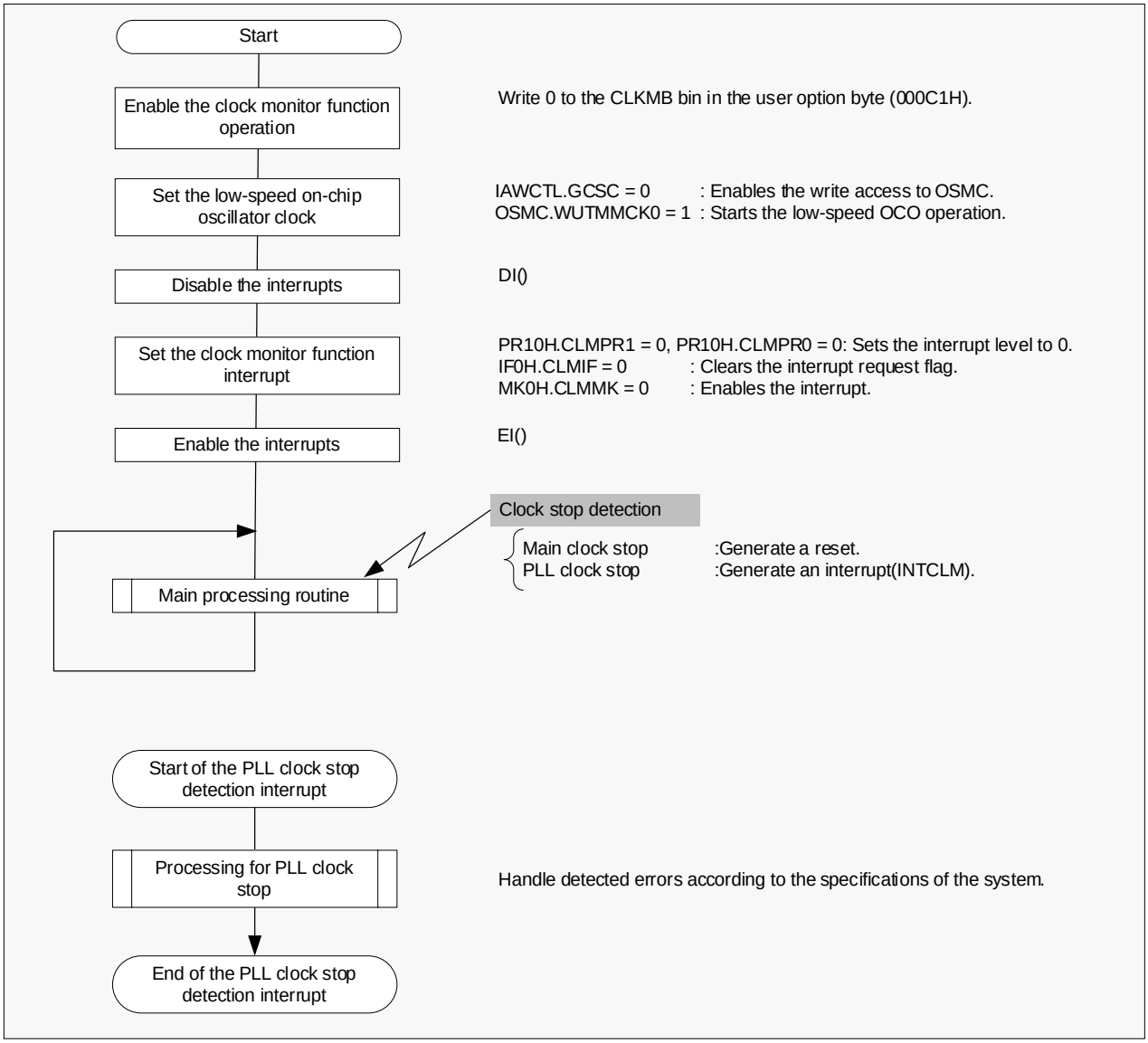



**Figure 8-1 Flow Chart of Clock Monitor Function**

## 8.4 Interrupt Determination of Clock Monitor Function

The PLL clock stop detection interrupt of the clock monitor function shares the same vector table address with the INTP13 interrupt. To use both interrupts together, the source of the interrupt needs to be determined by software.

Figure 8-2 is a flow chart of interrupt determination for when the PLL clock stop is detected.

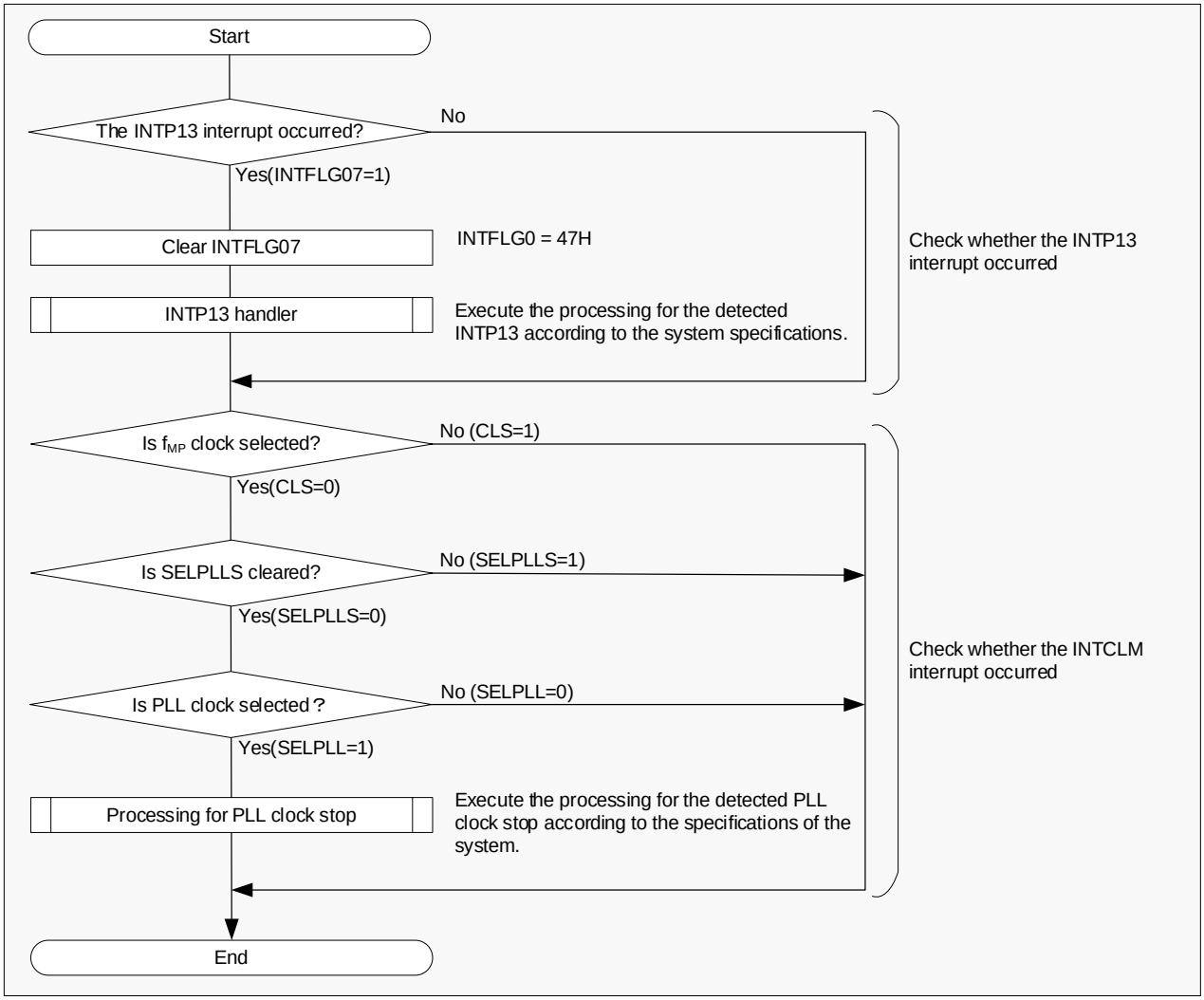



**Figure 8-2 Determination of PLL Clock Detection Interrupt and INTP13 Interrupt**

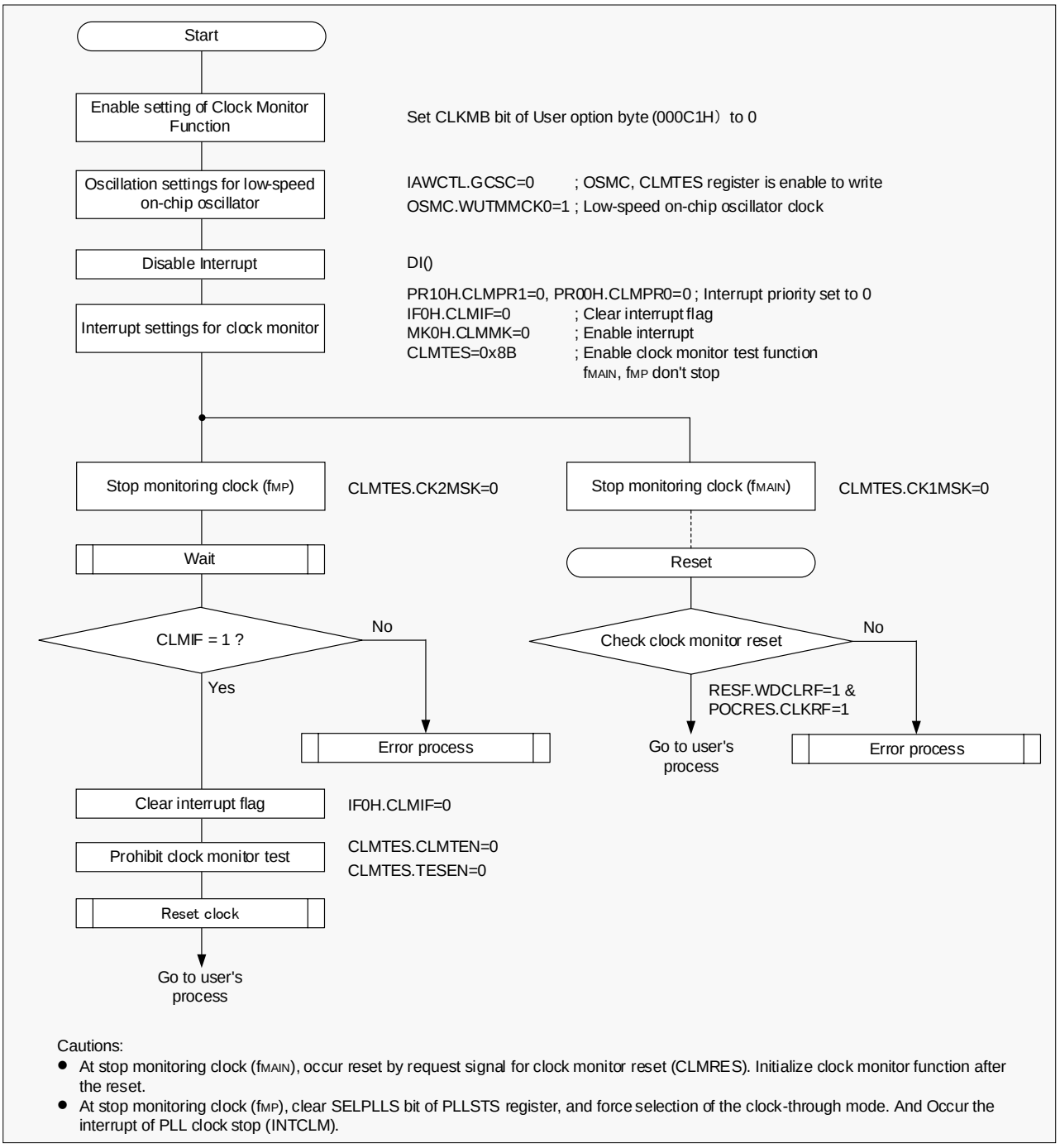
## 8.5 Flow Chart of Clock Monitor Function (Self-Test Mode)

Self-test mode of clock monitor function is a function to check whether the clock monitor is operating normally. This mode can stop the main system clock ($f_{MAIN}$), the main system / PLL select clock ($f_{MP}$) by software control.

Setting the CLMTES.CK1MSK to 0 by software control, main system clock ($f_{MAIN}$) can be input as a monitor clock with a fixed low level.

Setting the CLMTES.CK2MSK to 0 by software control, main system / PLL select clock ($f_{MP}$) can be input as a monitor clock with a fixed low level.

Figure 8-3 is flow chart of using self-test mode.




Cautions:

- At stop monitoring clock ($f_{MAIN}$), occur reset by request signal for clock monitor reset (CLMRES). Initialize clock monitor function after the reset.
- At stop monitoring clock ($f_{MP}$), clear SELPLLS bit of PLLSTS register, and force selection of the clock-through mode. And Occur the interrupt of PLL clock stop (INTCLM).

**Figure 8-3 Flow Chart of the Clock Monitor Function (Self-test mode)**

## 8.6 Cautions when Using Clock Monitor Function

The following are the cautions when using the clock monitor function.

(1) The clock monitor function is disabled (stopped) under the following conditions:

- The value of bit 4 (CLKMB) in the user option byte (000C1H) is 1.
- The sampling clock is stopped (the low-speed on-chip oscillator stops).
- In STOP/SNOOZE mode.
- While oscillation stabilization time is being counted after STOP mode is released.
- The CPU/peripheral hardware clock frequency ($f_{CLK}$) is equal to the subsystem clock ($f_{SUB}$) or the low-speed on-chip oscillator clock ($f_{IL}$).

(2) To transition the CPU to STOP mode by stopping the PLL while the clock monitor function is enabled, set the PLLCTL.PLLON bit to 0 (PLL stopped) prior to execution of the STOP instruction.

(3) The PLL clock stop detection interrupt of the clock monitor function shares the same vector table address with the INTP13 interrupt. To use both interrupts together, the source of the interrupt needs to be determined by software.

# 9. RAM Guard Function

## 9.1 Overview of RAM Guard Function

The RAM guard function protects data in a specified memory space. When the RAM guard function is enabled, writing to the protected space is invalid.

## 9.2 Registers used for RAM Guard Function

The registers used by the RAM guard function are described below.

(1) **Invalid memory access detection control register (IAWCTL)**

This register enables/disables the RAM guard function and specifies the space to be protected. This register can be accessed by an 8-bit memory manipulation instruction.

Address: F0078H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| IAWCTL | IAWEN | 0 | GRAM[1:0] | | 0 | GPORT | GINT | GCSC |

| Bit Name | Description |
|---|---|
| GRAM[1:0] | RAM guard space (protected area)<br>00B: Disabled<br>01B: 128 bytes starting from the RAM lowest address<br>10B: 256 bytes starting from the RAM lowest address<br>11B: 512 bytes starting from the RAM lowest address |

**Note** Set the RAM start address in the RAMSAR register before using the RAM guard function.

(2) **RAM Start Address Setting Register (RAMSAR)**

This register is used to set the start address of the RAM area to be used. Accessed with 8-bit memory manipulation instructions.

Address: FF076H After reset: EFH R/W

| Symbol | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|
| RAMSAR | RAMSAR[7:0] | | | | | | | |

| Bit Name | Description |
|---|---|
| RAMSAR[7:0] | RAM start address setting bits[Setting range: 9FH to FDH]<br>These bits set bit 15 to bit 8 of the RAM start address.<br>For example, when 9FH is set: RAM guard start address = F9F00H |

**Note** RAMSAR register can be written only once after reset release. Do not access the RAM area within 2 clocks after setting the RAMSAR register.

**Table 9-1 RAM Guard Function Setting Example**

| RAMSAR register | GRAM[1:0] Bit Value | Valid RAM Area | RAM Guard Space |
|---|---|---|---|
| 9FH | 01B (128 bytes) | F9F00H – FFEFFH (24KB) | F9F00H – F9F7FH |
| | 10B (256 bytes) | | F9F00H – F9FFFH |
| | 11B (512 bytes) | | F9F00H – FA0FFH |
| AFH | 01B (128 bytes) | FAF00H – FFEFFH (20KB) | FAF00H – FAF7FH |
| | 10B (256 bytes) | | FAF00H – FAFFFH |
| | 11B (512 bytes) | | FAF00H – FB0FFH |
| CFH | 01B (128 bytes) | FCF00H – FFEFFH (12KB) | FCF00H – FCF7FH |
| | 10B (256 bytes) | | FCF00H – FCFFFH |
| | 11B (512 bytes) | | FCF00H – FD0FFH |
| DFH | 01B (128 bytes) | FDF00H – FFEFFH (8KB) | FDF00H – FDF7FH |
| | 10B (256 bytes) | | FDF00H – FDFFFH |
| | 11B (512 bytes) | | FDF00H – FE0FFH |
| EFH | 01B (128 bytes) | FEF00H – FFEFFH (4KB) | FEF00H – FEF7FH |
| | 10B (256 bytes) | | FEF00H – FEFFFH |
| | 11B (512 bytes) | | FEF00H – FF0FFH |

## 9.3 Flow Chart of RAM Guard Function

Figure 9-1 is a flow chart of the RAM guard function.

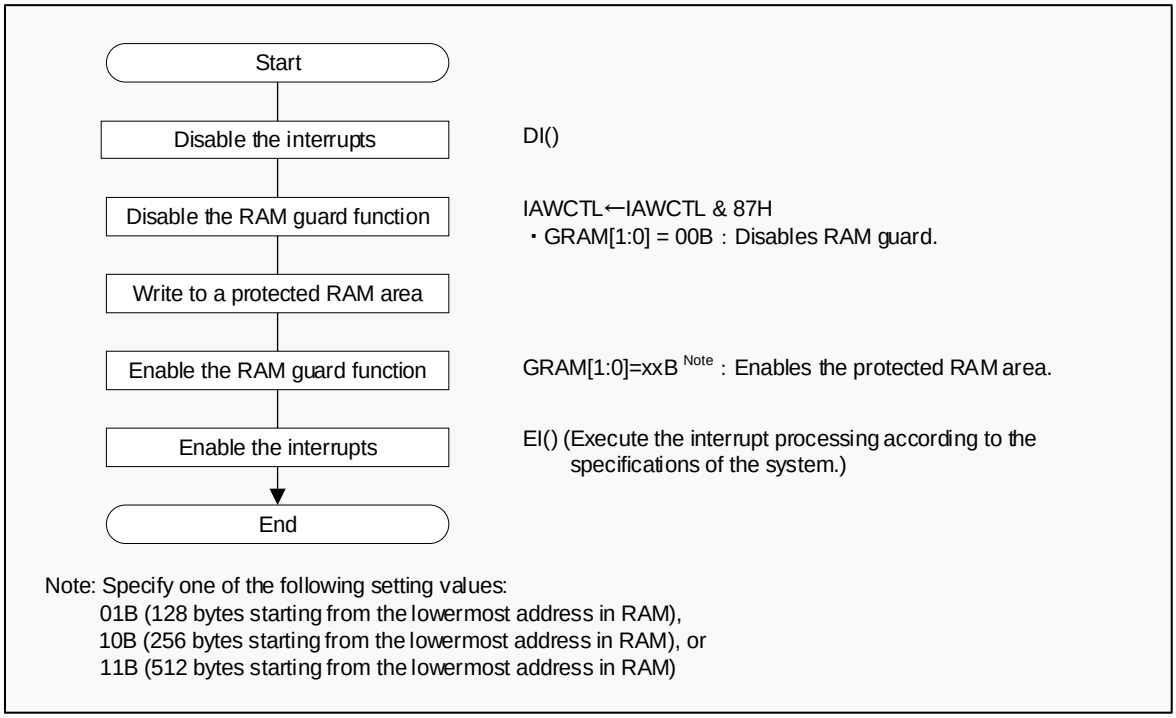



**Figure 9-1 Flow Chart of RAM Guard Function**

## 9.4 Cautions when Using the RAM Guard Function

The following are the cautions for when using the RAM guard function.

(1) When writing to a protected area in RAM by step execution during on-chip debugging, the RAM guard function is disabled.

(2) Do not specify stack area to an area where the RAM guard function is enabled.

(3) When using the RAM guard function, set the RAM start address with the RAMSAR register before setting the RAM guard space with the GRAM[1:0] bits of the IAWCTL register.

# 10. SFR Guard Function

## 10.1 Overview of SFR Guard Function

The SFR guard function protects data of the registers used for port/interrupt/clock control functions and the registers to control the voltage detection circuit. Once the SFR guard function is enabled, writing data to the SFR that is protected will be invalid. Table 10-1 lists the registers that the SFR guard function can protect.

**Table 10-1 SFR Guard-Target Registers**

| Function | Registers protected by SFR guard function Note 1 |
|---|---|
| Port function | PMxx, PUxx, PIMxx, POMxx, PMCxx, PITHLxx, PIORx |
| Interrupt function | IFxx, MKxx, PRxx, EGPx, EGNx |
| Clock control and voltage drop detection | CMC, CSC, OSTS, CKC, PERx, OSMC, LVIM, LVIS, CANCKSEL, LINCKSEL, CKSEL, PLLCTL, MDIV, RTCCL, POCRES, STPSTC, CLMTES, ADCKS Note 2 |

**Notes 1.** The target registers vary depending on products (according to ports or interrupt implemented).

**2.** Reading and writing of the ADCKS register is permitted when the ADCEN bit of the PER0 register is "1".

## 10.2 Registers Used for SFR Guard Function

The registers used for the SFR guard function are described below.

(1) **Invalid memory access detection control register (IAWCTL)**

This register is used to enable/disable the SFR guard function. This register can be accessed by an 8-bit memory manipulation instruction.

Address: F0078H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| IAWCTL | IAWEN | 0 | GRAM[1:0] | | 0 | GPORT | GINT | GCSC |

| Bit Name | Description |
|---|---|
| GPORT | Guard of registers for the port function<br>0: Disabled<br>1: Enabled |
| GINT | Guard of registers for the interrupt function<br>0: Disabled<br>1: Enabled |
| GCSC | Guard of registers for the clock control and the voltage detection circuit<br>0: Disabled<br>1: Enabled |

## 10.3 Flow Chart of SFR Guard Function

Figure 10-1 is a flow chart of the RAM guard function.

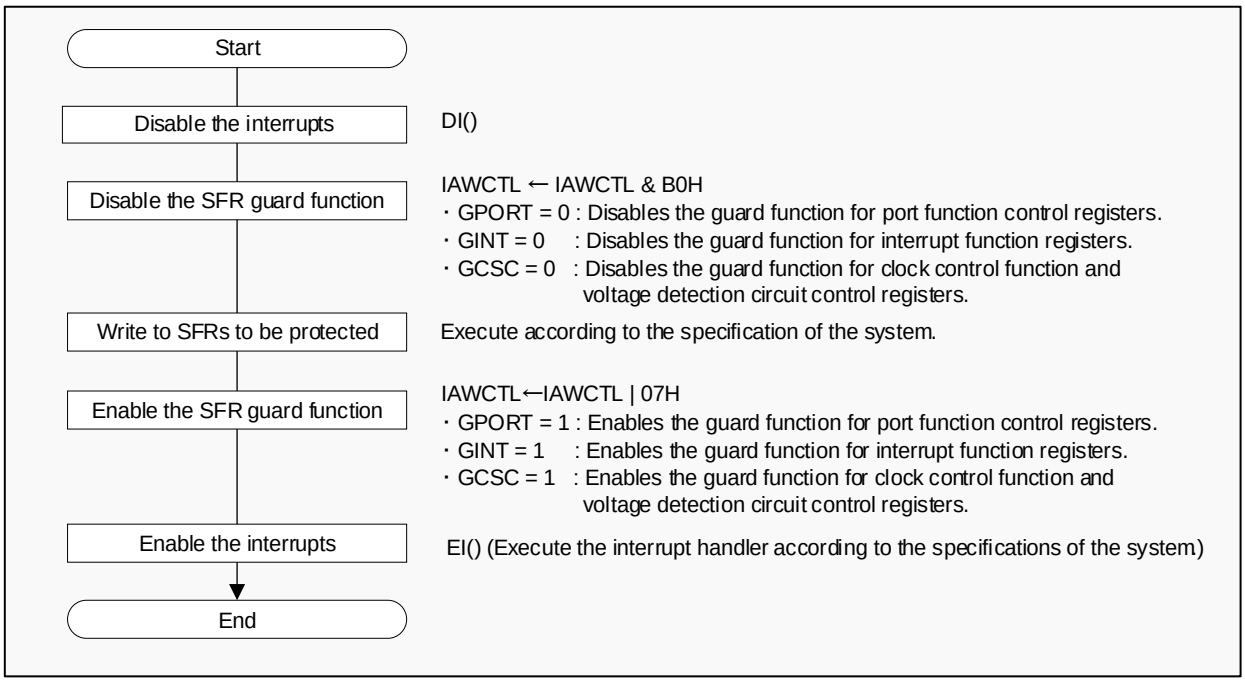



**Figure 10-1 Flow Chart of SFR Guard Function**

## 10.4 Cautions when Using SFR Guard Function

The following are the cautions when using the SFR guard function.

(1) This function will be disabled when reset is released (the values of the GPORT, GINT and GCSC bits are set to 0.)

(2) Port register (Pxx) is not guarded.

# 11. Invalid Memory Access Detection Function

## 11.1 Overview of Invalid Memory Access Detection Function

The invalid memory access detection function triggers a reset if an Invalid Access Detection space (see Figure 11-1) is accessed.

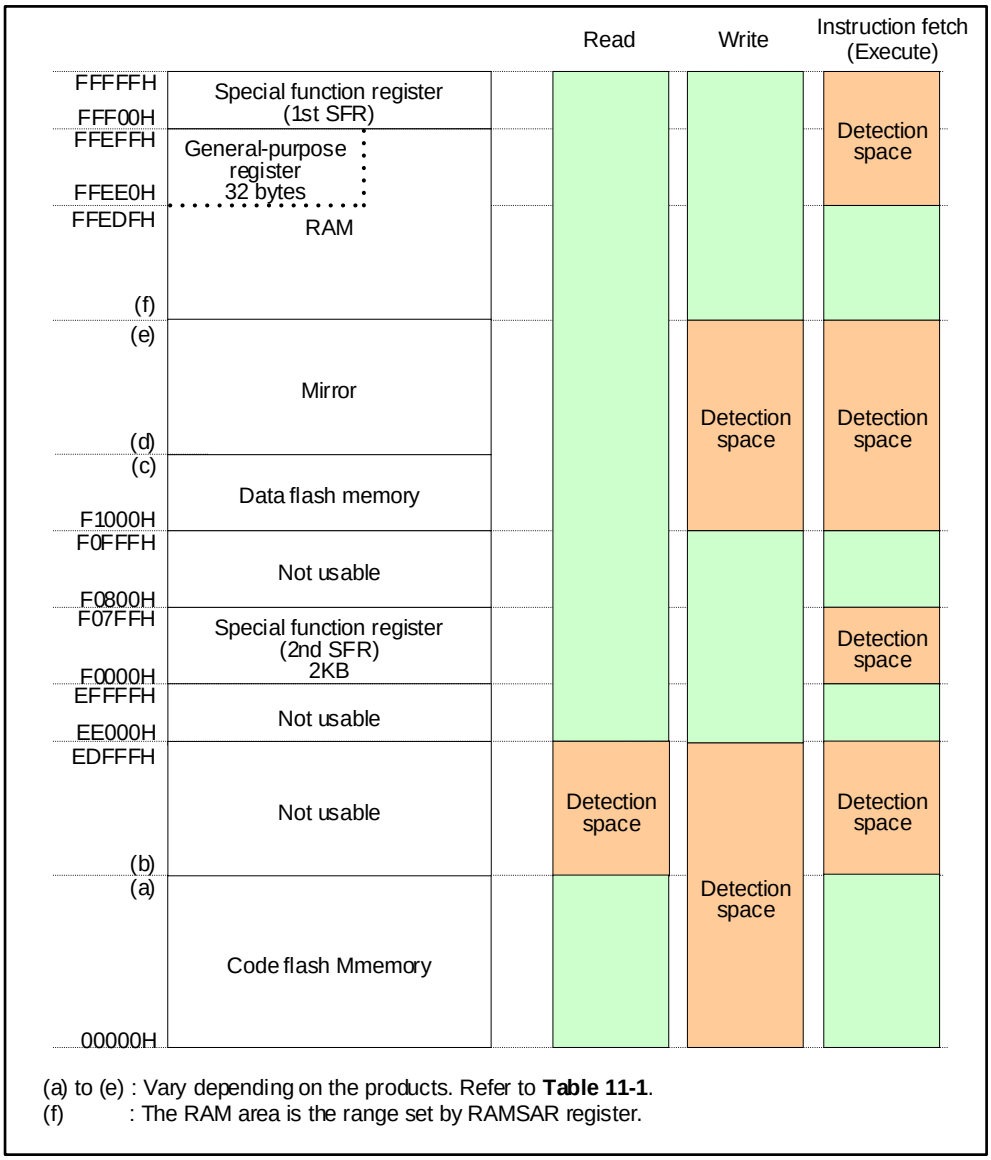



**Figure 11-1 Invalid Access Detection Area**

**Table 11-1 Memory Space by RL78 models**

| | Corresponding Address | RL78/F24 | RL78/F23 |
|---|---|---|---|
| (f) | RAM area start address (Determined by the setting of RAMSAR register) | F9F00H (RAMSAR=9FH) | FCF00H (RAMSAR=CFH) |
| (e) | Mirror area bottom address | F9EFFH | FCEFFH |
| (d) | Mirror area start address | F5000H | F3000H |
| (c) | Data flash memory area bottom address | F4FFFH | F2FFFH |
| (b) | Code flash memory area bottom address + 1 | 40000H | 20000H |
| (a) | Code flash memory area bottom address | 3FFFFH | 1FFFFH |

## 11.2 Register used for Invalid Memory Access Detection Function

The registers used for the invalid memory access detection function are described below.

(1) **Invalid memory access detection control register (IAWCTL)**

This register enables/disables detection of any invalid memory access. This register can be accessed by an 8-bit memory manipulation instruction.

Address: F0078H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| IAWCTL | IAWEN | 0 | GRAM[1:0] | | 0 | GPORT | GINT | GCSC |

| Bit Name | Description |
|---|---|
| IAWEN **Note** | 0: Disables detection of invalid memory access.<br>1: Enables detection of invalid memory access. |

**Note** Only writing 1 to the IAWEN bit is valid and writing 0 after setting the IAWEN bit to 1 is invalid.

## 11.3 Flow Chart of Invalid Memory Access Detection Function

Figure 11-2 is a flow chart of the invalid memory access detection function.

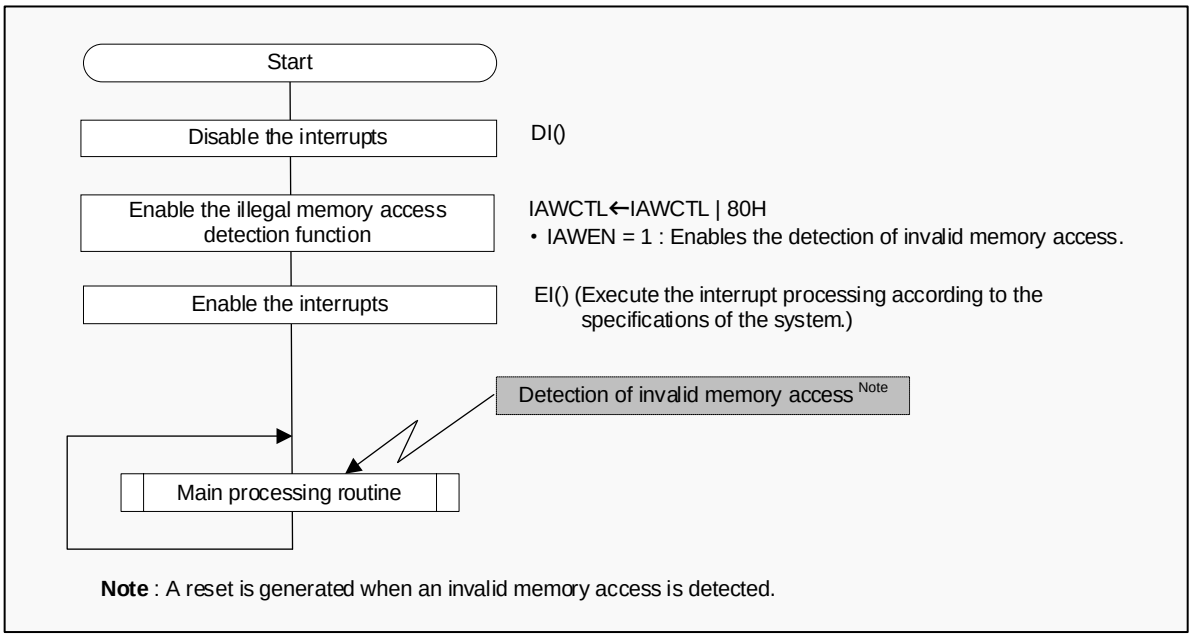



**Figure 11-2 Flow Chart of Invalid Memory Access Detection Function**

## 11.4 Cautions when Using Invalid Memory Access Function

The following are the cautions when using the invalid memory access detection function.

(1) When the value of bit 4 (WDTON) in the user option byte (000C0H) is set to 1, the invalid memory access detection function is enabled regardless of the setting value to the IAWEN bit.

# 12. Frequency Detection Function

## 12.1 Overview of Frequency Detection Function

When either of the high-speed on-chip oscillator clock ($f_{IH}$), external X1 oscillation clock ($f_{MX}$) or PLL clock ($f_{PLL}$) is selected for the CPU/peripheral hardware clock ($f_{CLK}$), the frequency detection function can detect any abnormality in the CPU/peripheral hardware clock ($f_{CLK}$) by comparing the selected clock with the low-speed on-chip oscillation clock ($f_{IL}$).

The number of clock cycles of the monitor clock (either one of $f_{IH}$, $f_{MX}$, or $f_{PLL}$) during a period of one clock cycle of the standard clock (low-speed on-chip oscillator clock) is counted using timer array unit 0 (TAU0). Based on the counted cycles, determine whether the frequency is correct or not using the user software.

## 12.2 Registers Used for Frequency Detection Function

The registers used for the frequency detection function are described below.

(1) **Timer input select register 0 (TIS0)**

This register selects the timer input used for channel 1 of timer array unit 0 (TAU0). To use the frequency detection function, select the low speed on chip oscillator clock as the input. This register can be accessed by an 8-bit memory manipulation instruction.

Address: F0074H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TIS0 | TIS07 | TIS06 | 0 | TIS04 | 0 | TIS0[2:0] | | |

| Bit Name | Description |
|---|---|
| TIS0[2:0] | Selects the timer input used for channel 1 of timer array unit 0.<br>000B, 010B, 011B: Input signal of timer input pin (TI01)<br>001B: Event input signal from ELC **Note**<br>100B: Low-speed on-chip oscillator clock ($f_{IL}$)<br>101B: Sub/low-speed on-chip oscillator select clock ($f_{SL}$)<br>Other than above: Setting prohibited |

**Note** Provided only in RL78/F24 products. Do not set in the RL78/F23 products.

(2) **Timer array unit 0-related registers**

- **Timer mode register 01 (TMR01)**

  CKS01[1:0] = Selects an operation clock (Select any one from CK00 to CK03 for the operation clock of channel 1.)

  CCS01= 0 (Sets 0 to “count clock selection”. (Selects the operation clock ($f_{MCK}$).)

  SPLIT01= 0 (Selects “16-bit timer operation”.)

  STS01[2:0] = 001B (The valid edge of the TI01 pin input is used as both the start trigger and capture trigger.)

  CIS01[1:0] = 00B (Selects falling edge detection for valid edge of TI01 pin input.)

  MD01[3:1] = 010B (Selects “capture mode” as the operating mode for channel 1.)

  MD010 = 0 (Selects “No generation of INTTM01 at count start”.)

- **Timer channel start register 0 (TS0)**

  TS01: Enables channel 1 operation.

- **Timer data register 01 (TDR01)**

  Captures the counter value with an input signal of the low-speed on-chip oscillator clock.

- **Timer status register 01 (TSR01)**

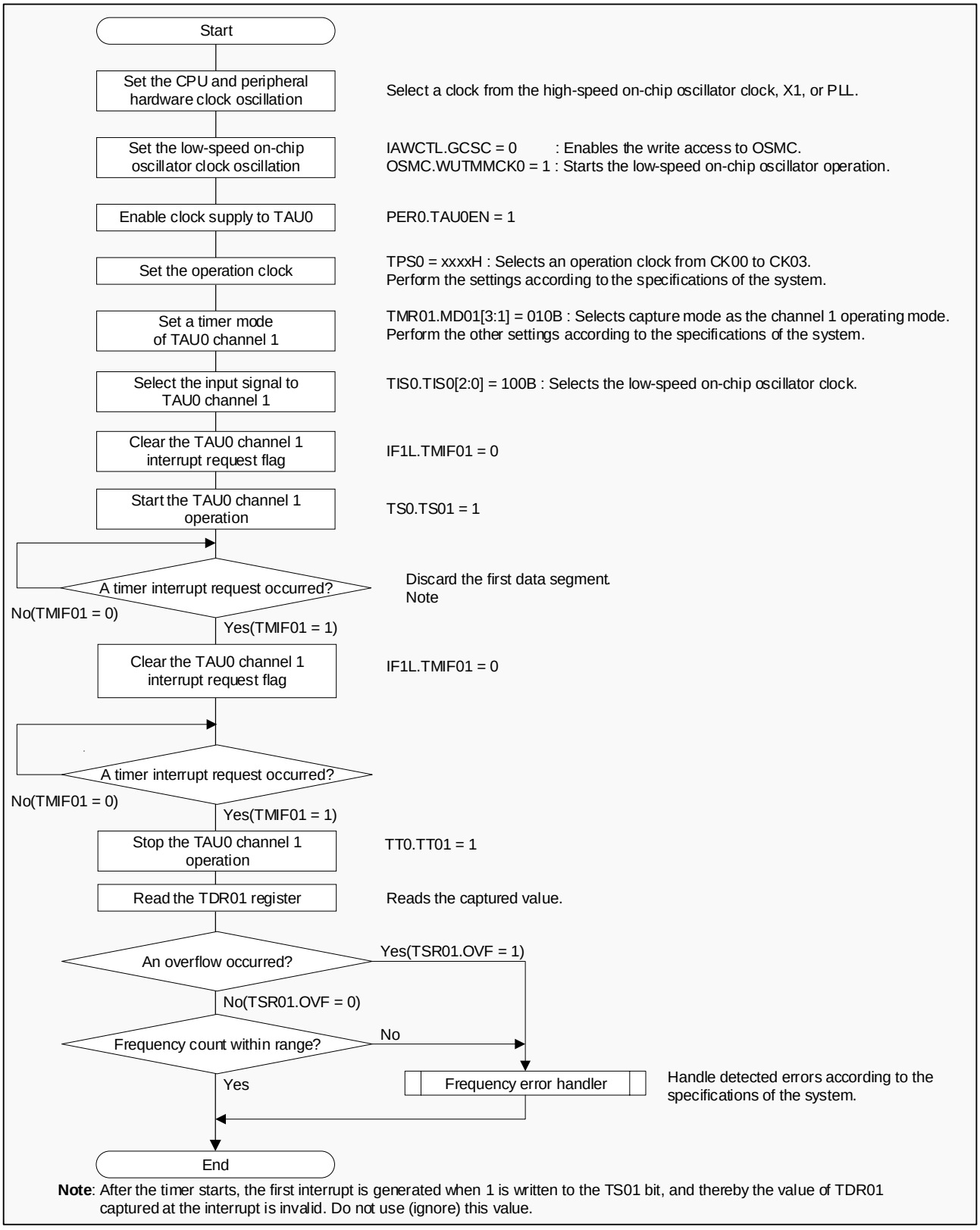
  OVF: Checks whether channel 1 overflowed or not.

- **Timer channel stop register 0 (TT0)**

  TT01: Stops the channel 1 operation.

## 12.3 Flow Chart of Frequency Detection Function

Figure 12-1 is a flow chart of the frequency detection function.




**Note**: After the timer starts, the first interrupt is generated when 1 is written to the TS01 bit, and thereby the value of TDR01 captured at the interrupt is invalid. Do not use (ignore) this value.

**Figure 12-1 Flow Chart of Frequency Detection Function**

# 13. A/D Test Function

## 13.1 Overview of A/D Test Function

This A/D test function supports the self-diagnosis by executing A/D conversions of internal voltage, and the disconnection detection assist function for the wire connected to analog input.

(1) Self-diagnosis of 12-bit A/D converter

This function confirms that the A/D converter is operating normally by A/D converting the internal voltage of the low-potential reference voltage, the high-potential reference voltage, and the high-potential reference voltage/2.

(2) Disconnection detection assist function

This function incorporates a pre-charge or dis-charge sampling capacitance before start A/D conversion. By using the function, User can use it to detect the disconnection of the wiring connected to the analog input. Figure 13-1 is overview of disconnection detection assist function. When using self-diagnosis, the disconnection detection assist function cannot be used.

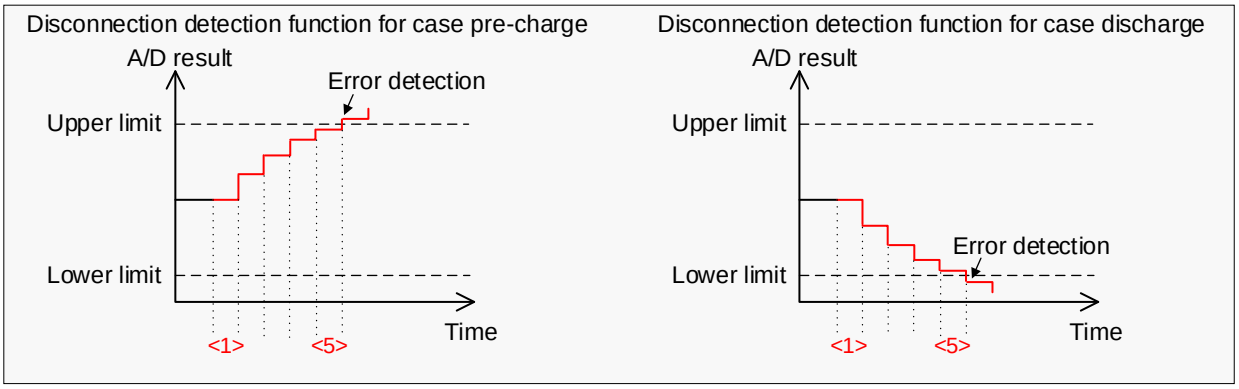



**Figure 13-1 Overview of Disconnection Detection Assist Function**

## 13.2 Registers used for A/D Test Function

The registers used for the A/D test function are described below.

(1) **A/D control expansion register (ADCER)**

This register sets self-diagnosis mode. This register can be accessed by a 16-bit memory manipulation instruction.

Address: F06BEH (ADWINR: 00H) After reset: 0000H R/W

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| ADCER | ADRFMT | 0 | 0 | 0 | DIAGM | DIAGLD | DIAGVAL[1:0] | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | ACE | 0 | 0 | 0 | 0 | 0 |

| Bit Name | Description |
|---|---|
| DIAGM | Setting of self-diagnosis of 12-bit A/D converter.<br>0: Disables the self-diagnosis.<br>1: Enables the self-diagnosis. |
| DIAGLD | Select of self-diagnosis mode.<br>0: Setting prohibited when self-diagnosis is enabled.<br>1: Fixed mode for self-diagnosis voltage. |
| DIAGVAL[1:0] | Setting of self-diagnosis conversion voltage.<br>00B: Setting prohibited in self-diagnosis voltage fixed mode.<br>01B: Uses the 0V.<br>10B: Uses the high-potential reference voltage × 1/2.<br>11B: Uses the high-potential reference voltage. |

(2) **A/D self-diagnosis data register (ADRD)**

This register can read the diagnosis results (A/D conversion results) based on the self-diagnosis of the A/D converter. This register can be accessed by a 16-bit memory manipulation instruction.

ADCER.ADRFMT = 0 (Right aligned)

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADRD | 0 | 0 | 0 | 0 | AD[11:0] | | | | | | | | | | | |

ADCER.ADRFMT = 1 (Left aligned)

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADRD | AD[11:0] | | | | | | | | | | | | 0 | 0 | 0 | 0 |

| Bit Name | Description |
|---|---|
| AD[11:0] | The A/D-converted value. |

(3) **A/D disconnection detection control register (ADDISCR)**

This register controls the disconnection detection channel of the analog voltage to be A/D converted. This register can be accessed by a 1-bit memory manipulation instruction or an 8-bit memory manipulation instruction.

Address: F06BAH (ADWINR: 07H) After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ADDISCR | 0 | 0 | 0 | ADNDIS[4:0] | | | | |

| Bit Name | Description |
|---|---|
| ADNDIS[4] | Selects the A/D disconnection detection assist function.<br>0: Select dis-charge.<br>1: Select pre-charge. |
| ADNDIS[3:0] | Setting the period of pre-charge or discharge.<br>0000B: Disables the disconnection detection assist function.<br>0001B: Setting prohibited.<br>0010B: 2 cycles, 0011B: 3 cycles, 0100B: 4 cycles, 0101B: 5 cycles,<br>0110B: 6 cycles, 0111B: 7 cycles, 1000B: 8 cycles, 1001B: 9 cycles,<br>1010B: 10 cycles, 1011B: 11 cycles, 1100B: 12 cycles, 1101B: 13 cycles,<br>1110B: 14 cycles, 1111B: 15 cycles |

## 13.3 Flow Chart of A/D Test Function (Self-test function)

Figure 13-2 is the flow chart of the A/D test function (Self-test function).

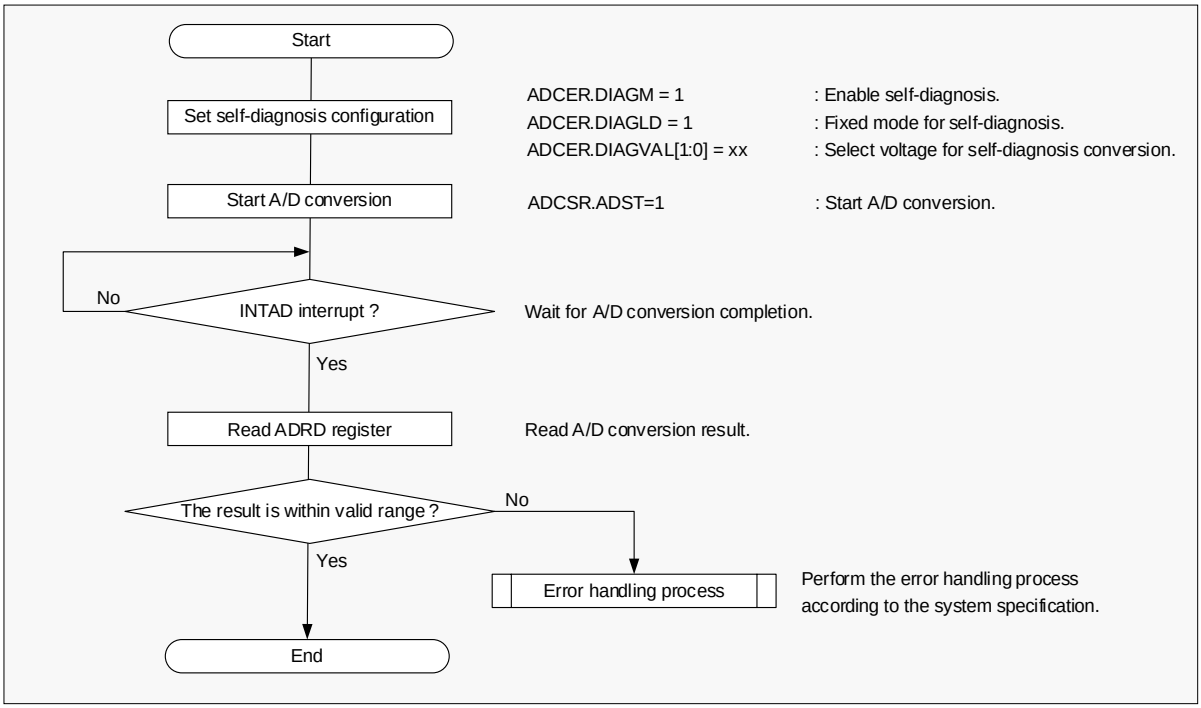



**Figure 13-2 Flow Chart of A/D Test Function (Self-test function)**

## 13.4 Flow Chart of A/D Test Function (Disconnection detection function)

Figure 13-3 is the flow chart of the A/D test function (Disconnection detection function).

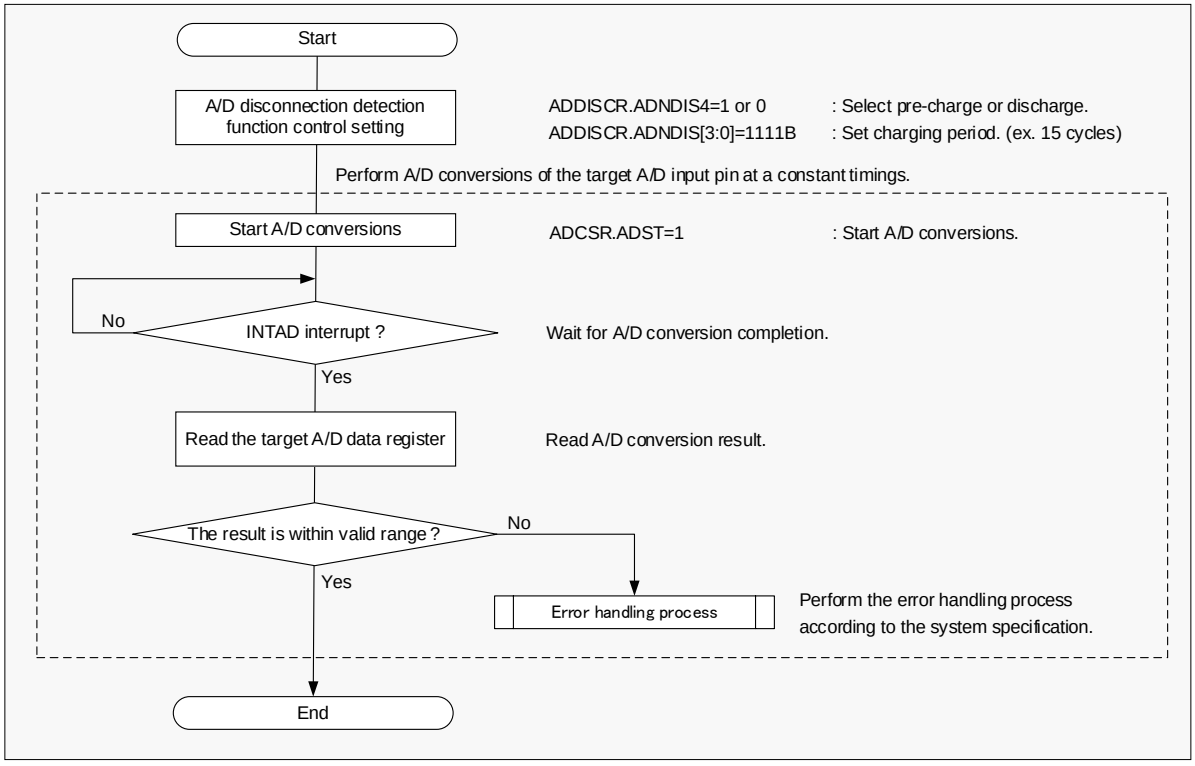



**Figure 13-3 Flow Chart of A/D Test Function (Disconnection detection function)**

## 13.5 Cautions when Using A/D Test Function

The following are the cautions for when using the A/D test function.

(1) Using disconnection detection assist function may lead to an error in absolute accuracy of the A/D converter, for an error voltage is input to the analog due to the resistive voltage division.

(2) The A/D conversion results must be checked with consideration of the accuracy of A/D conversions, or noise from the power supply by using several samples of evaluation data or values having ample margins.

# 14. Digital Output Signal Level Detection Function for I/O Ports

## 14.1 Overview of Digital Output Signal Level Detection Function for I/O Ports

This function can read the output level of the pin when the I/O ports are in output mode and check whether the output level is correct or not.

## 14.2 Registers used for Digital Output Signal Level Detection Function for I/O Ports

The registers used for the digital output signal level detection function for I/O ports are described below.

(1) **Port mode select register (PMS)**

This register specifies whether the output latch value of a port or the output level of a pin is to be read when the port is in output mode. This register can be accessed by a 1-bit memory manipulation instruction or an 8-bit memory manipulation instruction.

Address: F0077H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| PMS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PMS0 |

| Bit Name | Description |
|---|---|
| PMS0 | Selects the data to be read when the port is in output mode (PMmn=0).<br>0: Reads the value (output latch) of Pmn register.<br>1: Reads the output level of a pin. |

**Remark** m = 0 to 15, n = 0 to 7

## 14.3 Flow Chart of Digital Output Signal Level Detection for I/O Ports

Figure 14-1 is a flow chart of the digital output signal level detection for I/O ports.

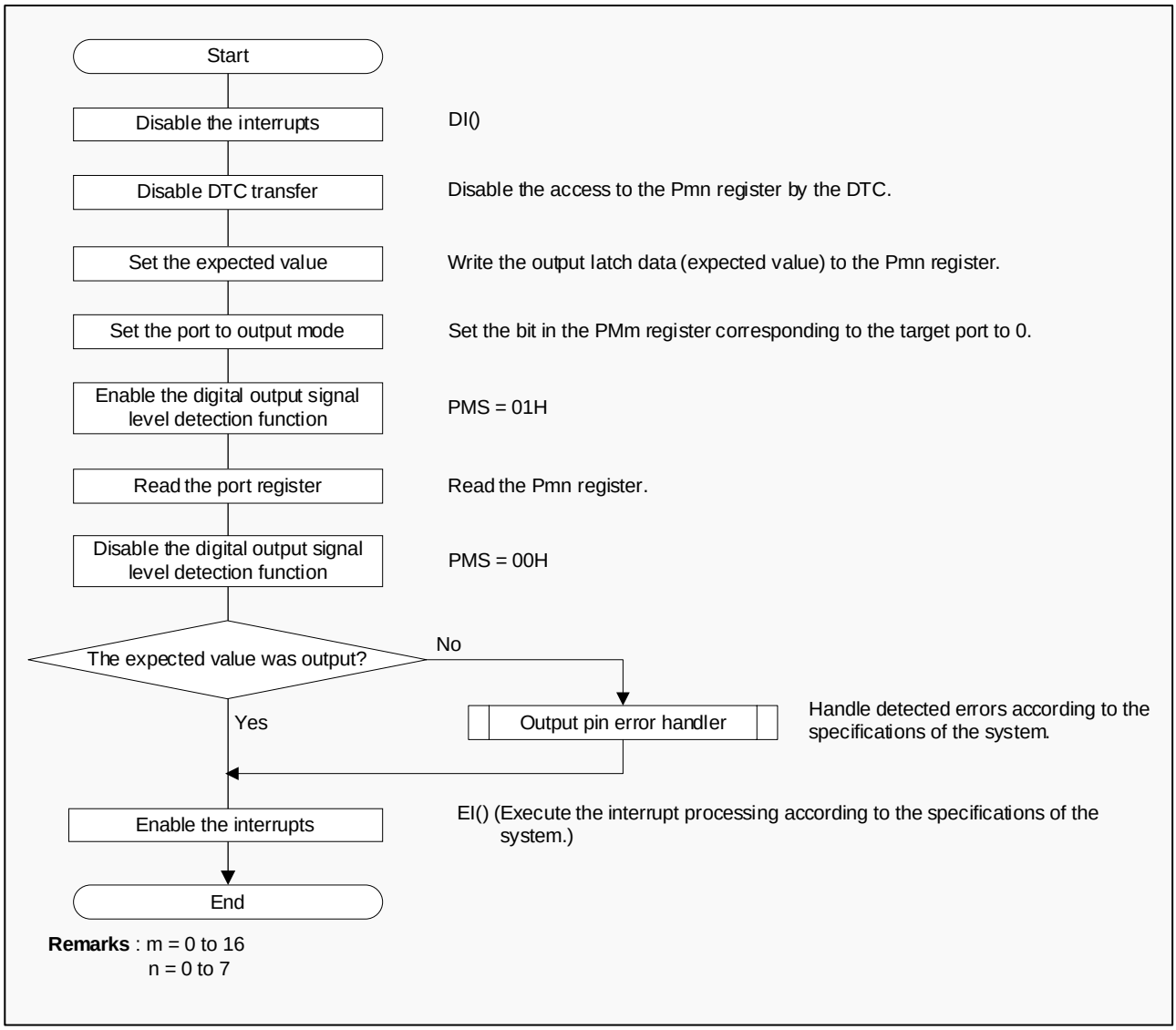



**Figure 14-1 Flow Chart of Digital Output Signal Level Detection for I/O ports**

## 14.4 Cautions when Using Digital Output Signal Level Detection for I/O Ports

The following are the cautions when using the digital output signal level detection for I/O ports.

(1) When any writing is executed for a port register (Pmn) with a bit manipulation instruction or an AND, OR instruction while the PMS0 bit in the PMS register is set to 1 (Output level of the pin is read), the output levels read are stored in the bits (other bits in the same port register). To write to the port register when the PMS0 bit is set to 1, an 8-bit data transfer instruction must be used. Also, the port register must be read while the DI (interrupt disabled) has been set.

# 15. Watchdog Timer Function

## 15.1 Overview of Watchdog Timer function

Watchdog timer function is used to detect an inadvertent program loop. If a program loop is detected, an internal reset signal is generated.

## 15.2 Registers used for Watchdog timer function

The following register is used to control the watchdog timer.

(1) **User option byte (000C0H/040C0H)**

Set the operation of watchdog timer.

Address: 000C0H/040C0H After reset: - (User setting value)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 000C0H/040C0H | WDTINT | WINDOW[1:0] | | WDTON | WDCS[2:0] | | | WDSTBYON |

| Bit Name | Description |
|---|---|
| WDTINT | Use of interval interrupt of watchdog timer.<br>0: Interval interrupt is not used.<br>1: Interval interrupt is generated when 75% of the overflow time + 1/2 $f_{WDT}$ is reached. |
| WINDOW[1:0] | Watchdog timer window open period.<br>00B: Setting prohibited<br>01B: 50%<br>10B: 75%<br>11B: 100% |
| WDTON | Operation control of watchdog timer counter.<br>0: Counter operation disabled. (counting stopped after reset)<br>1: Counter operation enabled. (counting started after reset) |
| WDCS[2:0] | Watchdog timer overflow time. ($f_{WDT}$ = 17.25kHz (MAX.))<br>000B: $2^6$ / $f_{WDT}$ ( 3.71 ms)<br>001B: $2^7$ / $f_{WDT}$ ( 7.42 ms)<br>010B: $2^8$ / $f_{WDT}$ ( 14.84 ms)<br>011B: $2^9$ / $f_{WDT}$ ( 29.68 ms)<br>100B: $2^{11}$ / $f_{WDT}$ ( 118.72 ms)<br>101B: $2^{13}$ / $f_{WDT}$ ( 474.89 ms)<br>110B: $2^{14}$ / $f_{WDT}$ ( 949.79 ms)<br>111B: $2^{16}$ / $f_{WDT}$ ( 3799.18 ms) |
| WDSTBYON | Operation control of watchdog timer counter (HALT/ STOP/ SNOOZE mode)<br>0: Counter operation stopped in HALT/ STOP/ SNOOZE mode.<br>1: Counter operation enabled in HALT/ STOP/ SNOOZE mode. |

(2) **Watchdog timer enable register (WDTE)**

This register clears the watchdog timer counter and restarts count again. This register can be accessed by an 8-bit memory manipulation instruction. When the WDTON bit in the user option byte is 1, writing ACH to this register, clears the count and starts counting.

Address: FFFABH After reset: 1AH/ 9AH Note R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| WDTE | | | | | | | | |

| Bit Name | Description |
|---|---|
| 7-0 | • Writing ACH during the window open period clears the watchdog timer counter. Note that the first write after reset release is not related to the window open period.<br>• If WDTON is “1” and written during the window close period, an internal reset will occur.<br>• If WDTON is “1” and a value other than ACH is written, or if it is written using a 1-bit memory manipulation instruction, an internal reset will occur. |

**Note** The reset value of the WDTE register varies depending on the setting value of the WDTON bit in the option byte (000C0H).
・When WDTON = 1: 1AH
・When WDTON = 0: 9AH

## 15.3 Flow Chart of Watchdog Timer function

Figure 15-1 is a flow chart of using the watchdog timer.

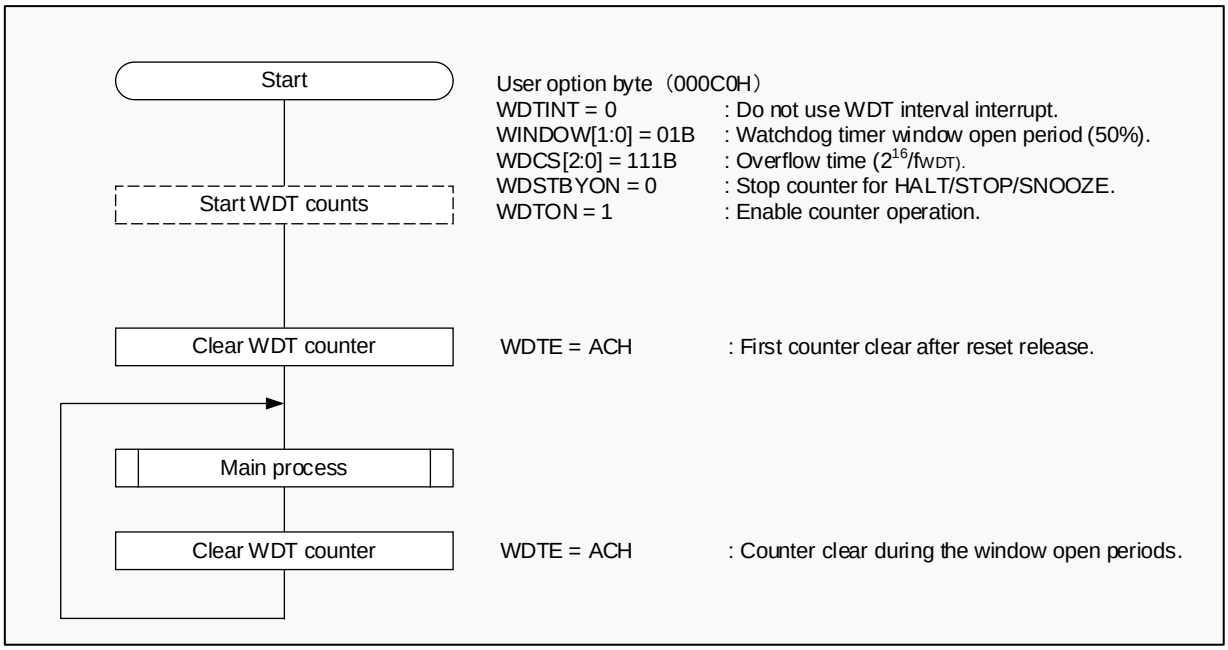



Figure 15-1 Flow Chart of Watchdog Timer Function

## 15.4 Cautions when using Watchdog Timer function

The following show the cautions for when using the watchdog timer function.

(1) When the window open period of watchdog timer is set to 75%, the processing to avoid the window close period and clear the counter. Table 15-1 shows the watchdog timer counter clear prohibition period when the window open period is set to 75%, and Figure 15-2 shows a processing example that the processing to avoid the window close and clear the counter in standby mode (intermittent operation).

**Table 15-1 WDT Counter Clear Prohibition Period**

| WDCS2 | WDCS1 | WDCS0 | WDT Overflow Time (In case of fWDT=17.25kHz (MAX.)) | WDT Counter Clear Prohibition Period |
|---|---|---|---|---|
| 0 | 0 | 0 | $2^6$ / $f_{WDT}$ ( 3.71ms) | 1.85ms to 2.51ms |
| 0 | 0 | 1 | $2^7$ / $f_{WDT}$ ( 7.42ms) | 3.71ms to 5.02ms |
| 0 | 1 | 0 | $2^8$ / $f_{WDT}$ ( 14.84ms) | 7.42ms to 10.04ms |
| 0 | 1 | 1 | $2^9$ / $f_{WDT}$ ( 29.68ms) | 14.84ms to 20.08ms |
| 1 | 0 | 0 | $2^{11}$ / $f_{WDT}$ ( 118.72ms) | 56.36ms to 80.32ms |
| 1 | 0 | 1 | $2^{13}$ / $f_{WDT}$ ( 474.89ms) | 237.44ms to 321.26ms |
| 1 | 1 | 0 | $2^{14}$ / $f_{WDT}$ ( 949.79ms) | 474.89ms to 642.51ms |
| 1 | 1 | 1 | $2^{16}$ / $f_{WDT}$ ( 3799.18ms) | 1899.59ms to 2570.04ms |

## 15.5 Processing Example when the window open period is set to 75%

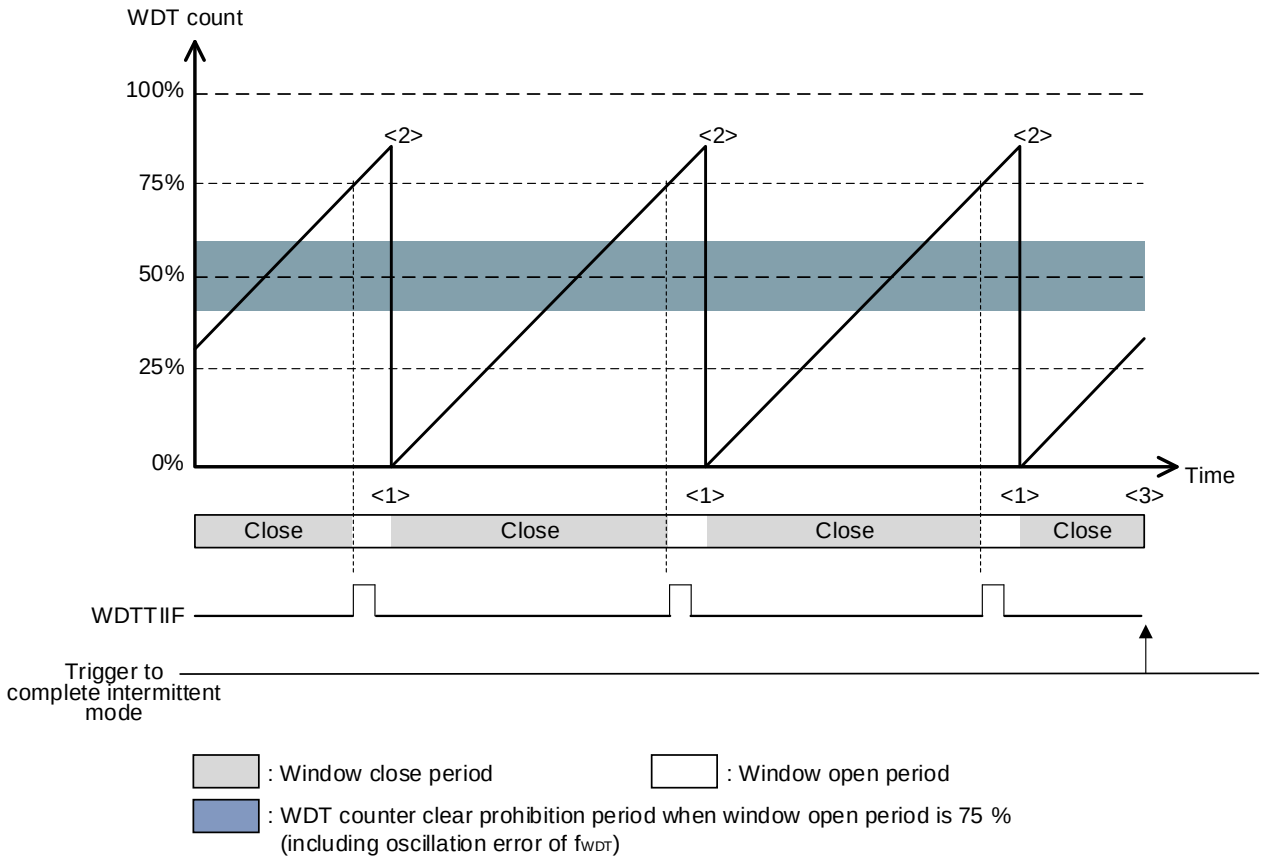
Show an example of processing when the window open period is set to 75%, the processing to avoid the WDT counter clear prohibition period and clear the WDT counter in standby mode (intermittent operation).

Figure 15-2 shows a processing example that returning from standby mode to normal mode by the WDT interval interrupt and clear the WDT counter (write “ACH” to the WDTE register).

Operating conditions: User option byte (000C0H) = 1101xxx1B

- WDTINT = 1 : Interval interrupt is generated when 75% + ½ $f_{WDT}$ of the overflow time is reached.
- WINDOW[1:0] = 10B : Window open period is 75%)
- WDTON = 1 : Counter operation enabled. (counting started after reset)
- WDCS[2:0] = xxxB : Overflow time is any setting.)
- WDSTBYON = 1 : Counter operation enabled in HALT/ STOP/ SNOOZE mode.

Set the WDT interval interrupt mask flag (WDTIMK) to “0” (interrupt enabled).




| | Software operation | Hardware condition |
|---|---|---|
| <1> | - | Occurs WDT interval interrupt |
| <2> | Write “ACH” to WDTE register | Clearing WDT counter |
| <3> | To normal operation mode process | Return from standby mode to normal operation mode |

**Figure 15-2 Processing Example When Using the WDT Function**

## 16. References

Documents referenced in this application note are shown below. When referring to these documents, make sure to obtain the latest version of each document from Renesas Electronics website.

- RL78/ F23, F24 User's Manual: Hardware Rev. 1.00
- RL78 family User's Manual: Software Rev. 2.30

## Revision History

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | 2022. 9.30 | - | First edition issued. |
| 1.10 | 2023.12.31 | P.19 | Corrected “4.5 Cautions when Using The Internal RAM ECC Function”. |
| | | P.25 | Revised “5.4 Cautions when Using CAN-RAM ECC Function”. |
| | | P.42 | Revised explanation of IAWCTL register, and added explanation of RAMSAR register at “9.2 Registers used for RAM Guard Function”. |
| | | P.44 | Revised “9.4 Cautions when Using the RAM Guard Function”. |
| | | P.45 | Revised “10.1 Overview of SFR Guard Function”. |
| | | P.52 | Added “Figure 13-1 Overview of Disconnection Detection Assist Function”. |
| | | P.55 | Revised “Figure 13-2 Flow Chart of A/D Test Function (Self-test function)”. |
| | | P.55 | Revised “Figure 13-3 Flow Chart of A/D Test Function (Disconnection detection function)”. |
| | | P.58 | Corrected “Figure 14-1 Flow Chart of Digital Output Signal Level Detection for I/O ports”. |
| | | P.60 | Revised “Figure 15-1 Flow Chart of Watchdog Timer Function”. |
| | | P.62 | Corrected “Figure 15-2 Processing Example When Using the WDT Function”. |
| | | | |

# General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

# Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, 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.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 November 2020)

## Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

## Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

## Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.