RL78/F13, F14, F15

Safety Function Application Note

R01AN2164EJ0240
Rev.2.40
Feb.15.2022

Introduction

This application note describes the safety functions implemented on the RL78/F13, RL78/F14 and RL78/F15 microcontrollers (MCUs).

Target Device

RL78/F13, RL78/F14, RL78/F15

Contents

1. **Overview of the Safety Function** 3
2. **Flash Memory CRC Operation Function (High-speed CRC)** 4
2.1 Overview of High-speed CRC Operation 4
2.2 Registers used for High-speed CRC Operation 5
2.3 Processing Example of High-speed CRC Operation 6
2.3.1 Processing Example of High-speed CRC Operation (Operation Range: 64 KB) 7
2.4 Cautions when Using High-speed CRC Operation 8
3. **CRC Operation Function (General-purpose CRC)** 9
3.1 Overview of General-purpose CRC Operation 9
3.2 General-purpose CRC Operation Registers 9
3.3 Processing Example of General-purpose CRC Operation 10
3.4 Cautions when Using General-purpose CRC Operation 10
4. **RAM-ECC Function** 11
4.1 Overview of RAM-ECC Function 11
4.2 Registers used for RAM-ECC Function 12
4.3 Processing Example of RAM-ECC Function 14
4.4 ECC Test Mode 15
4.5 Cautions when Using RAM-ECC Function 16
5. **CPU Stack Pointer Monitor Function** 17
5.1 Overview of CPU Stack Pointer Monitor Function 17
5.2 Registers used for CPU Stack Pointer Monitor Function 17
5.3 Processing Example of CPU Stack Pointer Monitor Function 18
5.4 Interrupt Determination of CPU Stack Pointer Monitor Function 19
5.5 Cautions when Using CPU Stack Pointer Monitor Function 19
6. **Clock Monitor Function** 20
6.1 Overview of Clock Monitor Function 20
6.2 Registers used for Clock Monitor Function 20
6.3 Processing Example of Clock Monitor Function 21
6.4 Interrupt Determination of Clock Monitor Function 22
6.5 Cautions when Using Clock Monitor Function 23
7. **RAM Guard Function** 24
7.1 Overview of RAM Guard Function 24

7.2 Registers used for RAM Guard Function 24
7.3 Processing Example of RAM Guard Function 25
7.4 Cautions when Using RAM Guard Function 25
8. **SFR Guard Function** 26
8.1 Overview of SFR Guard Function 26
8.2 Registers used for SFR Guard Function 26
8.3 Processing Example of SFR Guard Function 27
8.4 Cautions when Using SFR Guard Function 27
9. **Invalid Memory Access Detection Function** 28
9.1 Overview of Invalid Memory Access Detection Function 28
9.2 Registers used for Invalid Memory Access Detection Function 30
9.3 Processing Example of Invalid Memory Access Detection Function 30
9.4 Cautions when Using Invalid Memory Access Detection Function 31
10. **Frequency Detection Function** 32
10.1 Overview of Frequency Detection Function 32
10.2 Registers used for Frequency Detection Function 32
10.3 Processing Example of Frequency Detection Function 33
11. **A/D Test Function** 34
11.1 Overview of A/D Test Function 34
11.2 Registers used for A/D Test Function 34
11.3 Processing Example of A/D Test Function 35
11.4 Cautions when Using A/D Test Function 36
12. **I/O Port Output Signal Level Detection Function** 37
12.1 Overview of I/O Port Output Signal Level Detection Function 37
12.2 Registers used for I/O Port Output Signal Level Detection Function 37
12.3 Processing Example of I/O Port Output Signal Level Detection Function 38
12.4 Cautions when using I/O Port Output Signal Level Detection Function 38
Revision History 39

1. Overview of the Safety Function

To detect any errors and failures with the built-in self-test function, the RL78/F13, RL78/F14 and RL78/F15 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) RAM-ECC function

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

(3) CPU stack pointer monitor function

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

(4) 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.

(5) RAM guard function

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

(6) 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.

(7) 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.

(8) 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}).

(9) A/D test function

This function performs self-diagnosis for the A/D converter.

(10) I/O port output signal level detection function

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.

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}=32 MHz)	Register setting
16 KB (00000H – 03FFBH)	4095 clocks (approx. 128 µs)	CRC0CTL.FEA[5:0] = 000000B
32 KB (00000H – 07FFBH)	8191 clocks (approx. 256 µs)	CRC0CTL.FEA[5:0] = 000001B
48 KB (00000H – 0BFFBH)	12287 clocks (approx. 384 µs)	CRC0CTL.FEA[5:0] = 000010B
64 KB (00000H – 0FFFBH)	16383 clocks (approx. 512 µs)	CRC0CTL.FEA[5:0] = 000011B
80 KB (00000H – 13FFBH)	20479 clocks (approx. 640 µs)	CRC0CTL.FEA[5:0] = 000100B
96 KB (00000H – 17FFBH)	24575 clocks (approx. 768 µs)	CRC0CTL.FEA[5:0] = 000101B
112 KB (00000H – 1BFFBH)	28671 clocks (approx. 896 µs)	CRC0CTL.FEA[5:0] = 000110B
128 KB (00000H – 1FFFBH)	32767 clocks (approx. 1024 µs)	CRC0CTL.FEA[5:0] = 000111B
144 KB (00000H – 23FFBH)	36863 clocks (approx. 1152 µs)	CRC0CTL.FEA[5:0] = 001000B
160 KB (00000H – 27FFBH)	40959 clocks (approx. 1280 µs)	CRC0CTL.FEA[5:0] = 001001B
176 KB (00000H – 2BFFBH)	45055 clocks (approx. 1408 µs)	CRC0CTL.FEA[5:0] = 001010B
192 KB (00000H – 2FFFBH)	49151 clocks (approx. 1536 µs)	CRC0CTL.FEA[5:0] = 001011B
208 KB (00000H – 33FFBH)	53247 clocks (approx. 1664 µs)	CRC0CTL.FEA[5:0] = 001100B
224 KB (00000H – 37FFBH)	57343 clocks (approx. 1792 µs)	CRC0CTL.FEA[5:0] = 001101B
240 KB (00000H – 3BFFBH)	61439 clocks (approx. 1920 µs)	CRC0CTL.FEA[5:0] = 001110B
256 KB (00000H – 3FFFBH)	65535 clocks (approx. 2048 µs)	CRC0CTL.FEA[5:0] = 001111B
272 KB (00000H – 43FFBH)	69631 clocks (approx. 2176 µs)	CRC0CTL.FEA[5:0] = 010000B
288 KB (00000H – 47FFBH)	73727 clocks (approx. 2304 µs)	CRC0CTL.FEA[5:0] = 010001B
304 KB (00000H – 4BFFBH)	77823 clocks (approx. 2432 µs)	CRC0CTL.FEA[5:0] = 010010B
320 KB (00000H – 4FFFBH)	81919 clocks (approx. 2560 µs)	CRC0CTL.FEA[5:0] = 010011B
336 KB (00000H – 53FFBH)	86015 clocks (approx. 2688 µs)	CRC0CTL.FEA[5:0] = 010100B
352 KB (00000H – 57FFBH)	90111 clocks (approx. 2816 µs)	CRC0CTL.FEA[5:0] = 010101B
368 KB (00000H – 5BFFBH)	94207 clocks (approx. 2944 µs)	CRC0CTL.FEA[5:0] = 010110B
384 KB (00000H – 5FFFBH)	98303 clocks (approx. 3072 µs)	CRC0CTL.FEA[5:0] = 010111B
400 KB (00000H – 63FFBH)	102399 clocks (approx. 3200 µs)	CRC0CTL.FEA[5:0] = 011000B
416 KB (00000H – 67FFBH)	106495 clocks (approx. 3328 µs)	CRC0CTL.FEA[5:0] = 011001B
432 KB (00000H – 6BFFBH)	110591 clocks (approx. 3456 µs)	CRC0CTL.FEA[5:0] = 011010B
448 KB (00000H – 6FFFBH)	114687 clocks (approx. 3584 µs)	CRC0CTL.FEA[5:0] = 011011B
464 KB (00000H – 73FFBH)	118783 clocks (approx. 3712 µs)	CRC0CTL.FEA[5:0] = 011100B
480 KB (00000H – 77FFBH)	122879 clocks (approx. 3840 µs)	CRC0CTL.FEA[5:0] = 011101B
496 KB (00000H – 7BFFBH)	126975 clocks (approx. 3968 µs)	CRC0CTL.FEA[5:0] = 011110B
512 KB (00000H – 7FFFBH)	131071 clocks (approx. 4096 µs)	CRC0CTL.FEA[5:0] = 011111B

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. 1: Starts the high-speed CRC operation upon execution of the HALT instruction.
FEA[5:0]	Specify the high-speed CRC operation range. **Note** 000000B: 16 KB 000001B: 32 KB 000010B: 48 KB 000011B: 64 KB 000100B: 80 KB 000101B: 96 KB 000110B: 112 KB 000111B: 128 KB 001000B: 144 KB 001001B: 160 KB 001010B: 176 KB 001011B: 192 KB 001100B: 208 KB 001101B: 224 KB 001110B: 240 KB 001111B: 256 KB 010000B: 272 KB 010001B: 288 KB 010010B: 304 KB 010011B: 320 KB 010100B: 336 KB 010101B: 352 KB 010110B: 368 KB 010111B: 384 KB 011000B: 400 KB 011001B: 416 KB 011010B: 432 KB 011011B: 448 KB 011100B: 464 KB 011101B: 480 KB 011110B: 496 KB 011111B: 512 KB 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

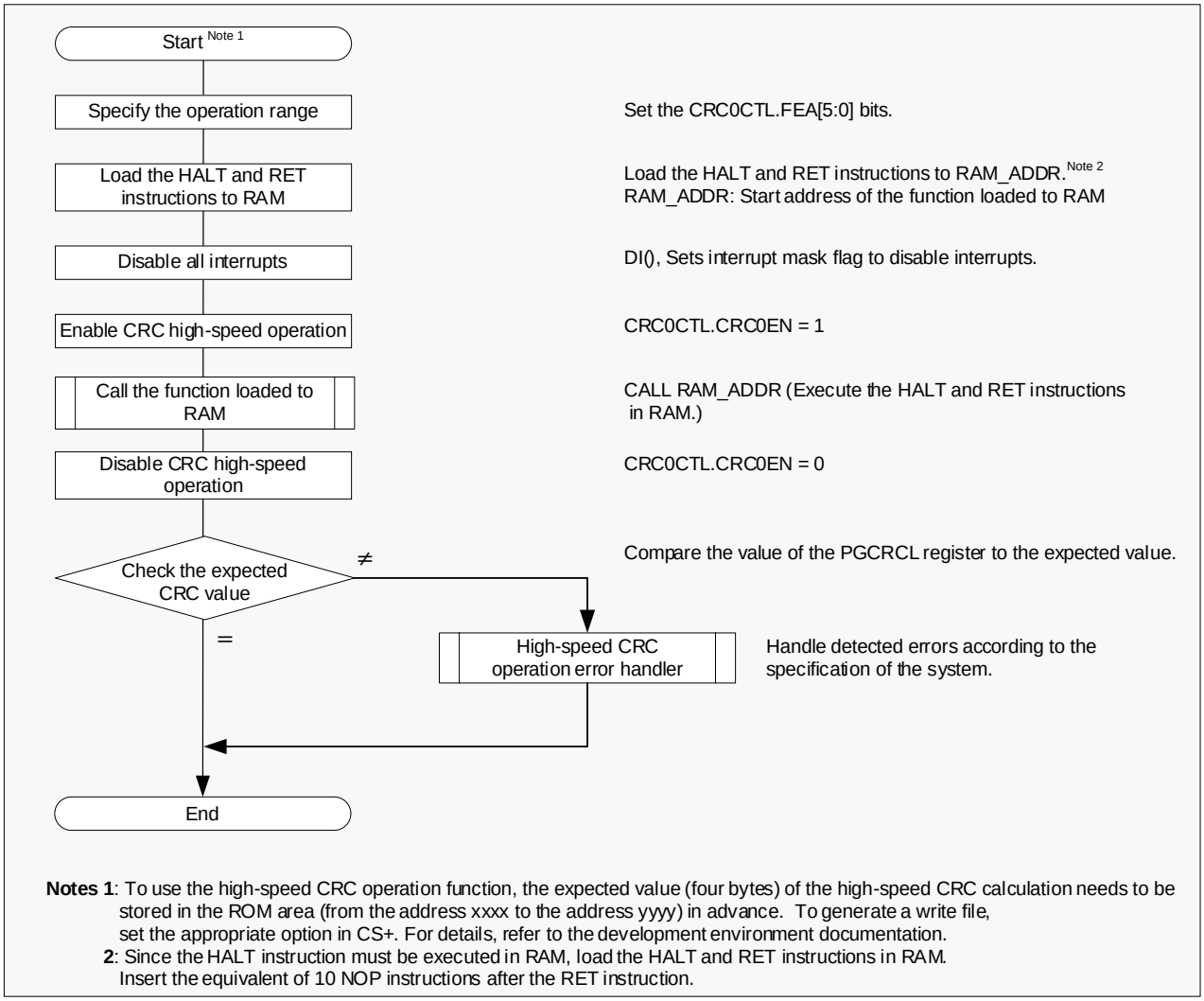
Symbol	15	0
PGCRCL	PGCRC[15:0]	

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 Processing Example of High-speed CRC Operation

Figure 2-1 is an example of using the high-speed CRC operation.

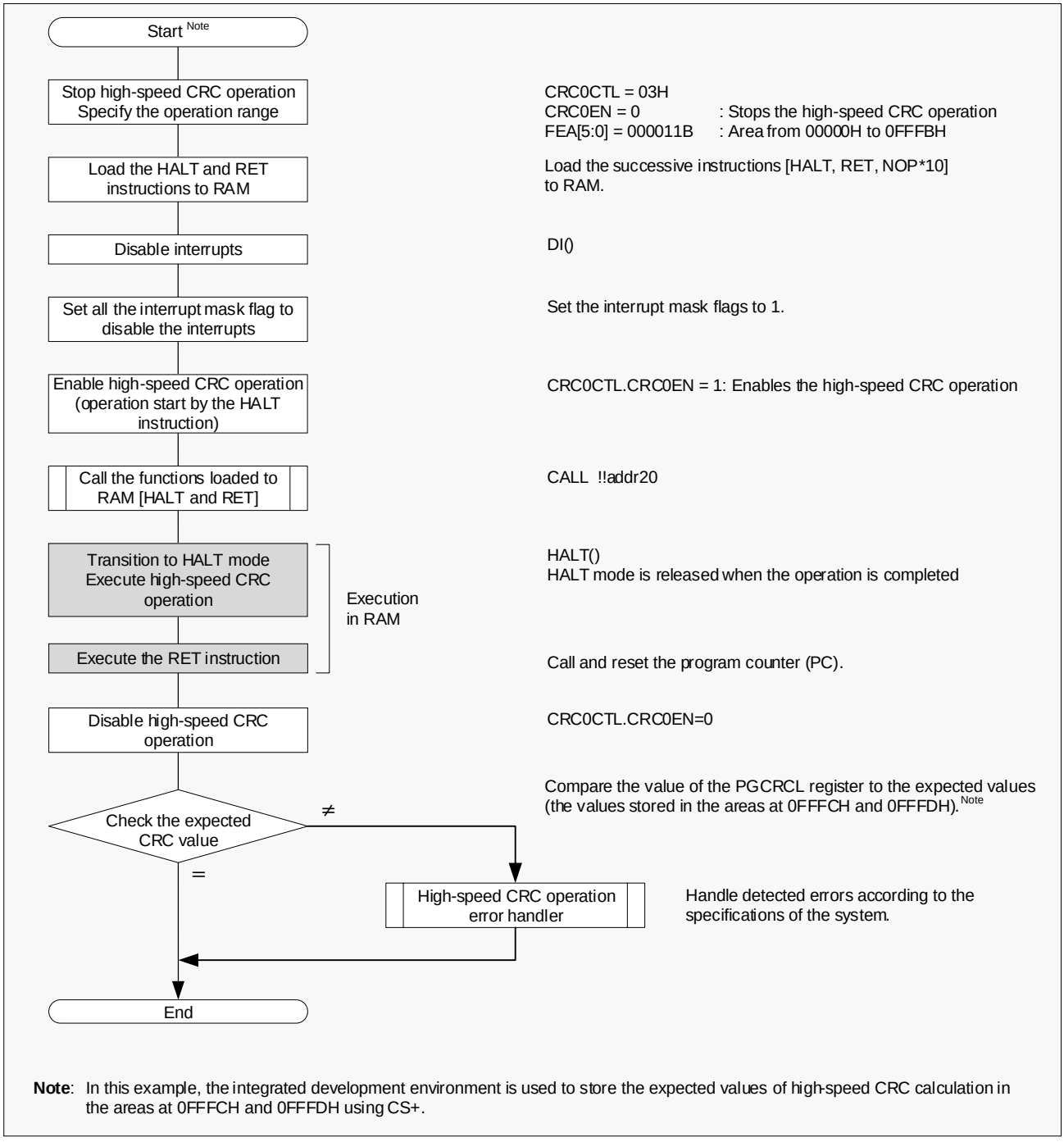


Notes 1: To use the high-speed CRC operation function, the expected value (four bytes) of the high-speed CRC calculation needs to be stored in the ROM area (from the address xxxx to the address yyyy) in advance. To generate a write file, set the appropriate option in CS+. For details, refer to the development environment documentation.
2: Since the HALT instruction must be executed in RAM, load the HALT and RET instructions in RAM. Insert the equivalent of 10 NOP instructions after the RET instruction.

Figure 2-1. Example of Using High-speed CRC Operation

2.3.1 Processing Example of High-speed CRC Operation (Operation Range: 64 KB)

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



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 Using High-speed CRC Operation (Range: 64 KB)

2.4 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. 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.

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	6	5	4	3	2	1	0
CRCIN	CRCIN[7:0]							

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

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

(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. When supporting CRC-CCITT: 0000H to FFFFH When conforming to SENT: 0000H to 000FH

3.3 Processing Example of General-purpose CRC Operation

Figure 3-1 is an example of using the general-purpose CRC operation.

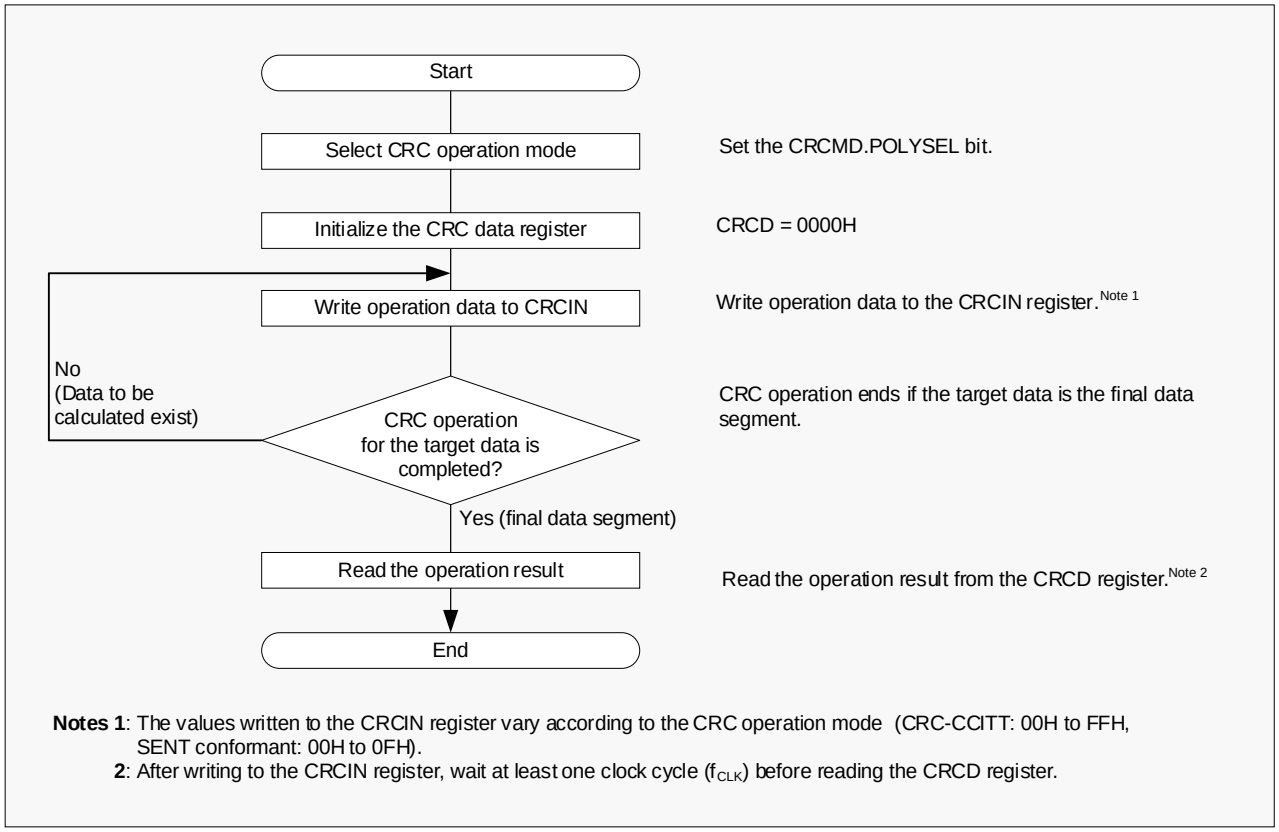


Figure 3-1. Example of Using General-purpose CRC Operation

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.

4. RAM-ECC Function

4.1 Overview of RAM-ECC Function

The 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 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 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

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.

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.

4.2 Registers used for RAM-ECC Function

The registers used for the 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	–	–	–	–	–	–	–	IEN

Bit Name	Description
IEN	0: Disables the interrupt generation for a 1-bit error detected. **Note** 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	–	–	–	–	–	–	–	DBERR

Bit Name	Description
DBERR **Note**	0: A 1-bit error detected (Error correction) 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	−	−	−	−	−	TPR[2:0]		

Bit Name	Description
TPR[2:0]	Other than 111B : Disables the access to the ECCTMDR register. 111B : Enables 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	−	−	−	−	−	TMD[2:0]		

Bit Name	Description
TMD[2:0]	000B: Normal operating mode 001B: ECC test mode 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	−	−	−	PRTYRV	ECCRV[3:0]			

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

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

Remarks i: 0 to 3, j: 0 to 7

4.3 Processing Example of RAM-ECC Function

Figure 4-1 is an example of using the RAM-ECC function.

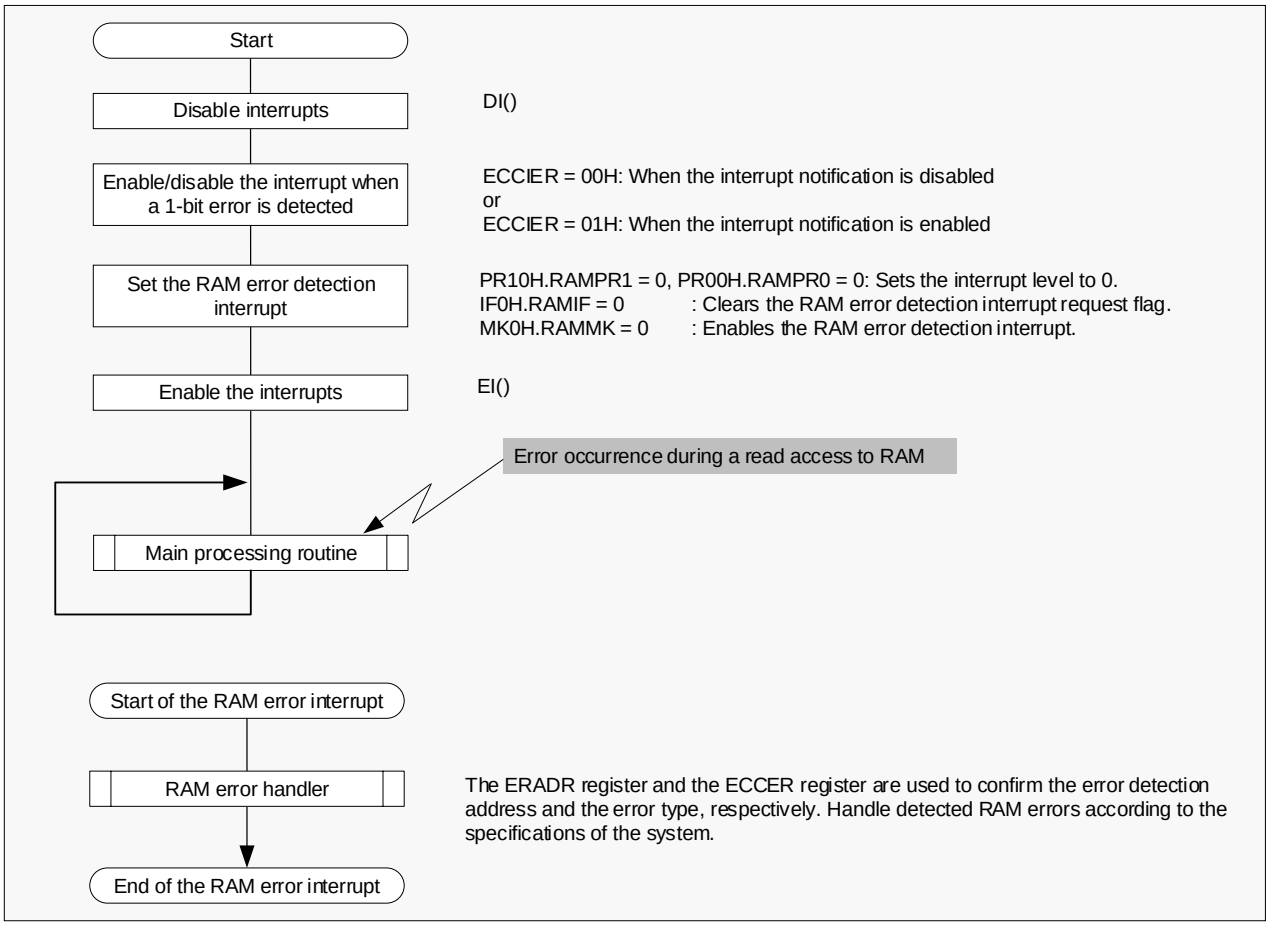


Figure 4-1. Example of Using RAM-ECC Function

4.4 ECC Test Mode

In ECC test mode, operations of the 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 RAM must not be accessed. (Enable this mode when, for example, initialization of RAM is being executed.)

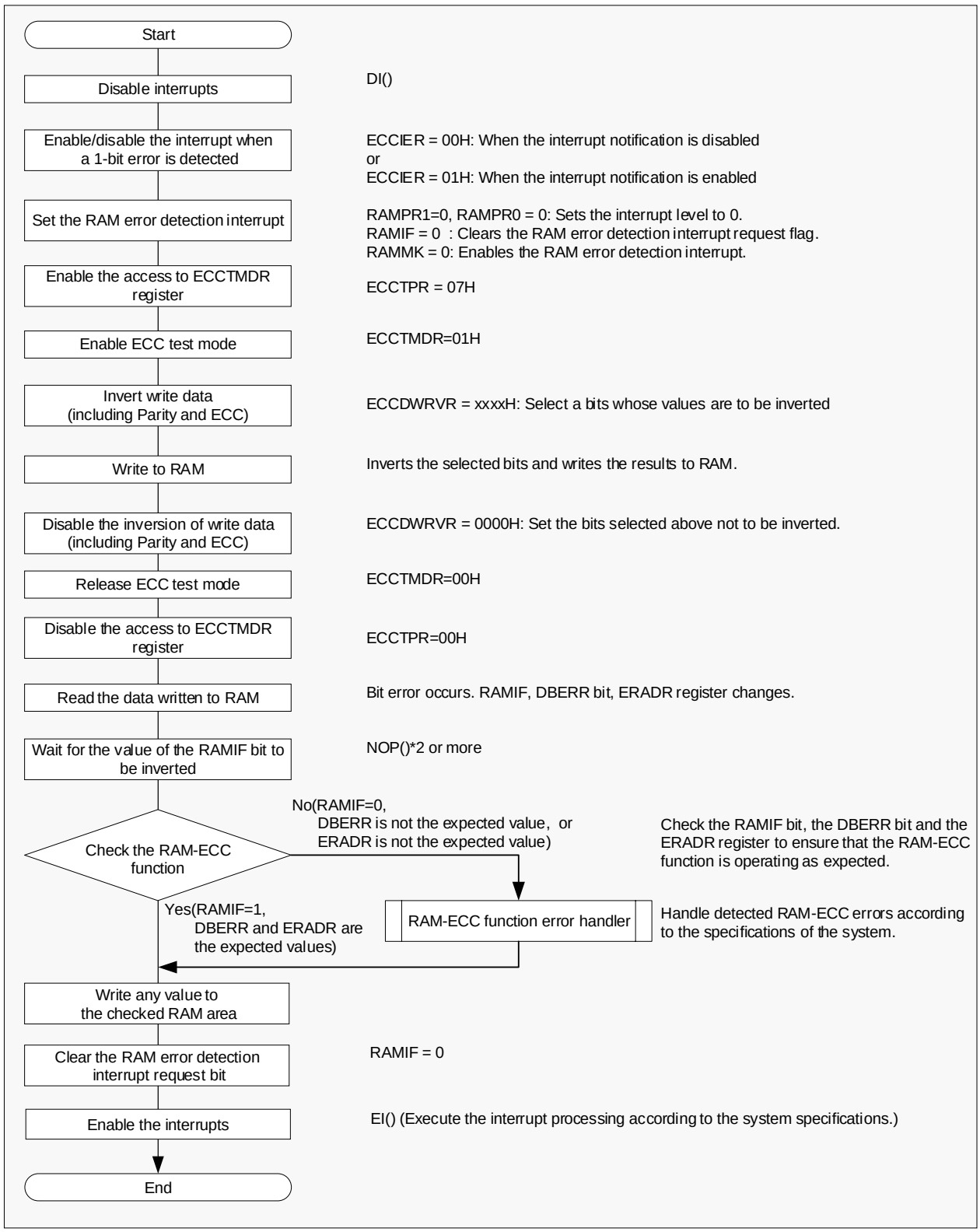


Figure 4-2. Example of ECC Test Mode

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

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

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.

4.5 Cautions when Using RAM-ECC Function

The following are the cautions when using the 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 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.

(4) When a program is executed in RAM, error detection/correction will not be performed even if a bit-error occurs at instruction fetch.

5. CPU Stack Pointer Monitor Function

5.1 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.

5.2 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: 0000H R/W

Symbol	7	6	5	4	3	2	1	0
SPMCTRL	SPMEN	−	−	−	−	−	−	−

Bit Name	Description
SPMEN Note	0: Disables the stack pointer monitor function. 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	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPOFR	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	0

Bit Name	Description
15 - 0 Note	Sets a stack pointer overflow address. 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 stack pointer underflow address (lower limit). This register can be accessed by a 16-bit memory manipulation instruction.

Address: F00DCH After reset: 0000H R/W

Symbol	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPUFR	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	0

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

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

5.3 Processing Example of CPU Stack Pointer Monitor Function

Figure 5-1 is an example of using the CPU stack pointer monitor function.

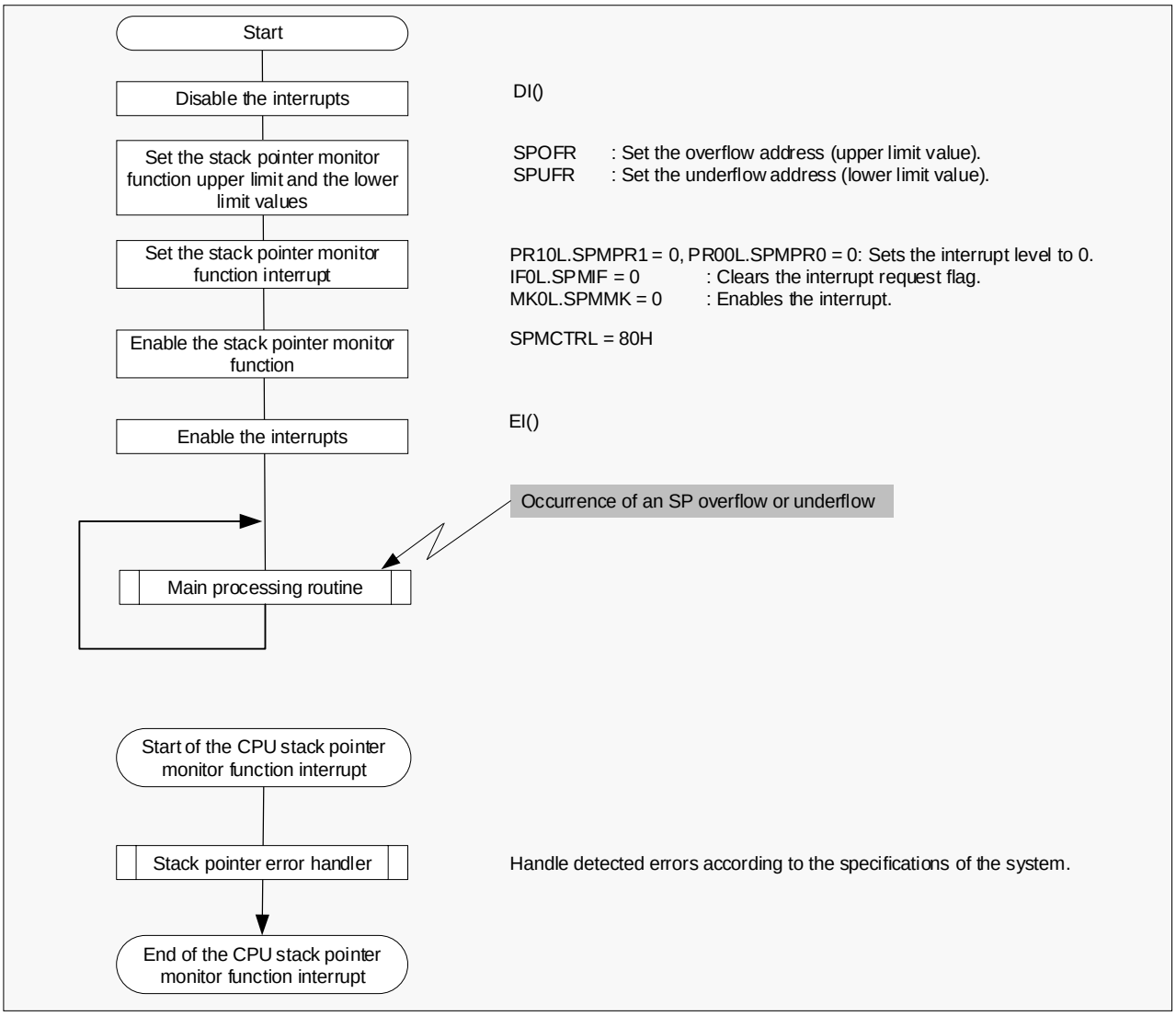


Figure 5-1. Example of Using CPU Stack Pointer Monitor Function

5.4 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 5-2 shows the interrupt determination processing of the CPU stack pointer monitor function.

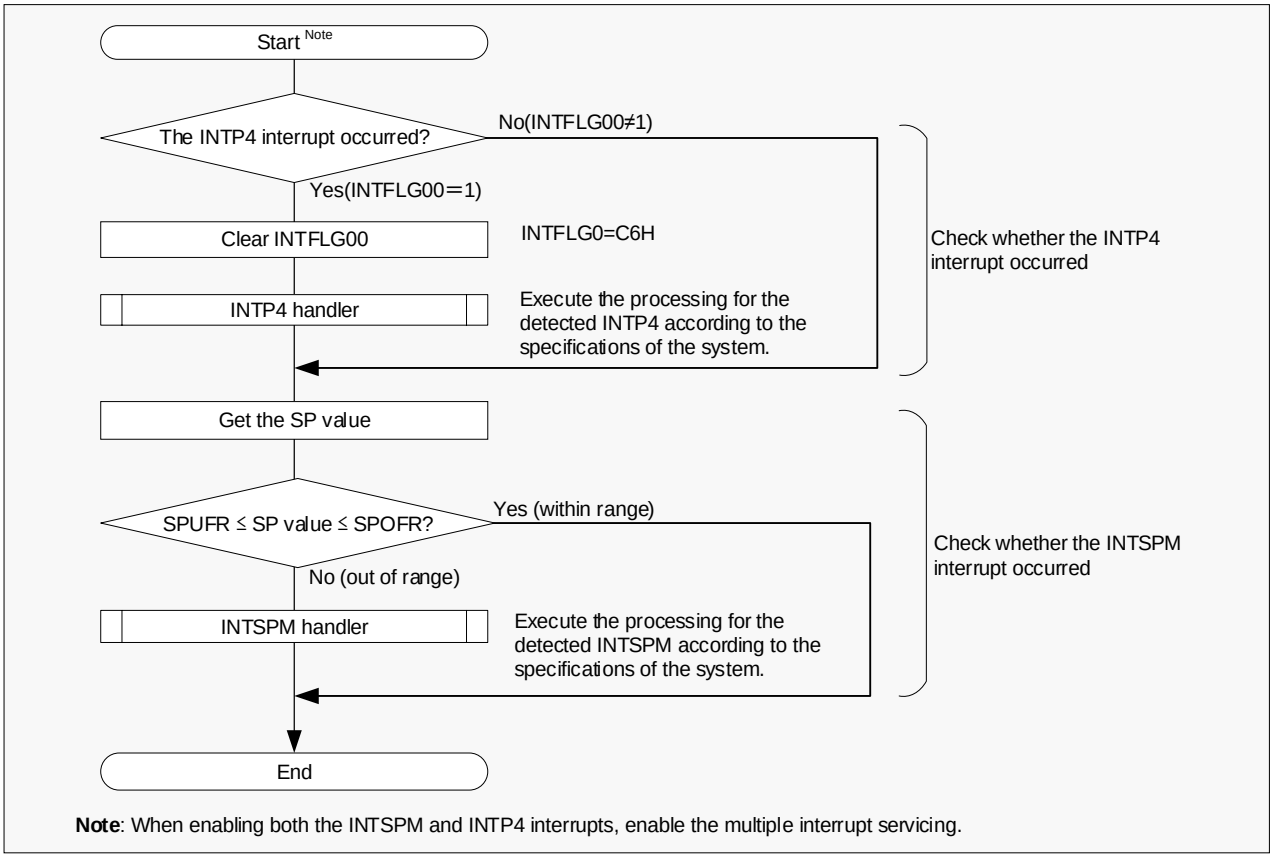


Figure 5-2. Determination of INTSPM and INTP4 Interrupts

5.5 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 interrupts together, the source of the interrupt needs to be determined by SP overflow or underflow interrupt.

6. Clock Monitor Function

6.1 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.

6.2 Registers used for Clock Monitor Function

The option byte area in the code flash memory used for the clock monitor function is described below.

(1) User option byte (000C1H/020C1H)

Set the operation enables/disables of the clock monitor function by setting this area.

Address: 000C1H/020C1H After reset: － (User setting value)

	7	6	5	4	3	2	1	0
000C1H/020C1H	VPOC[2:0]			CLKMB	LVIS[1:0]		LVIMDS[1:0]	

Bit Name	Description
CLKMB	0: Enables the clock monitor function. 1: Stops the clock monitor function.

6.3 Processing Example of Clock Monitor Function

Figure 6-1 is an example of using the clock monitor function.

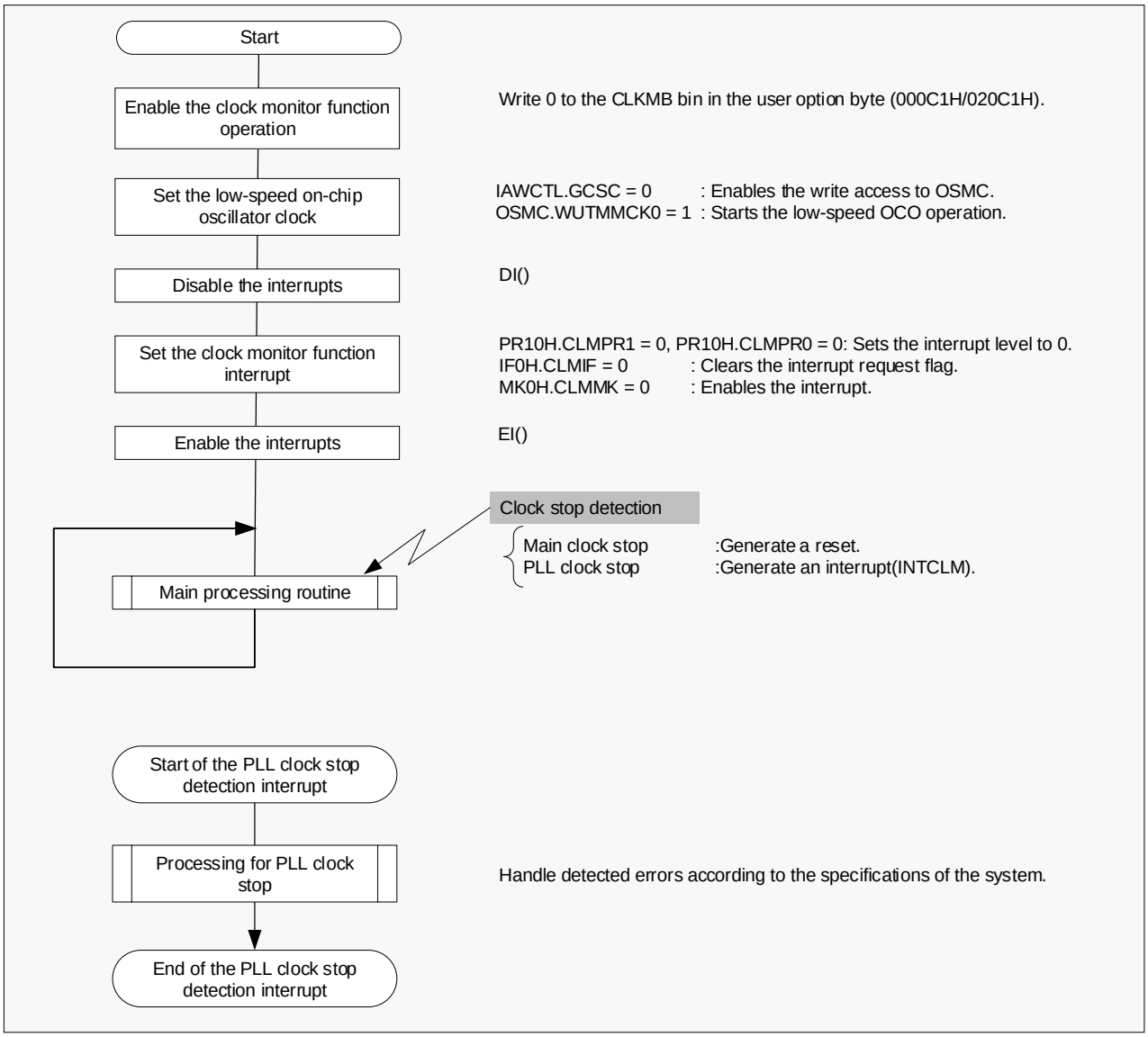


Figure 6-1. Example of Using Clock Monitor Function

6.4 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 6-2 shows the interrupt determination processing of the clock monitor function.

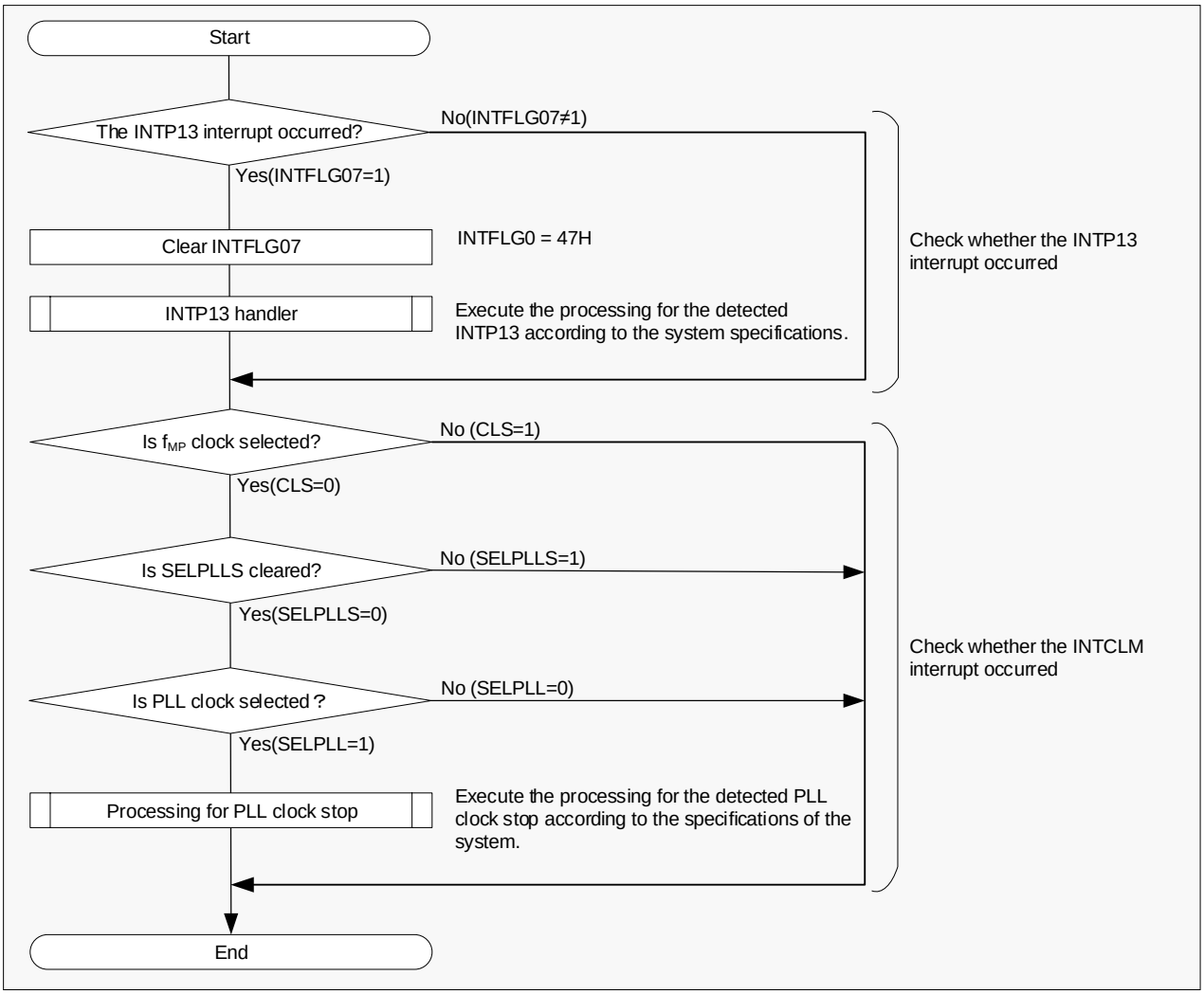


Figure 6-2. Determination of PLL Clock Stop Detection and INTP13 Interrupts

6.5 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/020C1H) 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.

7. RAM Guard Function

7.1 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.

7.2 Registers used for RAM Guard Function

The register used by the RAM guard function is 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) settings 00B: Disabled 01B: 128 bytes starting from the RAM lowest address 10B: 256 bytes starting from the RAM lowest address 11B: 512 bytes starting from the RAM lowest address

Table 7-1. RAM Guard Space by RL78/F13 (LIN-incorporated) products

Product	GRAM[1:0]=01B	GRAM[1:0]=10B	GRAM[1:0]=11B
R5F10AnA (n=6, A, B, G)	FFB00H – FFB7FH	FFB00H – FFBFFH	FFB00H – FFCFFH
R5F10AnC (n=6, A, B, G, L)	FF700H – FF77FH	FF700H – FF7FFH	FF700H – FF8FFH
R5F10AnD (n=6, A, B, G, L)	FF300H – FF37FH	FF300H – FF3FFH	FF300H – FF4FFH
R5F10AnE (n=6, A, B, G, L, M)	FEF00H – FEF7FH	FEF00H – FEFFFH	FEF00H – FF0FFH
R5F10AnF (n=G, L, M)	FE700H – FE77FH	FE700H – FE7FFH	FE700H – FE8FFH
R5F10AnG (n=G, L, M)	FDF00H – FDF7FH	FDF00H – FDFFFH	FDF00H – FE0FFH

Table 7-2. RAM Guard Space by RL78/F13 (CAN&LIN-incorporated) products

Product	GRAM[1:0]=01B	GRAM[1:0]=10B	GRAM[1:0]=11B
R5F10BnC (n=A, B, G, L)	FF700H – FF77FH	FF700H – FF7FFH	FF700H – FF8FFH
R5F10BnD (n=A, B, G, L)	FF300H – FF37FH	FF300H – FF3FFH	FF300H – FF4FFH
R5F10BnE (n=A, B, G, L, M)	FEF00H – FEF7FH	FEF00H – FEFFFH	FEF00H – FF0FFH
R5F10BnF (n=G, L, M)	FE700H – FE77FH	FE700H – FE7FFH	FE700H – FE8FFH
R5F10BnG (n=G, L, M)	FDF00H – FDF7FH	FDF00H – FDFFFH	FDF00H – FE0FFH

Table 7-3. RAM Guard Space by RL78/F14 products

Product	GRAM[1:0]=01B	GRAM[1:0]=10B	GRAM[1:0]=11B
R5F10PnD (n=A, B, GL)	FF300H – FF37FH	FF300H – FF3FFH	FF300H – FF4FFH
R5F10PnE (n=A, B, G, L, M, P)	FEF00H – FEF7FH	FEF00H – FEFFFH	FEF00H – FF0FFH
R5F10PnF (n=G, L, M, P)	FE700H – FE77FH	FE700H – FE7FFH	FE700H – FE8FFH
R5F10PnG (n=G, L, M, P)	FDF00H – FDF7FH	FDF00H – FDFFFH	FDF00H – FE0FFH
R5F10PnH (n=G, L, M, P)	FBF00H – FBF7FH	FBF00H – FBFFFH	FBF00H – FC0FFH
R5F10PnJ (n=G, L, M, P)	FAF00H – FAF7FH	FAF00H – FAFFFH	FAF00H – FB0FFH

Table 7-4. RAM Guard Space by RL78/F15 products

Product	GRAM[1:0]=01B	GRAM[1:0]=10B	GRAM[1:0]=11B
R5F113nG (n=P, T)	FDF00H – FDF7FH	FDF00H – FDFFFH	FDF00H – FE0FFH
R5F113nH (n=P, T)	FBF00H – FBF7FH	FBF00H – FBFFFH	FBF00H – FC0FFH
R5F113nJ (n=P, T)	FAF00H – FAF7FH	FAF00H – FAFFFH	FAF00H – FB0FFH
R5F113nK (n=G, L, M, P, T)	F9700H – F977FH	F9700H – F97FFH	F9700H – F98FFH
R5F113nL (n=G, L, M, P, T)	F7F00H – F7F7FH	F7F00H – F7FFFH	F7F00H – F80FFH

7.3 Processing Example of RAM Guard Function

Figure 7-1 is an example of using the RAM guard function.

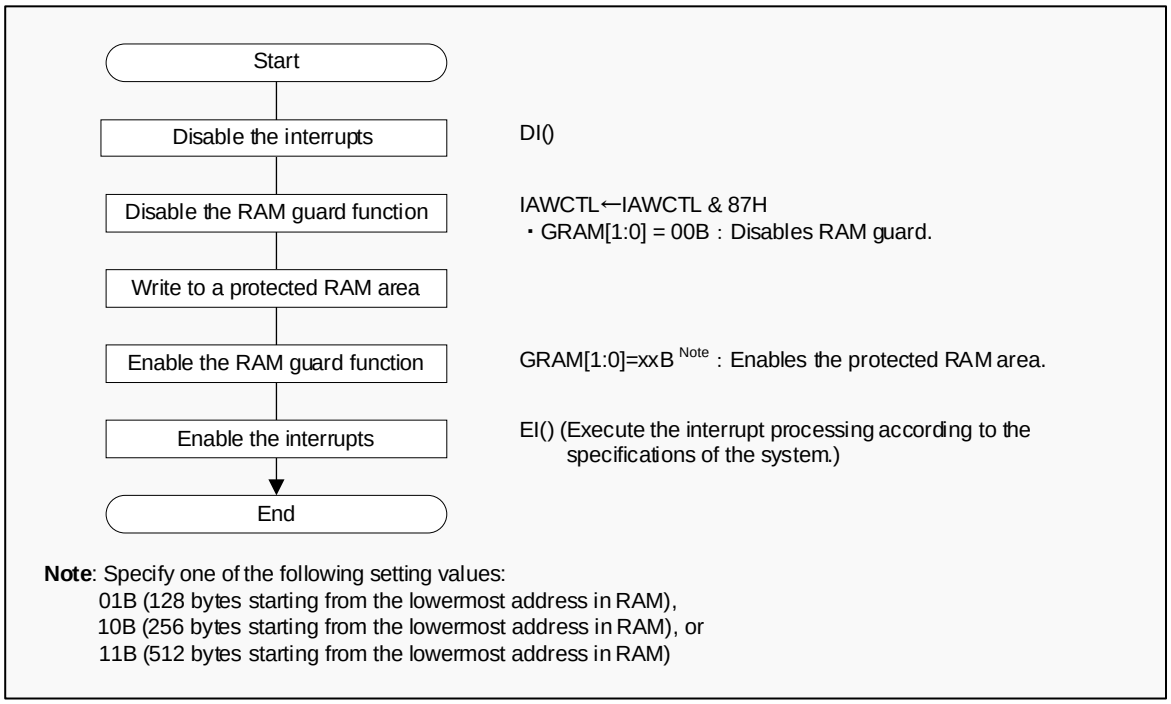


Figure 7-1. Example of Using RAM Guard Function

7.4 Cautions when Using RAM Guard Function

The following are the cautions 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 a RAM area that is used as a stack to an area where the RAM guard function is enabled.

8. SFR Guard Function

8.1 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 8-1 lists the registers that the SFR guard function can protect.

Table 8-1. SFR Guard-target Registers

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

Note: The target registers vary depending on products (according to ports or interrupt implemented).

8.2 Registers used for SFR Guard Function

The register used by the SFR guard function is 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 0: Disabled 1: Enabled
GINT	Guard of registers for the interrupt function 0: Disabled 1: Enabled
GCSC	Guard of registers for the clock control and the voltage detector 0: Disabled 1: Enabled

8.3 Processing Example of SFR Guard Function

Figure 8-1 is an example of using the SFR guard function.

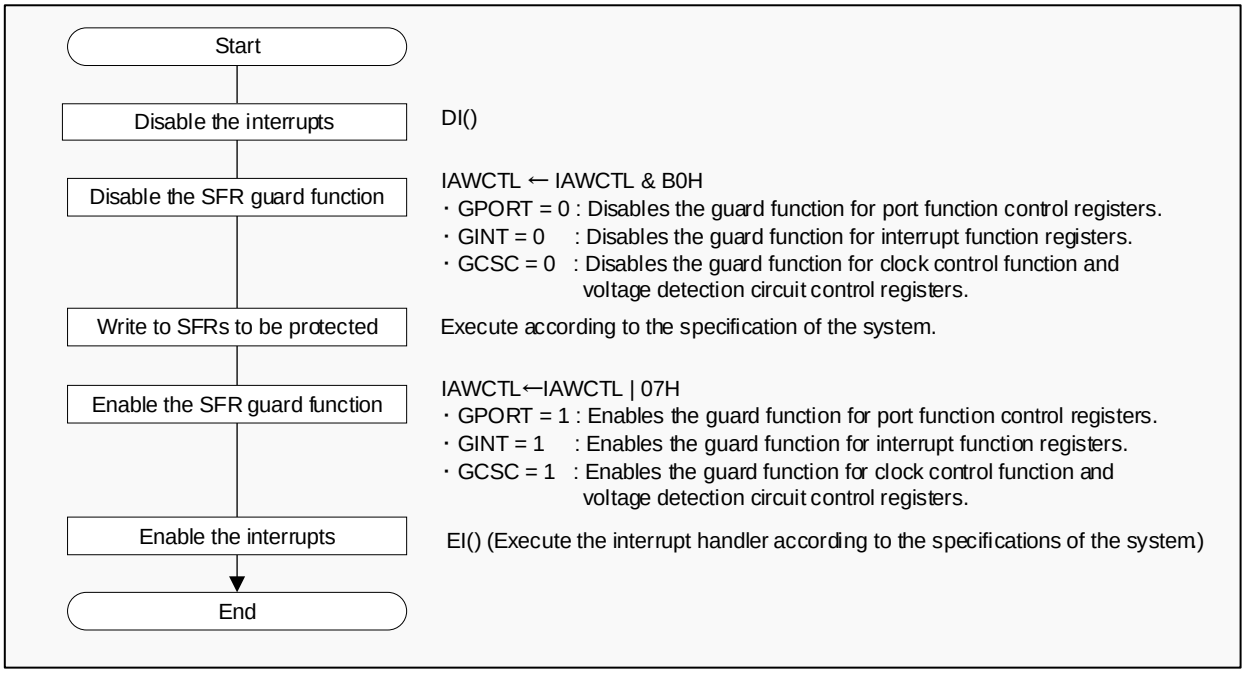


Figure 8-1. Example of Using SFR Guard Function

8.4 Cautions when Using SFR Guard Function

The following is the caution 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.)

9. Invalid Memory Access Detection Function

9.1 Overview of Invalid Memory Access Detection Function

The invalid memory access detection function triggers a reset if an invalid access detection space (see Figure 9-1) is accessed.

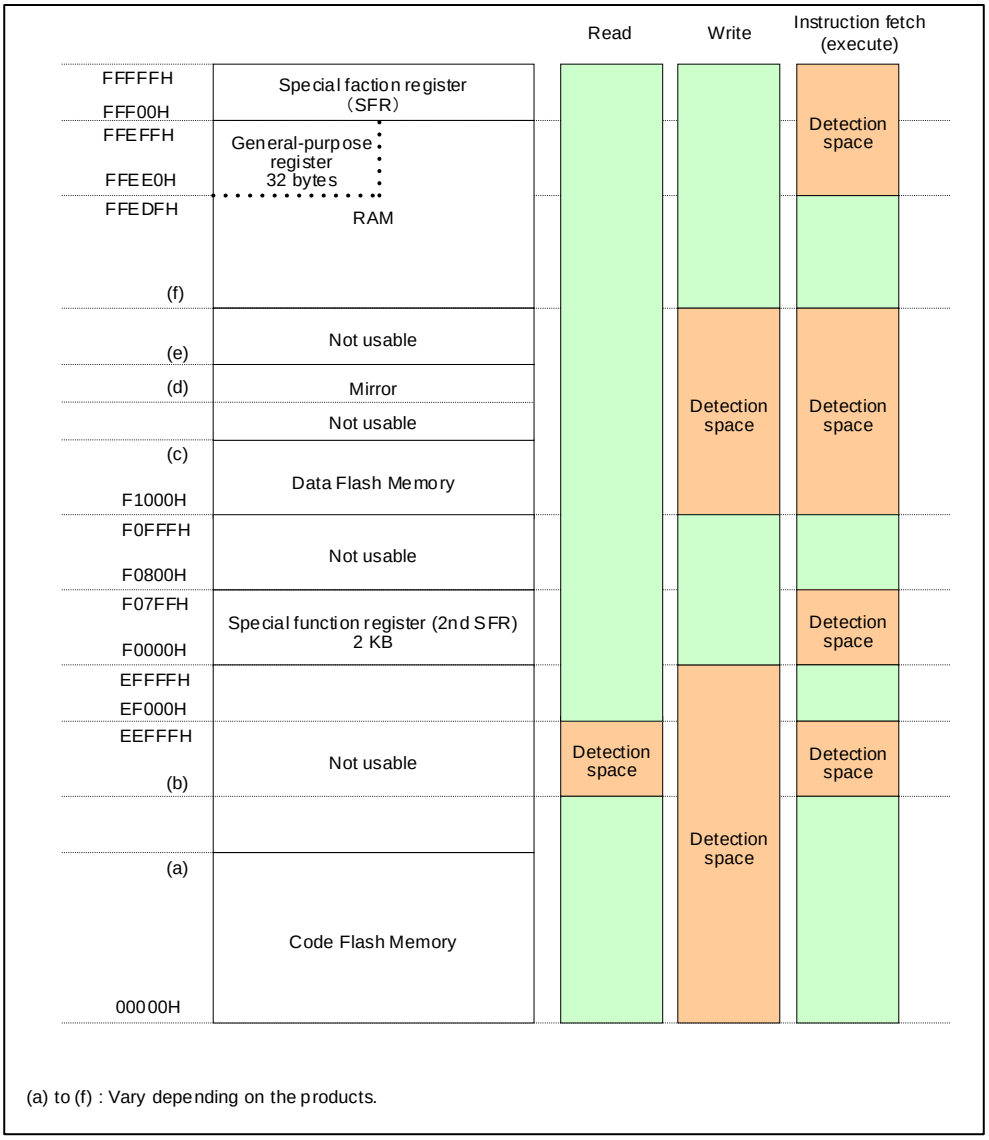


Figure 9-1. Invalid Access Detection Area

Table 9-1. Memory Space by RL78/F13 (LIN incorporated) Models

Products (models)	(a)	(b)	(c)	(d)	(e)	(f)
R5F10AnA (n=6, A, B, G)	03FFFH	10000H	F1FFFH	F2000H	F4000H	FFB00H
R5F10AnC (n=6, A, B, G, L)	07FFFH	10000H	F1FFFH	F2000H	F8000H	FF700H
R5F10AnD (n=6, A, B, G, L)	0BFFFH	10000H	F1FFFH	F2000H	FC000H	FF300H
R5F10AnE (n=6, A, B, G, L)	0FFFFH	10000H	F1FFFH	F2000H	−	FEF00H
R5F10AME	0FFFFH	10000H	F1FFFH	F2000H	FDF00H	FEF00H
R5F10AnF (n=G, L, M)	17FFFH	20000H	F1FFFH	F2000H	FDF00H	FE700H
R5F10AnG (n=G, L, M)	1FFFFH	20000H	F1FFFH	F2000H	−	FDF00H

Table 9-2. Memory Space by RL78/F13 (CAN&LIN incorporated) Models

Products (models)	(a)	(b)	(c)	(d)	(e)	(f)
R5F10BnC (n=A, B, G, L)	07FFFH	10000H	F1FFFH	F2000H	F8000H	FF700H
R5F10BnD (n=A, B, G, L)	0BFFFH	10000H	F1FFFH	F2000H	FC000H	FF300H
R5F10BnE (n=A, B, G, L, M)	0FFFFH	10000H	F1FFFH	F2000H	FDF00H	FEF00H
R5F10BnF (n=A, B, G, L, M)	17FFFH	20000H	F1FFFH	F2000H	FDF00H	FE700H
R5F10BnG (n=A, B, G, L, M)	1FFFFH	20000H	F1FFFH	F2000H	−	FDF00H

Table 9-3. Memory Space by RL78/F14 Models

Products (models)	(a)	(b)	(c)	(d)	(e)	(f)
R5F10PnD (n=A, B, G)	0BFFFH	10000H	F1FFFH	F2000H	FC000H	FEF00H
R5F10PnE (n=A, B, G, L, M)	0FFFFH	10000H	F1FFFH	F2000H	FDF00H	FE700H
R5F10PPE	0FFFFH	10000H	F1FFFH	F3000H	FAF00H	FE700H
R5F10PnF (n=G, L, M)	17FFFH	20000H	F1FFFH	F2000H	−	FDF00H
R5F10PPF	17FFFH	20000H	F1FFFH	F3000H	FAF00H	FDF00H
R5F10PnG (n=G, L, M, P)	1FFFFH	20000H	F2FFFH	F3000H	FAF00H	FDF00H
R5F10PnH (n=G, L, M, P)	2FFFFH	30000H	F2FFFH	F3000H	FAF00H	FBF00H
R5F10PnJ (n=G, L, M, P)	3FFFFH	40000H	F2FFFH	F3000H	−	FAF00H

Table 9-4. Memory Space by RL78/F15 Models

Products (models)	(a)	(b)	(c)	(d)	(e)	(f)
R5F113nG (n=P, T)	1FFFFH	20000H	F2FFFH	F5000H	F7F00H	FD700H
R5F113nH (n=P, T)	2FFFFH	30000H	F2FFFH	F5000H	F7F00H	FBF00H
R5F113nJ (n=P, T)	3FFFFH	40000H	F2FFFH	F5000H	F7F00H	FAF00H
R5F113nK (n=G, L, M, P, T)	5FFFFH	60000H	F4FFFH	F5000H	F7F00H	F9700H
R5F113nL (n=G, L, M, P, T)	7FFFFH	80000H	F4FFFH	F5000H	−	F7F00H

9.2 Registers used for Invalid Memory Access Detection Function

The register used by the invalid memory access detection function is 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. 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.

9.3 Processing Example of Invalid Memory Access Detection Function

Figure 9-2 is an example of using the invalid memory access detection function.

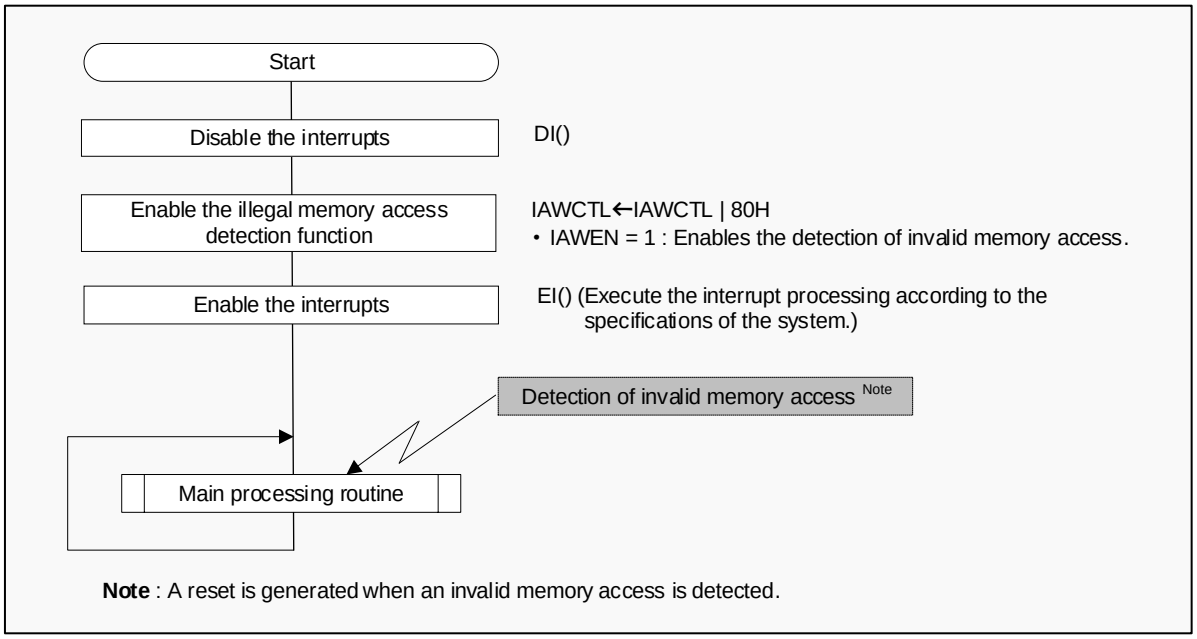


Figure 9-2. Example of Using Invalid Memory Access Detection Function

9.4 Cautions when Using Invalid Memory Access Detection 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/020C0H) is set to 1, the invalid memory access detection function is enabled regardless of the setting value to the IAWEN bit.

(2) With a product having a code flash memory whose size is 16KB/32KB/48KB, a value “FFH” will be read if the space [between the start address of the code flash memory (the address “(a)” in Figure 9-1) and the address 0FFFFH] is invalidly accessed. Also, if an instruction fetch is executed in the same space by an invalid access, a reset is generated due to execution of the invalid instruction.

(3) With a product having a code flash memory whose size is 96KB, a value “FFH” will be read if the space [between the start address of the code flash memory (an address “(a)” in Figure 9-1) and the address 1FFFFH] is invalidly accessed. Also, if an instruction fetch is executed in the same space by an invalid access, a reset is generated due to execution of the invalid instruction.

10. Frequency Detection Function

10.1 Overview of Frequency Detection Function

When either of the high-speed on-chip oscillator clock (f_{IH}), high-speed system 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.

10.2 Registers used for Frequency Detection Function

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

(1) Timer input select register (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]		

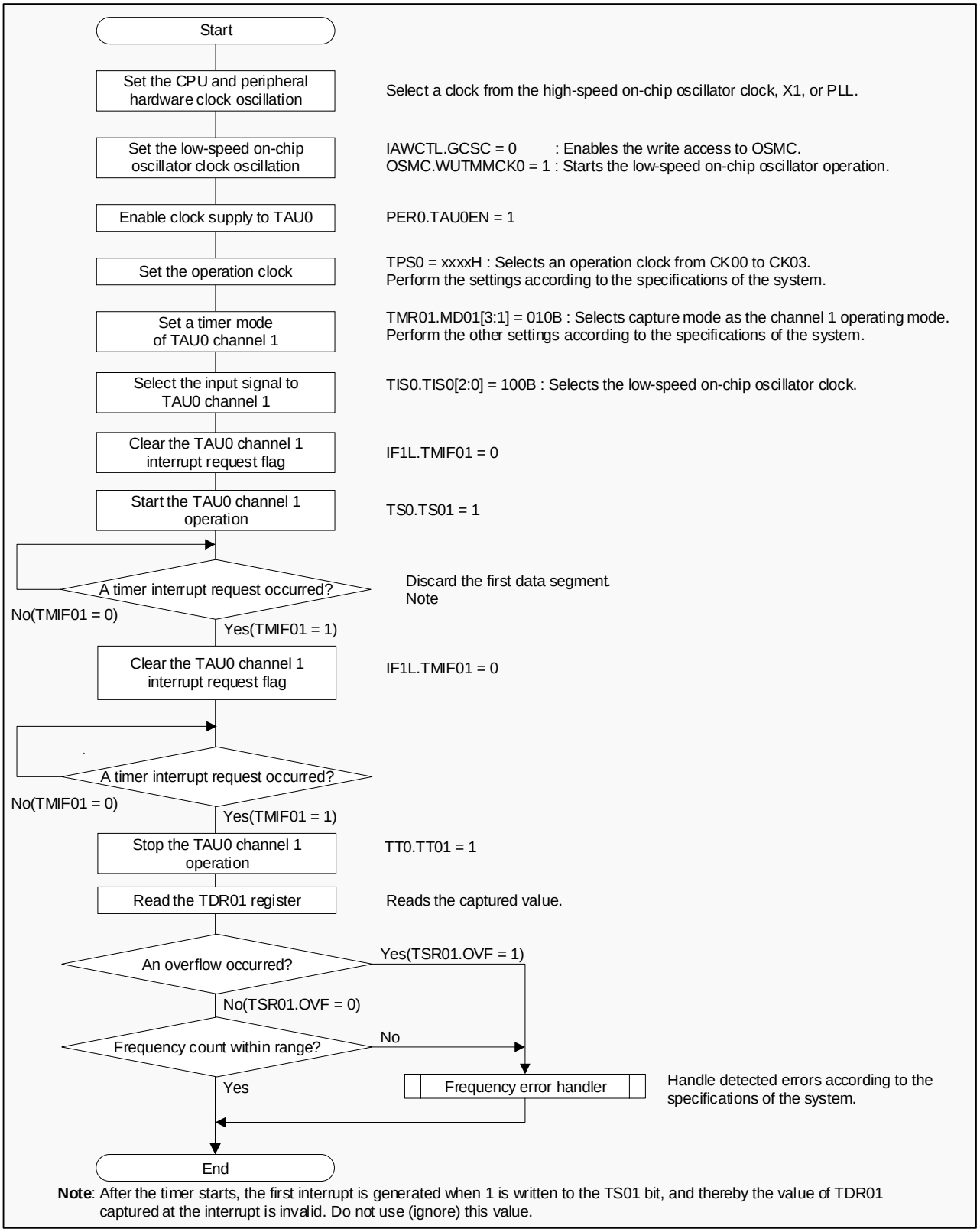
Bit Name	Description
TIS0[2:0]	Selects the timer input used for channel 1 of timer array unit 0. 000B, 010B, 011B: Input signal of timer input pin (TI01) 001B: Event input signal from ELC **100B: Low-speed on-chip oscillator clock (f_{IL})** 101B: Sub/low-speed on-chip oscillator select clock (f_{SL})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

(2) Timer array unit 0-related registers (TMR01, TS0, TDR01, TSR01, TT0)

- **Timer mode register 01 (TMR01)**
  CKS01[1:0]=Selects an operation clock (select any one from CK00-CK03 for the operation clock TAU01)
  CCS01=0 (Set the count clock selection bit to 0 (operation clock specified the CKS01[1:0] bits).)
  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 operation mode for TAU01.)
  MD010=0 (Selects “no generation of INTTM01 at count start”.)
- **Timer channel start register 0 (TS0)**
  TS01=1 (Enables TAU01 operation.)
- **Timer data register 01 (TDR01)**
  OVF: Checks whether TAU01 overflowed or not.
- **Timer channel stop register 0 (TT0)**
  TT01: Stops the TAU01 operation.

10.3 Processing Example of Frequency Detection Function

Figure 10-1 is an example of using 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 10-1. Example of Using Frequency Detection Function

11. A/D Test Function

11.1 Overview of A/D Test Function

This A/D test function checks whether the A/D converter is operating normally by executing A/D conversions on three different internal voltages of 0V, the AV_{REF} voltage, and the internal reference voltage (1.45V).

11.2 Registers used for A/D Test Function

The registers used by the A/D test function is described below.

(1) A/D test register (ADTES)

This register selects the A/D conversion target. This register can be accessed by an 8-bit memory manipulation instruction.

Address: F0013H After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
ADTES	0	0	0	0	0	0	ADTES[1:0]	

Bit Name	Description
ADTES[1:0]	Selects an A/D conversion target. 00B: ANIx/ temperature sensor output/ internal reference voltage output (1.45V) This is specified by the ADS register. 01B: Setting prohibited 10B: AV_{REFM} 11B: AV_{REFP}

(2) Analog input channel specification register (ADS)

This register specifies the input 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.

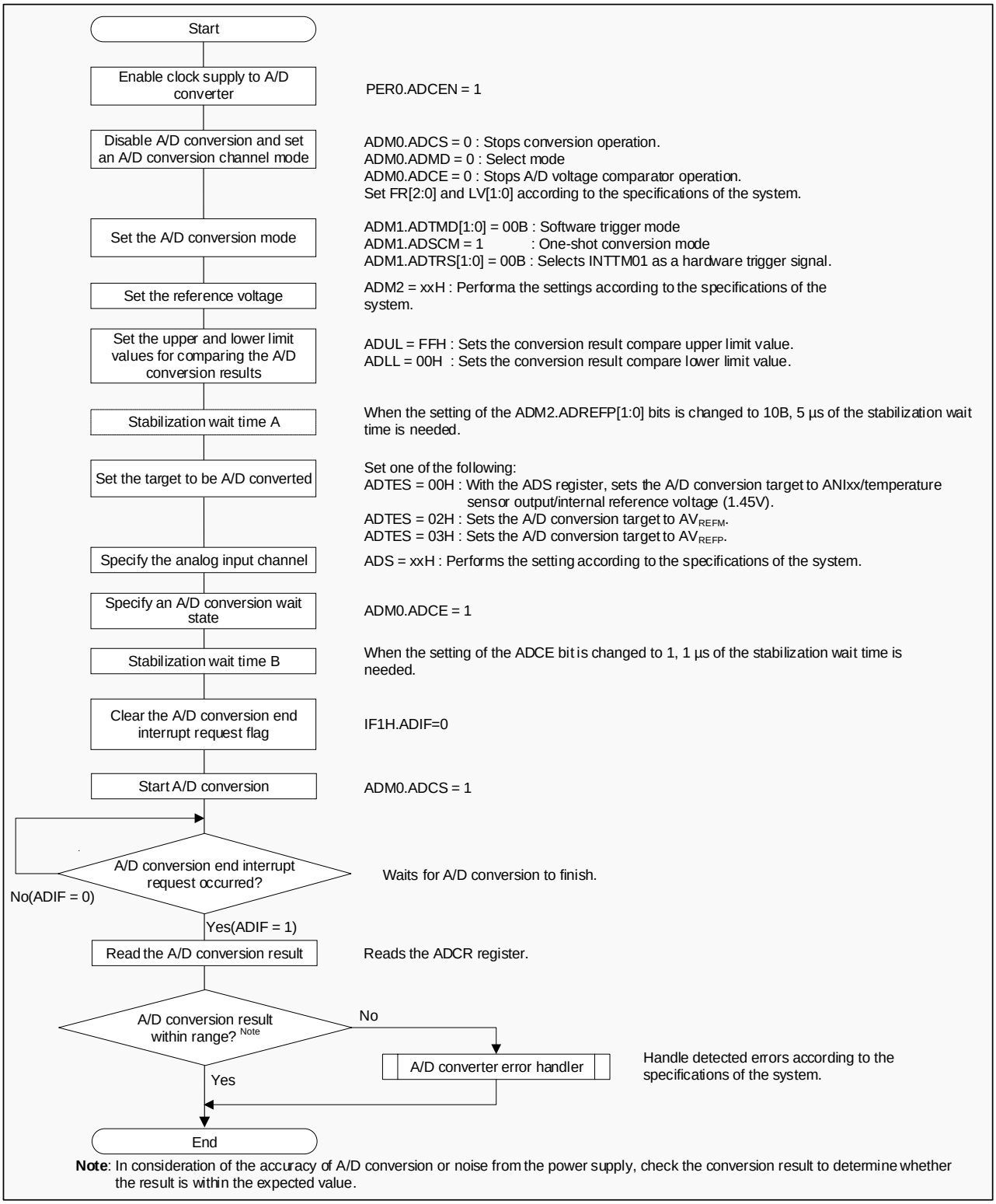
Address: FFF31H After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
ADS	ADISS	0	0	ADS[4:0]				

Bit Name	Description
ADISS	Selects an A/D conversion target. 0: Analog input channel (ANIx) 1: Temperature sensor output/ internal reference voltage output (1.45V)
ADS[4:0]	Channel selection ・Select mode (ADMD=0) ADISS=0 [When the analog input channel (ANIx) is selected]: 00000B (ANI0) to 11110B (ANI30), 11111B (Setting prohibited) ADISS=1 [When a temperature sensor or internal reference voltage output is selected]: 00000B (Temperature sensor output), 00001B (Internal reference voltage output), Other than 00000B and 00001B (Setting prohibited) ・Scan mode (ADMD=1) ADISS=0: 00000B (ANI0 to ANI3), …, 10100B (ANI20 to ANI23), Other than above (Setting prohibited) ADISS=1: Setting prohibited

11.3 Processing Example of A/D Test Function

Figure 11-1 is an example of using the A/D test function.



Note: In consideration of the accuracy of A/D conversion or noise from the power supply, check the conversion result to determine whether the result is within the expected value.

Figure 11-1. Example of Using A/D Test Function

11.4 Cautions when Using A/D Test Function

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

(1) When the ADISS bit in the ADS register is set to 1, the result of the first A/D conversion cannot be used.

(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.

12. I/O Port Output Signal Level Detection Function

12.1 Overview of I/O Port Output Signal Level Detection Function

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.

12.2 Registers used for I/O Port Output Signal Level Detection Function

The register used by the I/O port output signal level detection function is 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). 0: Reads the value (output latch) of Pm register (Pmn bit). 1: Reads the output level of a pin.

Remarks m : 0 to 16, n: 0 to 7

12.3 Processing Example of I/O Port Output Signal Level Detection Function

Figure 12-1 is an example of using the I/O port output signal level detection function.

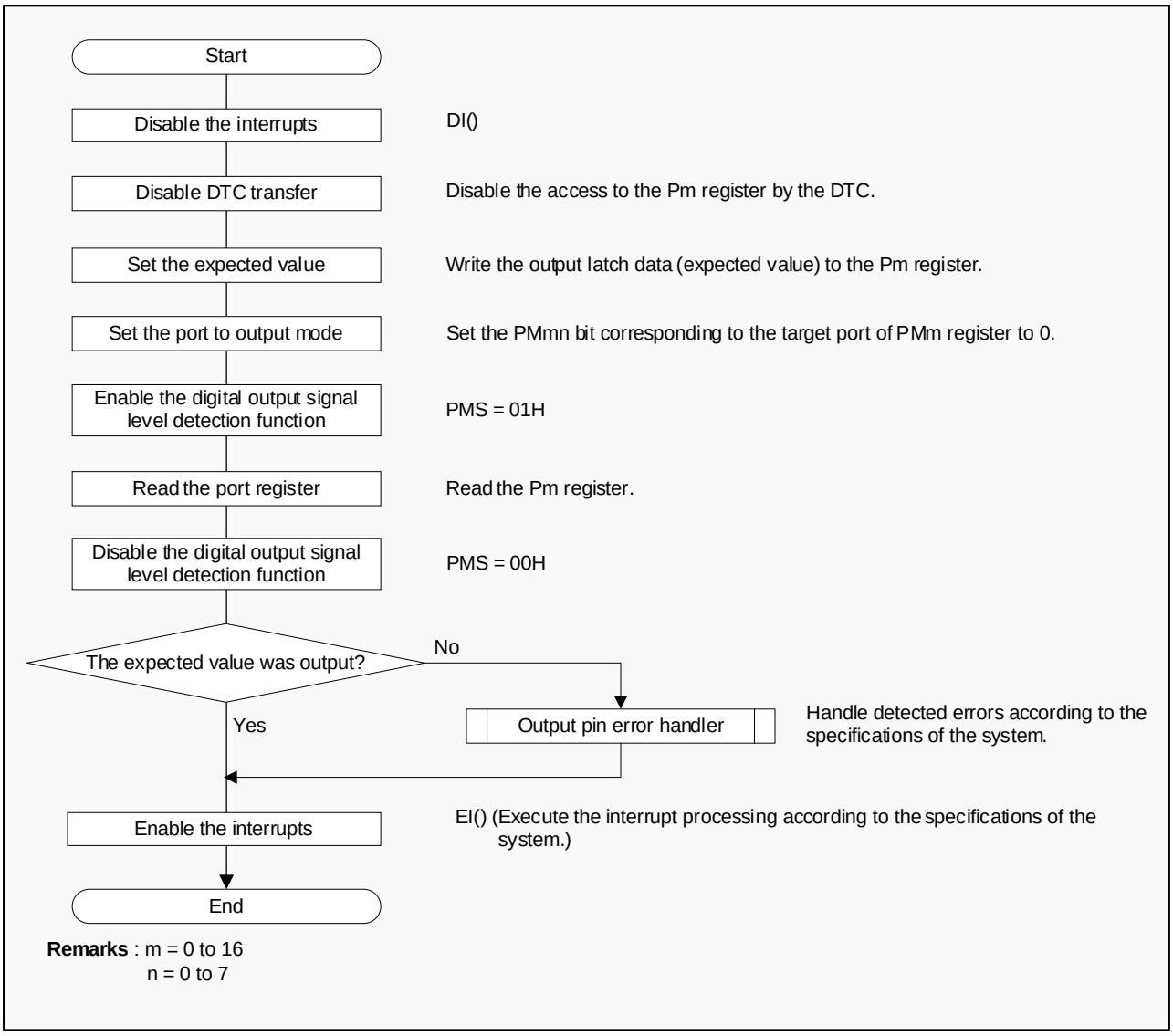


Figure 12-1. Example of using I/O Port Output Signal Level Detection Function

12.4 Cautions when using I/O Port Output Signal Level Detection Function

The following is the caution when using the I/O port output signal level detection function.

(1) When any writing is executed for a port register (Pm) 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.

Revision History

Rev.	Date	Description	
		Page	Summary
2.00	Jun.30,2017	−	First edition issued
2.10	Apr.26,2018	−	Revised flowcharts
2.20	Aug.31,2018	P.11, 16	Added Table 4-1 and Table 4-2
2.40	Feb.15,2022	−	Added description of RL78/F15 products
		P.19, 23	Added description of restrictions on the use of development tool

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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 October 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/.

© 2022 Renesas Electronics Corporation. All rights reserved.