RZ/T1 Group

Initial Settings

R01AN2554EJ0141
Rev.1.41
Jul. 13, 2018

Introduction

This application note explains a sample program that makes initial settings of RZ/T1 operating modes.

The major features of the sample program for initial settings are listed below

- The program has boot modules for each operating mode of the device: 16-bit bus boot version, SPI boot version and RAM execution version, which only uses the internal RAM (TCM) and does not require a flash memory.
- This sample program consists of two sections, the loader program and user application program and makes initial settings of the RZ/T1 in sequence after booting.
- The loader program, after booting RZ/T1, initializes the clock generator circuit and bus state controller, and copies the user application program.
- The user application program sets up ports, ECM functions and interrupts, and flashes LEDs.

Target Devices

RZ/T1 group

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation and testing of the modified program.

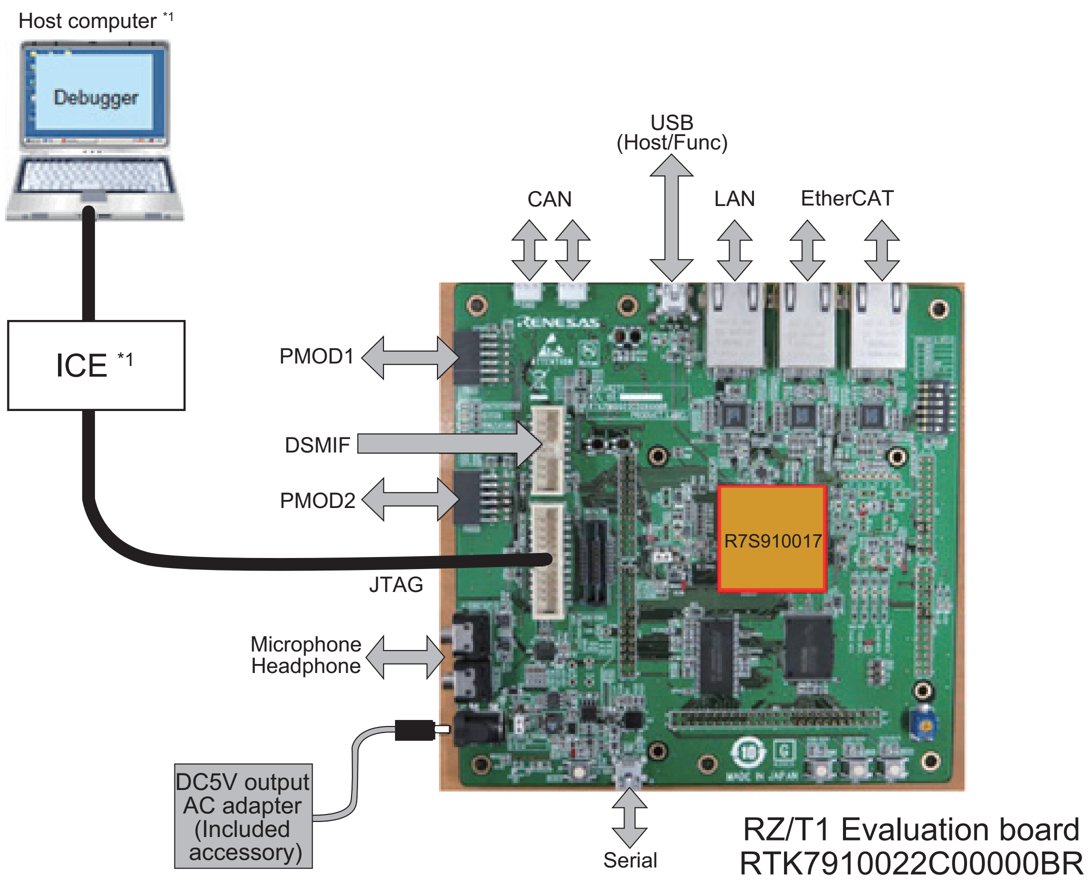
Table of Contents

1. Specifications

Table 1.1 lists the peripheral functions to be used and their applications and Figure 1.1 shows the operating environment.

Table 1.1 Peripheral Functions and Applications

Peripheral Function	Application
Clock generator (CPG)	Used as a CPU clock and low-speed on-chip oscillator
Interrupt controller (ICUA)	Used with external interrupt input pins (IRQ 5 and 12)
Bus state controller (BSC)	Used to attach NOR flash memory to CS0 and CS1 spaces and SDRAM to CS2 and CS3 spaces
SPI multi I/O bus controller (SPIBSC)	Used to attach serial flash memory to SPI multi I/O space
Error control module (ECM)	Used to initialize ERROROUT# pin and for extended pseudo error 35
Reset	Used to determine reset using reset status flag
General purpose I/O ports	Used to control pins to light LEDs on and off



Note 1. Indicates the device that the user needs to prepare.

Figure 1.1 Operating Environment

2. Operating Environment

The sample program covered in this application note is for the environment below.

Table 2.1 Operating Environment

Item	Description
Microcomputer	RZ/T1 Group
Operating frequency	CPUCLK = 450 MHz
Operating voltage	3.3 V
Integrated Development Environment	Embedded Workbench® for Arm, version 8.20.2 from IAR Systems DS-5™ 5.26.2 from Arm e2studio 6.1.0 from Renesas
Operating mode	SPI boot mode 16-bit bus boot mode
Board	RZ/T1 Evaluation Board (RTK7910022C00000BR)
Device (functions to be used on the board)	• NOR flash memory (connected to CS0 and CS1 spaces) Manufacturer: Macronix International Co., Ltd. Model: MX29GL512FLT2I-10Q • SDRAM (connected to CS2 and CS3 spaces) Manufacturer: Integrated Silicon Solution Inc. Model: IS42S16320D-7TL • Serial flash memory Manufacturer: Macronix International Co., Ltd. Model: MX25L51245G

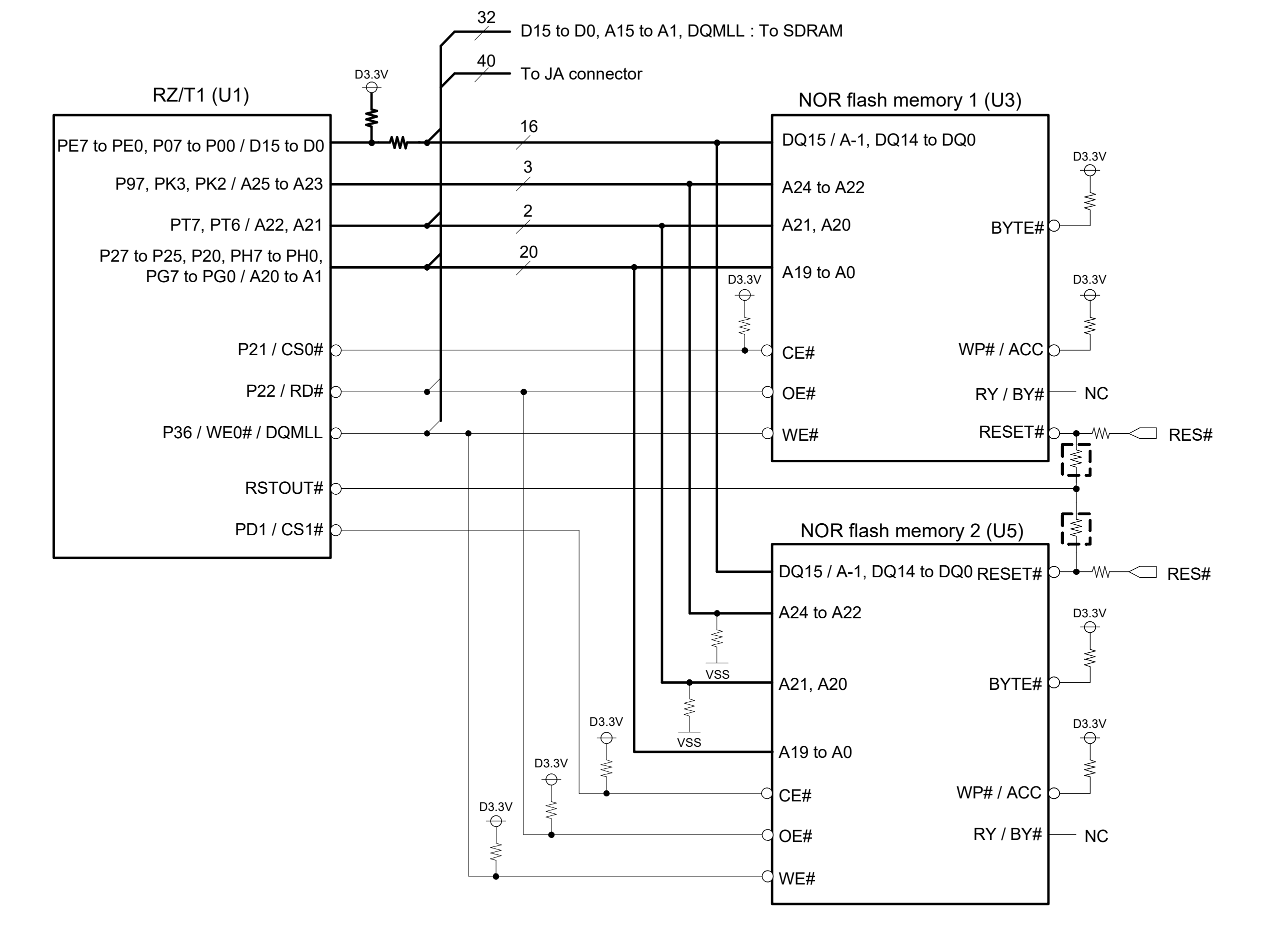
3. Peripheral Functions

See the RZ/T1 Group User's Manual: Hardware for basic descriptions for operating mode, clock generator (CPG), interrupt controller (ICUA), bus state controller (BSC), SPI multi I/O bus controller (SPIBSC), error control module (ECM), reset and general purpose I/O ports.

4. Hardware

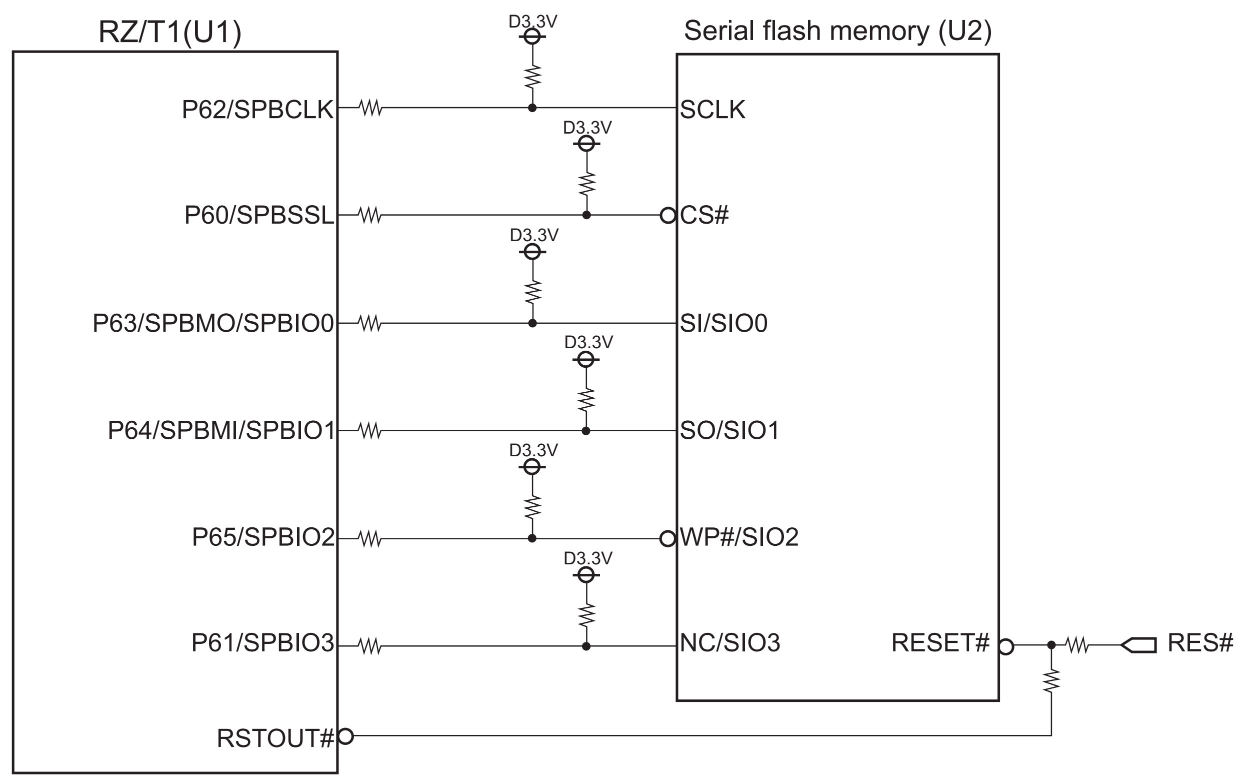
4.1 Hardware Configuration Examples

Figure 4.1 shows a hardware configuration example for 16-bit bus boot mode. Figure 4.2 shows a hardware configuration example for SPI boot mode.



Note: The mark "#" indicates negative logic (or active flow).

Figure 4.1 Hardware configuration example for 16-bit bus boot mode



Note: The mark “#” indicates negative logic (or active flow).

Figure 4.2 Hardware configuration example for SPI boot mode

4.2 Pins

Table 4.1 lists pins to be used and their functions.

Table 4.1 Pins and Functions

Pin Name	Input/Output	Function
A0 to A20	Output	Address signal output to NOR flash memory and SDRAM
D0 to D15	Input/Output	Data signal input and output to NOR flash memory and SDRAM
CS0#	Output	Device selection signal output to NOR flash memory attached to CS0 space
CS1#	Output	Device selection signal output to NOR flash memory attached to CS1 space
CS2#	Output	Device selection signal output to NOR flash memory attached to CS2 space
CS3#	Output	Device selection signal output to NOR flash memory attached to CS3 space
RAS#	Output	RAS# control signal output to SDRAM
CAS#	Output	CAS# control signal output to SDRAM
RD/WR#	Output	Read control signal or write control signal output to SDRAM
CKE	Output	Clock enabling control signal output to SDRAM
RD#	Output	Strobe signal output indicating reading
WE0#/DQMLL	Output	Write strobe signal output to D15 to D8
WE1#/DQMLU	Output	Write strobe signal output to D15 to D8
SPBSSL	Output	Slave select
SPBCLK	Output	Clock output
SPBMO/SPBIO0	Input/Output	Master output data / data 0
SPBMI/SPBIO1	Input/Output	Master input data / data 1
SPBIO2	Input/Output	Data 2
SPBIO3	Input/Output	Data 3
MD0	Input	Boot mode selection: MD0="L", MD1="L", MD2="L" (SPI boot mode) MD0="L", MD1="H", MD2="L" (16-bit bus boot mode)
MD1	Input	
MD2	Input	
IRQ5	Input	SW2 (IRQ pin interrupt)
IRQ12	Input	SW3 (IRQ pin interrupt)
PF7	Output	Lighting LED0 on and off
P56	Output	Lighting LED1 on and off
P77	Output	Lighting LED2 on and off
P53	Output	Lighting LED5 on and off

Note: The mark "#" indicates negative logic (or active low).

5. Software

This section explains the case of EWARM (from IAR systems) unless otherwise stated.

5.1 Operation Overview

After the reset is cancelled, the loader program for each operating mode (16-bit bus boot/SPI boot) stored on the external flash memory (NOR/Serial) is copied to the internal RAM (BTCM).

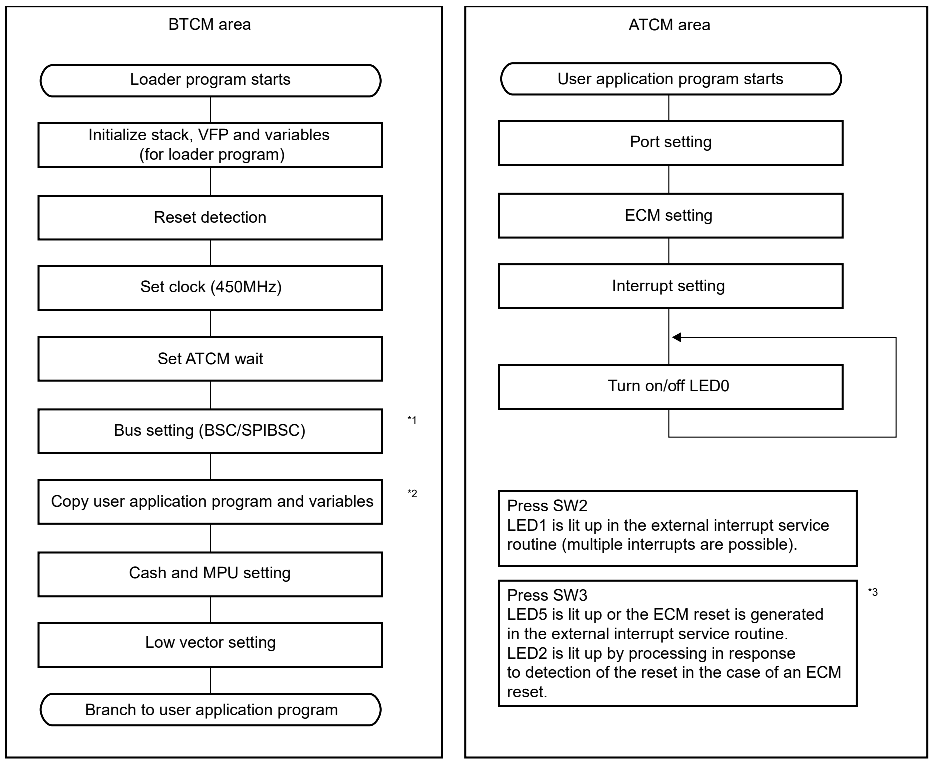
The loader program, after the boot processing, judges the device to have been reset, sets up the clock and bus, and copies the user application program stored in external flash memory (NOR/Serial) to the internal RAM (ATCM). Then, it sets the MPU and cache, switches the exception vector to the low vector state, and branches to the point where the user application program starts.

The user application program sets up ports, ECM function and external pin interrupts, and flashes LED0.

Pressing SW2 generates external pin interrupt 5 which lights up LED1 for about five seconds. Multiple interrupts can also be generated by pressing SW3 during this processing.

Pressing SW3 generates external pin interrupt 12 which switches LED5 on / off each time SW3 is pressed if LED1 is lit up at the time. When LED1 is switched off, pressing SW3 generates the ECM reset. After release from the reset state, judgment that the reset has occurred corresponds to detection of the ECM reset and LED2 lights up in response.

Figure 5.1 shows operation overview after boot process.


BTCM area
Loader program starts
Initialize stack, VFP and variables (for loader program)
Reset detection
Set clock (450MHz)
Set ATCM wait
Bus setting (BSC/SPIBSC)
*1
Copy user application program and variables
*2
Cash and MPU setting
Low vector setting
Branch to user application program
ATCM area
User application program starts
Port setting
ECM setting
Interrupt setting
Turn on/off LED0
Press SW2
LED1 is lit up in the external interrupt service routine (multiple interrupts are possible).
Press SW3
LED5 is lit up or the ECM reset is generated in the external interrupt service routine.
LED2 is lit up by processing in response to detection of the reset in the case of an ECM reset.
*3

Note 1. Not performed in the RAM execution version; SPIBSC setting only applies to SPI boot version.

Note 2. Not performed in the RAM execution version. The user application program is not copied.

Note 3. In the RAM execution version, if LED1 is not lit up, the ECM reset is not executed and the LED2 is lit up within the interrupt service routine.

Figure 5.1 Operation overview after boot process

5.1.1 Project Setting

The sample program for initial settings contains the following three types of projects.

(1) RZ_T1_init_boot ─┬ RZ_T1_init_nor_boot.eww: 16-bit bus boot version
└ RZ_T1_init_serial_boot.eww: SPI bus boot version

(2) RZ_T1_init_ram ── RZ_T1_init.eww: RAM execution version

(1) The project downloads the loader program and user application program for each operating mode (16-bit bus boot/SPI boot) to the external flash (NOR/Serial) and starts booting following the sequence described in Section 5.1, Operation Overview.

(2) The project directly downloads the loader program and user application program to the internal RAM (BTCM or ATCM) from EWARM, and executes them on the RAM.

The following table lists project settings used on the EWARM development environment.

Settings for which default values are used and do not require changes when making a new project are not listed.

Table 5.1 Project Settings of Sample Program (1 / 2)

General Option		
Target	Core	Cortex-R4F
	Device	Renesas R7S910017
	Endian mode	Little
	FPU	VFPv3
C/C++ Compiler		
Optimization	Level	Low
	Enabled transformations	Do not specify
Preprocessor	Ignore standard include directories	Do not select
	Additional include directories	$PROJ_DIR$¥inc
	Pre-include	Blank
	Defined symbols	Blank
	Preprocessor output to file	Do not select
Linker		
Config	Linker configuration file	Override default: Select 16-bit bus boot version: $PROJ_DIR$¥src¥common¥nor_boot¥RZ_T1_init_nor_boot.icf SPI boot version: $PROJ_DIR$¥src¥common¥serial_boot¥RZ_T1_init_serial_boot.icf RAM execution version: $PROJ_DIR$¥src¥common¥RZ_T1_init.icf
	Configuration file symbol definition	Blank
Library	Automatic runtime library selection	Select
	Additional libraries	Blank
	Override default program entry	Select loader_init1 Entry symbol: loader_init1
Output	Output files	$PROJ_FNAME$.out
	Generate debug information	Select

Table 5.1 Project Settings of Sample Program (2 / 2)

Debugger *1		
Setup	Driver	I-jet/JTAGjet
	Setup macros *2	Use macro file: Select 16-bit bus boot version/SPI boot version: $PROJ_DIR$¥RZT1_init_boot.mac RAM execution version: $PROJ_DIR$¥RZT1_init_RAM.mac
Download	Attach to program	Do not select
	Verify download	Select
	Suppress download	Do not select
	Use flash loader(s)	Select Override default board file: Select $TOOLKIT_DIR$¥config¥flashloader¥Renesas¥FlashRZT1.board

Note 1. This sample program is configured to use I-jet/JTAGjet as a default driver of the debugger. Modify the driver according to the debugger to be used.

Note 2. The macros are executed for initialization when connecting the debugger or flash loader.

RZT1_init_boot.mac: Initializes TMC and bus (BSC and SPIBSC) setting for each operating mode when starting debugging after download.

RZT1_init_RAM.mac: RAM execution version device downloads a program from EWARM directly to TCM instead of performing RZ/T1 boot process. Therefore this macro enables a non-maskable interrupt (FIQ) used for TCM initialization and boot process, and provides IRQ exception vector address with VIC.

See “Using C-SPY macros” in C-SPY Debugging Guide from IAR Systems for details about macros. See “Operating Modes” in RZ/T1 Group User’s Manual: Hardware for details about RZ/T1 boot process.

5.1.2 Preparation

Settings for SW4 on the RZ/T1 evaluation board (RTK7910022C00000BR) depend on the project to be used. Table 5.2 lists the settings of SW4. See the RZ/T1 Evaluation Board RTK7910022C00000BR User's Manual about each setting of the switch. The document is listed in Section 7, Related Documents.

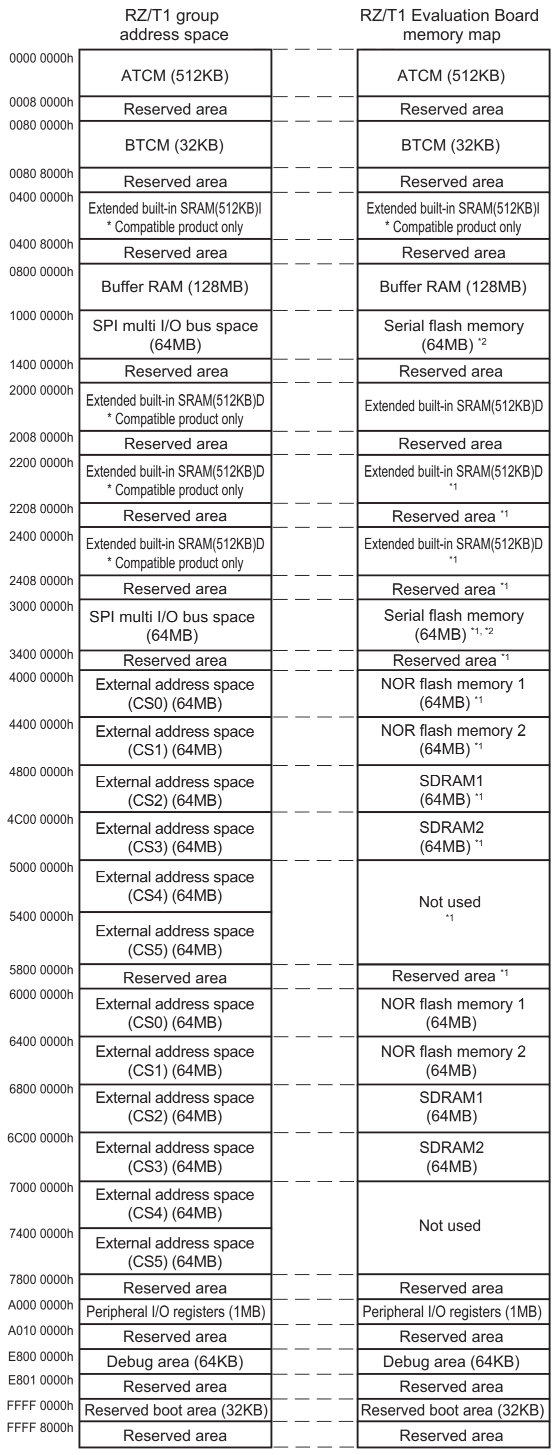
Table 5.2 SW4 Settings

Sample Program	SW4-1	SW4-2	SW4-3	SW4-4	SW4-5	SW4-6
16-bit bus boot mode version	ON	OFF	ON	ON	ON	OFF
SPI boot mode version	ON	ON	ON	ON	ON	OFF
RAM execution version	Either of the settings above					

For the usage procedures of sample programs in each of the available development environments, see Appendix 1. Supplementary Notes on Development Environments.

5.2 Memory Map

Figure 5.2 shows the address space of the RZ/T1 group and the memory map of the RZ/T1 Evaluation Board.



Address	RZ/T1 group address space	RZ/T1 Evaluation Board memory map
0000 0000h	ATCM (512KB)	ATCM (512KB)
0008 0000h	Reserved area	Reserved area
0080 0000h	BTCM (32KB)	BTCM (32KB)
0080 8000h	Reserved area	Reserved area
0400 0000h	Extended built-in SRAM(512KB)I * Compatible product only	Extended built-in SRAM(512KB)I * Compatible product only
0400 8000h	Reserved area	Reserved area
0800 0000h	Buffer RAM (128MB)	Buffer RAM (128MB)
1000 0000h	SPI multi I/O bus space (64MB)	Serial flash memory (64MB) *2
1400 0000h	Reserved area	Reserved area
2000 0000h	Extended built-in SRAM(512KB)D * Compatible product only	Extended built-in SRAM(512KB)D
2008 0000h	Reserved area	Reserved area
2200 0000h	Extended built-in SRAM(512KB)D * Compatible product only	Extended built-in SRAM(512KB)D *1
2208 0000h	Reserved area	Reserved area *1
2400 0000h	Extended built-in SRAM(512KB)D * Compatible product only	Extended built-in SRAM(512KB)D *1
2408 0000h	Reserved area	Reserved area *1
3000 0000h	SPI multi I/O bus space (64MB)	Serial flash memory (64MB) *1, *2
3400 0000h	Reserved area	Reserved area *1
4000 0000h	External address space (CS0) (64MB)	NOR flash memory 1 (64MB) *1
4400 0000h	External address space (CS1) (64MB)	NOR flash memory 2 (64MB) *1
4800 0000h	External address space (CS2) (64MB)	SDRAM1 (64MB) *1
4C00 0000h	External address space (CS3) (64MB)	SDRAM2 (64MB) *1
5000 0000h	External address space (CS4) (64MB)	Not used *1
5400 0000h	External address space (CS5) (64MB)	
5800 0000h	Reserved area	Reserved area *1
6000 0000h	External address space (CS0) (64MB)	NOR flash memory 1 (64MB)
6400 0000h	External address space (CS1) (64MB)	NOR flash memory 2 (64MB)
6800 0000h	External address space (CS2) (64MB)	SDRAM1 (64MB)
6C00 0000h	External address space (CS3) (64MB)	SDRAM2 (64MB)
7000 0000h	External address space (CS4) (64MB)	Not used
7400 0000h	External address space (CS5) (64MB)	
7800 0000h	Reserved area	Reserved area
A000 0000h	Peripheral I/O registers (1MB)	Peripheral I/O registers (1MB)
A010 0000h	Reserved area	Reserved area
E800 0000h	Debug area (64KB)	Debug area (64KB)
E801 0000h	Reserved area	Reserved area
FFFF 0000h	Reserved boot area (32KB)	Reserved boot area (32KB)
FFFF 8000h	Reserved area	Reserved area

Note 1. Mirror area.

Note 2. Used only by SPI boot mode sample program.

Figure 5.2 Memory Map

5.2.1 Section Assignment in Sample Program

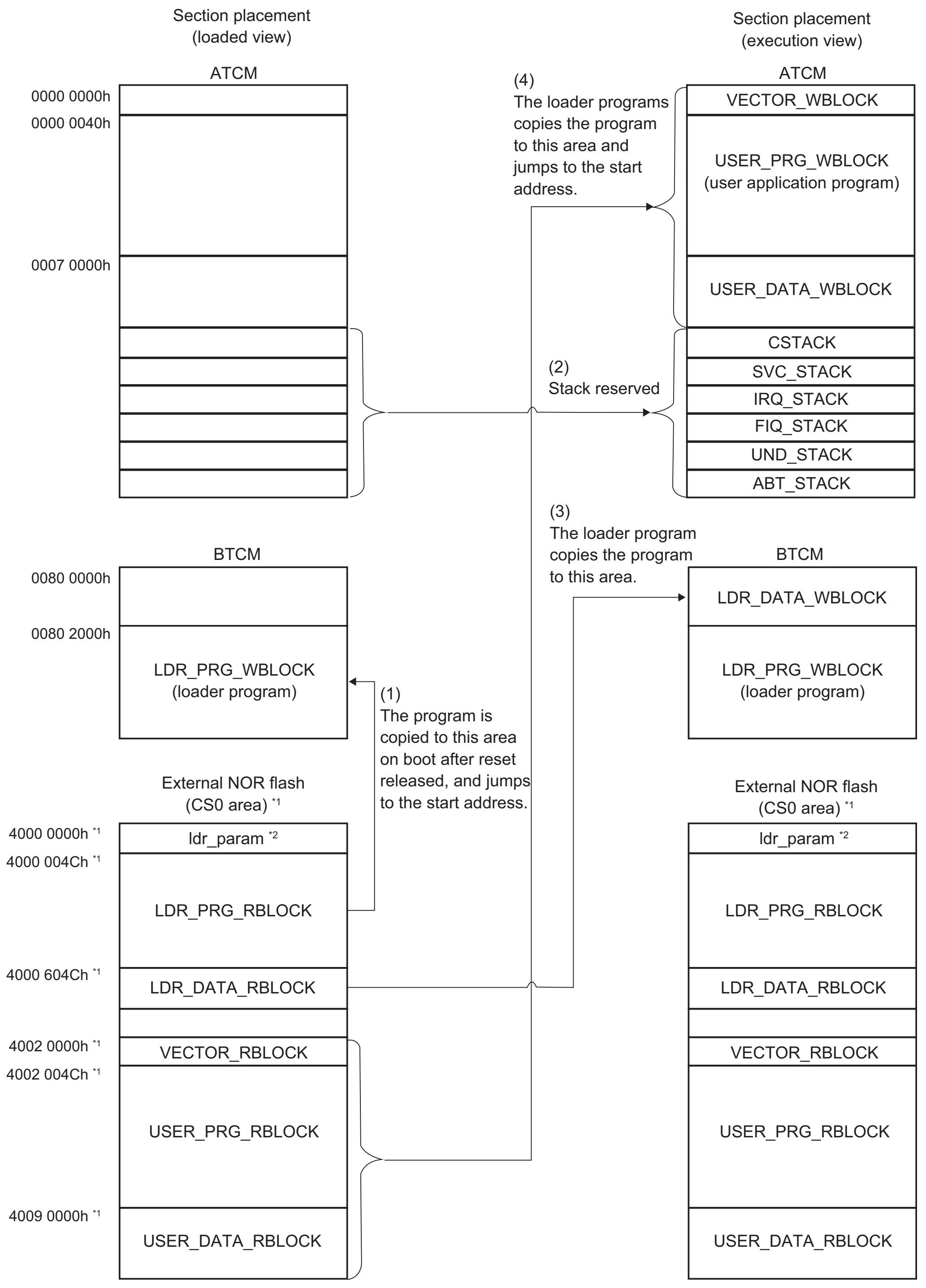
Table 5.3 lists the sections to be used in this sample program. Figure 5.3 shows the section assignment for 16-bit bus boot version example.

Table 5.3 Sections to Be Used

Area Name	Description	Type	Loading Area	Execution Area
VECTOR_WBLOCK	Reset and exception processing vector table	Code	—	ATCM
USER_PRG_WBLOCK	User application program area (for execution)	Code	—	ATCM
USER_DATA_WBLOCK	User application program variable area (for execution)	Data	—	ATCM
CSTACK	Stack area	Data	—	ATCM
SVC_STACK	Supervisor (SVC) mode stack area	Data	—	ATCM
IRQ_STACK	IRQ mode stack area	Data	—	ATCM
FIQ_STACK	FIQ mode stack area	Data	—	ATCM
UND_STACK	Undefined (UND) mode stack area	Data	—	ATCM
ABT_STACK	Abort (ABT) mode stack area	Data	—	ATCM
LDR_DATA_WBLOCK *3	Loader program variable area (for execution)	Data	—	BTCM
LDR_PRG_WBLOCK *3	Loader program area (for execution)	Code	—	BTCM
ldr_param	Loader parameters *1	Data	FLASH *2	—
LDR_PRG_RBLOCK *3	Loader program area (for storing)*1	Code	FLASH *2	—
LDR_DATA_RBLOCK *3	Loader program variable area (for storing) *1	Data	FLASH *2	—
VECTOR_RBLOCK	Reset and exception processing vector table area (for storing) *1	Code	FLASH *2	—
USER_PRG_RBLOCK	User application program area (for storing) *1	Code	FLASH *2	—
USER_DATA_RBLOCK	User application program variable area (for storing) *1	Data	FLASH *2	—

Note 1. The RAM execution version does not have this area.
Note 2. This is assigned to the NOR flash memory in the case of 16-bit bus boot version and to the serial flash memory in the case of SPI boot mode version.
Note 3. This sample program requires specification of functions (objects) used for the loader program in the loader program sections (LDR_PRG_xxx and LDR_DATA_xxx). To use an additional function in the loader program, specify the object in the linker configuration file (RZ_T1_init_xxx.icf).
Example: When you add r_func() (r_func.c) used in the loader program, add them in the appropriate section as shown below.

```
define block LDR_PRG_RBLOCK with fixed order
{
  ro code object loader_init.o,
  ro code object loader_init2.o,
  ro code object r_atcm_init.o,
  ro code object r_cpg.o,
  ro code object r_ram_init.o,
  ro code object r_mpc.o,
  ro code object bus_init_nor_boot.o,
  ro code object r_reset.o,
  ro code object r_func.o                  // Added line
}
```



Note 1. This corresponds to the serial flash (SPIBSC) area in case of SPI boot version.
Read addresses as if the start address is 1000 0000h.

Note 2. The loader parameters (in ldr_param section) are used in boot processing.
See the RZ/T1 User's Manual: Hardware for detail.

Figure 5.3 Section Assignment (for 16-bit bus boot version)

5.2.2 MPU Settings

Table 5.4 lists the MPU settings.

Table 5.4 MPU Settings

Contents	Address	Size	Memory Type
Extended built-in SRAM I	0400 0000H to 0407 FFFFH	512 KB	Area 0 Normal, cache disabled, shared
SPI multi I/O bus space	1000 0000H to 13FF FFFFH	64 MB	Area 1 Normal, cache disabled, shared
Extended built-in SRAM D and I mirror space	2000 0000H to 2800 0000H	128 MB	Area 2 Normal, cache disabled, shared
SPI multi I/O bus space mirror	3000 0000H to 33FF FFFFH	64 MB	Area 3 Normal, cache disabled, shared
CS0 and CS1 space mirror	4000 0000H to 47FF FFFFH	128 MB	Area 4 Normal, cache disabled, shared
CS2 and CS3 space mirror	4800 0000H to 4FFF FFFFH	128 MB	Area 5 Normal, cache disabled, shared
CS4 and CS5 space mirror	5000 0000H to 5FFF FFFFH	128 MB	Area 6 Device, shared, instruction fetch disabled
CS0 and CS1 space (NOR flash memory)	6000 0000H to 67FF FFFFH	128 MB	Area 7 Normal, cache disabled, shared
CS2 and CS3 space (SDRAM)	6800 0000H to 6FFF FFFFH	128 MB	Area 8 Normal, cache disabled, shared
CS4 and CS5 space	7000 0000H to 77FF FFFFH	128 MB	Area 9 Device, shared, instruction fetch disabled
Peripheral I/O register Area for debug Dedicated area for boot	8000 0000H to FFFF FFFFH	2 GB	Area 10 Device, shared, instruction fetch disabled

5.2.3 Exception Processing Vector Table

RZ/T1 has 7 types of exception processing (reset, undefined instruction, software interrupt, prefetch abort, Data abort, IRQ and FIQ exceptions) that are allocated to the 32-byte area starting from address 0000 0000H (address 0000 0000H to 0000 001F). Specify a branch instruction to each exception processing in the exception processing vector table.

Table 5.5 lists the contents of exceptional processing vector table for this sample program. Modify the setting to suit your need.

Table 5.5 Exception Processing Vector Table

Exception	Handler Address	Remark
RESER exception	0000 0000h	Branches to itself (to avoid branching to an unknown address)
Undefined instruction exception	0000 0004h	Branches to itself (user definable)
Software interrupt exception	0000 0008h	Branches to itself (user definable)
Prefetch abort exception	0000 000Ch	Branches to itself (user definable)
Data abort exception	0000 0010h	Branches to itself (user definable)
Reserved	0000 0014h	Branches to itself (user definable)
IRQ exception	0000 0018h	Branches to itself (to avoid branching to an unknown address)
FIQ exception	0000 001Ch	Branches to itself (user definable)

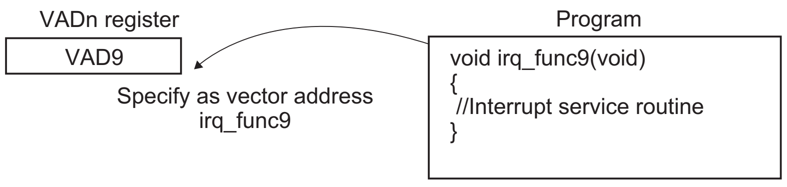
Note: When the boot processing finishes, it is in high vector state where SCTLR V[13] = 1 (FFFF 0000h). Use the loader program to write appropriate instructions in the low vectors (0000 0000h), and then move the state to low vector where V[13] = 0 (0000 0000h).

Interrupts

The RZ/T1 adopts the vector interrupt controller (VIC) to control interrupts for Cortex-R4F and accepts interrupts as follows:
FIQ interrupt: Always accepted as non-maskable interrupts from the NMI pin or ECM.
IRQ interrupt: Accepted as interrupts from external pins or on-chip peripheral modules except non-maskable interrupts.

Specifying the interrupt address storage register (VADn where n is the interrupt vector number) in advance enables fast exception processing because processing directly branches to the specified address when an IRQ interrupt is accepted.

Example: When specifying irq_func9() for IRQ9 interrupt service routine



See the "Interrupt Controller" section in the RZ/T1 Group User's Manual: Hardware for details.

5.3 Interrupts

Table 5.6 lists interrupts used in the sample program.

Table 5.6 Interrupts Used in the Sample Program

Interrupt (Source ID)	Priority	Process Outline
IRQ pin interrupt 5 (IRQ9)	15	Pressing SW2 lights LED1 up for about five seconds. Multiple interrupts are possible.
IRQ pin interrupt 12 (IRQ16)	10	Processing in response to pressing SW3 depends on the state of LED1. When LED1 is lit up, LED5 is switched on / off each time SW3 is pressed. When LED1 is not lit up, pressing SW3 generates the ECM reset by producing an extended pseudo error. After release from the reset state, LED2 is lit up in response to judgment of the reset.

5.4 Fixed-Width Integer Types

Table 5.7 lists fixed-width integers used in the sample program.

Table 5.7 Fixed-width Integers Used in the Sample Program

Symbol	Description
int8_t	8-bit signed integer (defined in the standard library)
int16_t	16-bit signed integer (defined in the standard library)
int32_t	32-bit signed integer (defined in the standard library)
int64_t	64-bit signed integer (defined in the standard library)
uint8_t	8-bit unsigned integer (defined in the standard library)
uint16_t	16-bit unsigned integer (defined in the standard library)
uint32_t	32-bit unsigned integer (defined in the standard library)
uint64_t	64-bit unsigned integer (defined in the standard library)

5.5 Constants/Error Codes

Table 5.8 lists constants to be used in the sample program.

Table 5.8 Constants to be Used in the Sample Program (1 / 2)

Constant Name	Setting Value	Description
ATCM_WAIT_1_OPT	(0)	ATCM wait control: 1-wait with optimization
ATCM_WAIT_1	(1)	ATCM wait control: 1-wait without optimization
ATCM_WAIT_0	(2)	ATCM wait control: 0-wait
CPG_CPUCLK_150_MHz	(0)	CPU clock frequency: 150 MHz
CPG_CPUCLK_300_MHz	(1)	CPU clock frequency: 300 MHz
CPG_CPUCLK_450_MHz	(2)	CPU clock frequency: 450 MHz
CPG_CPUCLK_600_MHz	(3)	CPU clock frequency: 600 MHz
CPG_PLL1_OFF	(0)	CPG PLL1 operation control: Stop
CPG_PLL1_ON	(1)	CPG PLL1 operation control: Run
CPG_SELECT_PLL0	(0)	System clock source selection: PLL0
CPG_SELECT_PLL1	(1)	System clock source selection: PLL1
CPG_CKIO_75_MHz	(0)	External bus clock selection: 75 MHz
CPG_CKIO_50_MHz	(1)	External bus clock selection: 50 MHz
CPG_CKIO_37_5_MHz	(2)	External bus clock selection: 37.5 MHz
CPG_CKIO_30_MHz	(3)	External bus clock selection: 30 MHz
CPG_CKIO_25_MHz	(4)	External bus clock selection: 25 MHz
CPG_CKIO_21_43_MHz	(5)	External bus clock selection: 21.43 MHz
CPG_CKIO_18_75_MHz	(6)	External bus clock selection: 18.75 MHz
CPG_LOCO_ENABLE	(0x00000000)	Low-speed on-chip oscillator control: Run
CPG_LOCO_DISABLE	(0x00000001)	Low-speed on-chip oscillator control: Stop
ECM_COMMAND_KEY	(0x000000A5)	ECM Write protected register release key
ICU_EXT_PIN_0 to ICU_EXT_PIN_15	(0) to (15)	IRQ pin interrupt number: 0 to 15
ICU_DETECT_LOW	(0x00)	IRQ detection sense selection: Low level
ICU_DETECT_FALL	(0x04)	IRQ detection sense selection: Falling edge
ICU_DETECT_RISE	(0x08)	IRQ detection sense selection: Rising edge
ICU_DETECT_RISE_FALL	(0x0C)	IRQ detection sense selection: Both edges
ICU_DNF_DIVISION_1	(0)	IRQ digital noise filter setting: Used for the sampling clock PCLKB
ICU_DNF_DIVISION_8	(1)	IRQ digital noise filter setting: Used for the sampling clock PCLKB / 8
ICU_DNF_DIVISION_32	(2)	IRQ digital noise filter setting: Used for the sampling clock PCLKB / 32
ICU_DNF_DIVISION_64	(3)	IRQ digital noise filter setting: Used for the sampling clock PCLKB / 64
ICU_DNF_NO_USE	(4)	IRQ digital noise filter setting: Digital noise filter disabled
ICU_VEC_NUM_1 to ICU_VEC_NUM_300	(0) to (300)	Interrupt vector number: 0 to 300
ICU_TYPE_LEVEL	(0)	Interrupt input detection type: Level detection
ICU_TYPE_EDGE	(1)	Interrupt input detection type: Edge detection
ICU_PRIORITY_0 to ICU_PRIORITY_15	(0) to (15)	Interrupt priority level: 0 to 15
ICU_IEC_MASK_SET	(1)	Interrupt request clear: Interrupt mask (disabled) state. Clears the corre-sponding bit in the IEN register
ICU_PIC_EDGE_CLEAR	(1)	Edge detection clear: Clears edge detection

Table 5.8 Constants to be Used in the Sample Program (2 / 2)

Constant Name	Setting Value	Description
MPC_IRQ_DISABLE	(0)	Interrupt input function selection: Do not use as IRQ input pin
MPC_IRQ_ENABLE	(1)	Interrupt input function selection: Used as IRQ input pin
PORT_DIRECTION_HIZ	(0)	PORT direction control: Non-use (Hi-z input protection)
PORT_DIRECTION_INPUT	(2)	PORT direction control: Input (works as an input port)
PORT_DIRECTION_OUTPUT	(3)	PORT direction control: Output (works as an output port, port read enabled)
PORT_OUTPUT_LOW	(0)	PORT output data: Low output
PORT_OUTPUT_HIGH	(1)	PORT output data: High output
PORT_MODE_GENERAL	(0)	PORT pin mode control: Used as a general input/output port
PORT_MODE_PERIPHERAL	(1)	PORT pin mode control: Works as peripheral function
PORT_PULL_UPDOWN_DISABLE	(0)	PORT input pull-up/pull-down resistor control: Input pull-up resistor/pull-down resistor disabled
PORT_PULL_DOWN	(1)	PORT input pull-up/pull-down resistor control: Input pull-down resistor enabled
PORT_PULL_UP	(2)	PORT input pull-up/pull-down resistor control: Input pull-up resistor enabled
PORT_P10_NORMAL_DRIVE	(0)	P10 driving ability control: Normal output
PORT_P10_HIGH_DRIVE	(1)	P10 driving ability control: High-drive output
RST_SOURCE_RES	(0x00000002)	RES# pin reset detect flag: RES# pin reset not detected
RST_SOURCE_ECM	(0x00000004)	ECM reset detect flag: Detects ECM reset
RST_SOURCE_SWR1	(0x00000008)	Software reset detect flag: Software reset detected

5.6 Structures/Unions/Enumerated Types

Figure 5.4 and Figure 5.5 list Structures/Unions/Enumerated Types to be used in the Sample Program.

```
typedef enum
{
    ECM_MASTER,
    ECM_CHECKER,
    ECM_COMMON,
    ECM_TYPE_MAX
} ecm_reg_type_t;

typedef struct
{
    uint32_t cdb;
    uint32_t ocdb;
    uint32_t adb;
    uint32_t opdb;
    uint32_t spidb;
    uint32_t cde;
    uint32_t ocde;
    uint32_t ade;
    uint32_t opde;
    uint32_t spide;
    uint32_t sslkp;
    uint32_t spire;
    uint32_t spiwe;

    uint32_t dme;        /* Dummy cycle enable at the time of a SPI mode */
    uint32_t addre;    /* Address DDR enable                              */
    uint32_t opdre;    /* Option data DDRenable                           */
    uint32_t spidre;  /* Transmission data DDR enable                    */

    uint32_t dmdb;       /* The dummy cycle bit width of the time of a SPI mode      */
    uint32_t dmcyc;      /* The number of dummy cycles of the time of a SPI mode */

    uint32_t cmd;
    uint32_t ocmd;
    uint32_t addr;
    uint32_t opd[4];
    uint32_t smrdr;
    uint32_t smwdr;

} st_spibsc_spimd_reg_t;
```

Figure 5.4 Structures/Unions/Enumerated Types to be Used in the Sample Program

```
typedef struct
{
        uint32_t udef_cmd;
        uint32_t udef_cmd_width;
        uint32_t udef_opd3;
        uint32_t udef_opd2;
        uint32_t udef_opd1;
        uint32_t udef_opd0;
        uint32_t udef_opd_enable;
        uint32_t udef_opd_width;
        uint32_t udef_dmycyc_num;
        uint32_t udef_dmycyc_enable;
        uint32_t udef_dmycyc_width;
        uint32_t udef_data_width;
        uint32_t udef_spbr;
        uint32_t udef_brdv;
        uint32_t udef_addr_width;
        uint32_t udef_addr_mode;

} st_spibsc_cfg_t;
```

Figure 5.5 Structures/Unions/Enumerated Types Used to be used in the Sample Program

5.7 Global Variables

Table 5.9 lists global variables.

Table 5.9 Global variables

Type	Variable Name	Description	Function
static uint32_t *	g_pcmd_reg_adrr[]	ECM master/checker protection command register	R_ECM_Write_Reg8 R_ECM_Write_Reg32

5.8 Functions

Table 5.10 lists functions.

Table 5.10 Functions

Function Name	Page Number
loader_init1	25
loader_init2	25
reset_check	25
cpg_init	26
bsc_init	26
copy_to_atcm	26
cache_init	27
set_low_vec	27
main	27
ecm_init	28
icu_init	28
atcm_waitset	28
cpg_pll_wait	29
R_ECM_Init	29
R_ICU_Enable	29
R_ICU_Disable	30
R_ICU_ExtPinInit	30
R_ICU_Regist	31
Userdef_SFLASH_Set_Mode	31
Userdef_SFLASH_Write_Enable	32
Userdef_SFLASH_Busy_Wait	32
sflash_exmode_init	32
sflash_exmode_setting	33
sflash_wait_tend	33
sflash_set_config	34
SPIBSC_Exread_Mode_Config	34
R_IRQ9_isr	35
R_IRQ16_isr	35

5.9 Specifications of Functions

Specifications of the functions used in the sample program are as follows:

5.9.1 loader_init1

loader_init1	
Synopsis	Loader program 1
Declaration	loader_init1
Description	Initializes stacks, VFP (FPU) and loader program variables with initial values, and branches to the loader_init2 function.
Arguments	None
Return values	None
Remarks	None

5.9.2 loader_init2

loader_init2	
Synopsis	Loader program 2
Declaration	loader_init2
Description	Make initial settings of the CPU, clock and bus, transfers the application program to ATCM and branches to the main process.
Arguments	None
Return values	None
Remarks	None

5.9.3 reset_check

reset_check	
Synopsis	Reset flag detection processing
Declaration	void reset_check(void)
Description	Checks a reset cause and performs each sequence. Sets output of P77 pin to "H" (to light on LED2) when ECM reset occurs and the error cause number is 35.
Arguments	None
Return values	None
Remarks	None

5.9.4 cpg_init

cpg_init	
Synopsis	Clock setting processing
Declaration	void cpg_init(void)
Description	Uses PLL1 to set the CPU clock to 450 MHz. Runs the low-speed on-chip oscillator.
Arguments	None
Return values	None
Remarks	None

5.9.5 bsc_init

bsc_init	
Synopsis	Bus setting processing
Declaration	void bus_init (void)
Description	Makes settings for connection to the external memory (NOR flash memory, serial flash memory or SDRAM).
Arguments	None
Return values	None
Remarks	Serial flash memory can be set only on the SPI boot version. This process is not performed on the RAM execution version.

5.9.6 copy_to_atcm

copy_to_atcm	
Synopsis	Copy processing of the user application program
Declaration	void copy_to_atcm(void)
Description	Copies the user application program from the external flash memory (NOR/Serial) to the internal RAM (ATCM) in 4-byte unit. Also copies exception vectors and user application variables with initial values.
Arguments	None
Return values	None
Remarks	This process is not performed on the RAM execution version.

5.9.7 cache_init

cache_init	
Synopsis	Setting processing for MPU and cache
Declaration	void cache_init(void)
Description	Disables all entries of I1 and D1 caches and sets MPU undefined areas to the default memory map. After specifying each area mentioned in Table 5.4 MPU Settings, enables MPU and selects enabling of I1 and D1 caches enabled.
Arguments	None
Return values	None
Remarks	None

5.9.8 set_low_vec

set_low_vec	
Synopsis	Low vector setting processing
Declaration	void set_low_vec (void)
Description	Switches the exception vector table from high vector to low vector.
Arguments	None
Return values	None
Remarks	None

5.9.9 main

main	
Synopsis	Main processing
Declaration	int main (void)
Description	This is the user application program.
Arguments	None
Return values	None
Remarks	None

5.9.10 port_init

port_init	
Synopsis	Port setting processing
Declaration	void port_init(void)
Description	Configures the PF7, P56, and P53 ports for low output.
Arguments	None
Return values	None
Remarks	None

5.9.11 ecm_init

ecm_init	
Synopsis	ECM initialization processing
Declaration	void ecm_init(void)
Description	Initializes the ECM function
Arguments	None
Return values	None
Remarks	None

5.9.12 icu_init

icu_init	
Synopsis	Interrupt setting processing
Declaration	void icu_init(void)
Description	Configures PN5 (IRQ5) and P44 (IRQ12) ports as external interrupt input pins and enables interrupts.
Arguments	None
Return values	None
Remarks	None

5.9.13 atcm_waitset

atcm_waitset	
Synopsis	ATCM access wait setting processing
Header	r_atcm_init.h
Declaration	void atcm_waitset(uint32_t atcm_wait)
Description	Sets a wait for access to the ATCM.
Arguments	uint32_t atcm_wait
Return values	None
Remarks	Call this setting function from a program assigned to a memory area outside ATCM to avoid fetch access from the CPU.

5.9.14 cpg_pll_wait

cpg_pll_wait	
Synopsis	PLL stability waiting (100us) processing
Header	r_cpg.h
Declaration	void cpg_pll_wait(void)
Description	Uses CMT0 to wait 100us while PLL1 oscillation becomes stable.
Arguments	None
Return values	None
Remarks	Note that CMT0 is initialized or stopped in this function. Be careful when you use CMT0.

5.9.15 R_ECM_Init

R_ECM_Init	
Synopsis	ECM function initialization processing
Header	r_ecm.h
Declaration	void R_ECM_Init(void)
Description	Clears all error causes and sets the ERROROUT# pin output inactive (High).
Arguments	None
Return values	None
Remarks	Note that CMT0 is initialized or stopped in this function. Be careful when you use CMT0.

5.9.16 R_ICU_Enable

R_ICU_Enable		
Synopsis	Interrupt enabling setting	
Header	r_icu_init.h	
Declaration	void R_ICU_Enable(uint32_t vec_num)	
Description	Calculates the value of the interrupt enable register from an interrupt vector number as the argument and enables an interrupt corresponding to the vector number.	
Arguments	uint32_t vec_num	Specify an interrupt vector number. Setting range: 1 to 300
Return values	None	
Remarks	None	

5.9.17 R_ICU_Disable

R_ICU_Disable		
Synopsis	Interrupt disabling setting	
Header	r_icu_init.h	
Declaration	void R_ICU_Disable(uint32_t vec_num)	
Description	Calculates the value of the interrupt enable clear register from an interrupt vector number as the argument and disables an interrupt corresponding to the vector number.	
Arguments	uint32_t vec_num	Specify an interrupt vector number. Setting range: 1 to 300
Return values	None	
Remarks	None	

5.9.18 R_ICU_ExtPinInit

R_ICU_ExtPinInit		
Synopsis	External interrupt setting processing	
Header	r_icu_init.h	
Declaration	void R_ICU_ExtPinInit(uint16_t pin_num, uint8_t detect, uint32_t dnf_set)	
Description	Configures an external interrupt pin according to the setting passed as an argument.	
Arguments	uint16_t pin_num	Specify a pin number for setting external interrupt. Setting range: 0 to 15
	uint8_t detect	Specify a detection method. Low level, falling edge, rising edge and both edges
	uint32_t dnf_set	Enable or disable digital noise filtering. If it is enabled, the number of sampling clock cycles is specified.
Return values	None	
Remarks	None	

5.9.19 R_ICU_Regist

R_ICU_Regist		
Synopsis	Registration to interrupt controller	
Header	r_icu_init.h	
Declaration	void R_ICU_Regist(uint32_t vec_num, uint32_t type, uint32_t priority, uint32_t isr_addr)	
Description	Configures the interrupt controller according to the setting passed as an argument.	
Arguments	uint32_t vec_num	Specify an interrupt vector number. Setting range: 1 to 300
	uint32_t type	Interrupt detection type
	uint32_t priority	Specify an interrupt priority level. Specify a vector number 1 to 255 for interrupt priority level 0 to 15 and 256 to 300 for interrupt priority level 16 to 31.
	uint32_t isr_addr	Specify a starting address of interrupt service routine.
Return values	None	
Remarks	None	

5.9.20 Userdef_SFLASH_Set_Mode

Userdef_SFLASH_Set_Mode		
Synopsis	Setting registers in serial flash memory (user defined)	
Declaration	int32_t Userdef_SFLASH_Set_Mode(uint32_t data_width, uint32_t addr_mode)	
Description	Implement a process to set up a required register of a serial flash memory according to the serial flash memory used in this function when you use SPIBSC in external address space read mode.	
Arguments	uint32_t data_width	Data read bit size Data read bit size of serial flash memory for converting a read access to the SPI multi I/O bus space into SPI communication. SPIBSC_1BIT: 1-bit size SPIBSC_4BIT: 4-bit size
	uint32_t addr_mode	Address mode setting Specifies an address to output a read access to serial flash memory when converting into the SPI multi I/O bus space. SPIBSC_OUTPUT_ADDR_24: 24-bit address output SPIBSC_OUTPUT_ADDR_32: 32-bit address output
Return values	0: Success -1: Failure	
Remarks	This function is not executed in 16-bit bus boot mode version and RAM execution version.	

5.9.21 Userdef_SFLASH_Write_Enable

Userdef_SFLASH_Write_Enable		
Synopsis	Setting for enabling serial flash memory write (user defined)	
Declaration	int32_t Userdef_SFLASH_Write_Enable(void)	
Description	Implement a process to set up and write enable a required register of a serial flash memory according to the serial flash memory used in this function.	
Arguments	None	
Return values	0: Success -1: Failure	
Remarks	This function is not executed in 16-bit bus boot mode version and RAM execution version.	

5.9.22 Userdef_SFLASH_Busy_Wait

Userdef_SFLASH_Busy_Wait		
Synopsis	Wait processing for serial flash memory to become ready (user defined)	
Declaration	int32_t Userdef_SFLASH_Busy_Wait(uint32_t data_width)	
Description	Implement a process to read a register of a serial flash memory according to the serial flash memory used in this function and make the serial flash memory transit to ready state.	
Arguments	uint32_t data_width	Data read bit size Data read bit size of serial flash memory for converting a read access to the SPI multi I/O bus space into SPI communication. SPIBSC_1BIT: 1-bit size SPIBSC_4BIT: 4-bit size
Return values	None	
Remarks	This function is not executed in 16-bit bus boot mode version and RAM execution version.	

5.9.23 sflash_exmode_init

sflash_exmode_init		
Synopsis	SPIBSC external address mode initial settings	
Header	r_spibsc_ioset_api.h	
Declaration	int32_t sflash_exmode_init(st_spibsc_cfg_t *spibsccfg)	
Description	Makes necessary initial settings to use SPIBC in external address space read mode. After the initial settings, set it to external address space read mode.	
Arguments	st_spibsc_cfg_t *spibsccfg	Setting for SPIBSC external address space read
Return values	0: Normal termination -1: Error	
Remarks	This function is not executed in 16-bit bus boot mode version and RAM execution version.	

5.9.24 sflash_exmode_setting

sflash_exmode_setting	
Synopsis	SPIBSC initial settings
Header	r_spibsc_ioset_api.h
Declaration	int32_t sflash_exmode_setting(st_spibsc_cfg_t *spibsccfg)
Description	Configures SPIBSC for controlling serial flash memory and to use SPIBSC in external address space read mode. Also performs register settings on serial flash memory according to initial settings. After the initial settings, sets it to external address space read mode. Executes the SPIBSC external address mode initial setting function (sflash_exmode_init) in this function.
Arguments	st_spibsc_cfg_t *spibsccfg Setting for SPIBSC external address space read
Return values	0: Normal termination -1: Error
Remarks	This function is not executed in 16-bit bus boot mode version and RAM execution version.

5.9.25 sflash_wait_tend

sflash_wait_tend	
Synopsis	SPIBSC data transfer waiting processing
Header	r_spibsc_ioset_api.h
Declaration	void sflash_wait_tend(void)
Description	Waits for data transfer from SPIBSC to finish
Arguments	None
Return values	None
Remarks	This function is not executed in 16-bit bus boot mode version and RAM execution version.

5.9.26 sflash_set_config

sflash_set_config	
Synopsis	SPIBSC external address mode setting processing
Header	r_spibsc_ioset_api.h
Declaration	int32_t sflash_set_config(st_spibsc_cfg_t *spibsccfg)
Description	Determines settings to use SPIBSC in external address space read mode according to a serial flash memory to be used. Executes a user defined function (SPIBSC external address space read setting function: Userdef_SPIBSC_Set_Config) in this function.
Arguments	st_spibsc_cfg_t *spibsccfg SPIBSC external address space read setting
Return values	0: Normal termination -1: Error
Remarks	This function is not executed in 16-bit bus boot mode version and RAM execution version.

5.9.27 SPIBSC_Exread_Mode_Config

SPIBSC_Exread_Mode_Config	
Synopsis	SPIBSC configuration processing
Declaration	static int32_t SPIBSC_Exread_Mode_Config(st_spibsc_cfg_t *spibsccfg)
Description	Checks ranges of each data passed with argument
Arguments	st_spibsc_cfg_t *spibsccfg SPIBSC external address space read setting
Return values	0: Success -1: Failure
Remarks	This function is not executed in 16-bit bus boot mode version and RAM execution version.

5.9.28 Userdef_SPIBSC_Set_Config

Userdef_SPIBSC_Set_Config	
Synopsis	SPIBSC external address space read setting processing (user defined)
Declaration	void Userdef_SPIBSC_Set_Config(st_spibsc_cfg_t *spibsccfg)
Description	Determines settings of SPIBSC external address space read mode according to a serial flash memory to be used. Use this function for make necessary initial settings to use SPIBSC in external address space read mode in an area specified with argument spibsccfg.
Arguments	st_spibsc_cfg_t *spibsccfg SPIBSC external address space read setting
Return values	0: Normal termination -1: Error
Remarks	This function is not executed in 16-bit bus boot mode version and RAM execution version.

5.9.29 R_IRQ9_isr

R_IRQ9_isr	
Synopsis	IRQ9 interrupt (IRQ pin interrupt 5) processing
Declaration	void R_IRQ9_isr(void)
Description	LED1 is switched on for about five seconds. Multiple interrupts are possible.
Arguments	None
Return values	None
Remarks	None

5.9.30 R_IRQ16_isr

R_IRQ16_isr	
Synopsis	IRQ16 interrupt (IRQ pin interrupt 12) processing
Declaration	void R_IRQ16_isr(void)
Description	The output state of LED5 is inverted if LED1 is currently lit up. Extended pseudo error 35 is produced and an ECM reset is generated by that if LED1 is not currently lit up.
Arguments	None
Return values	None
Remarks	The ECM reset is not generated and LED2 is lit up in the RAM execution version.

5.10 Flowchart

5.10.1 Loader Program 1 Processing

Figure 5.6 shows the loader program 1 flow chart.

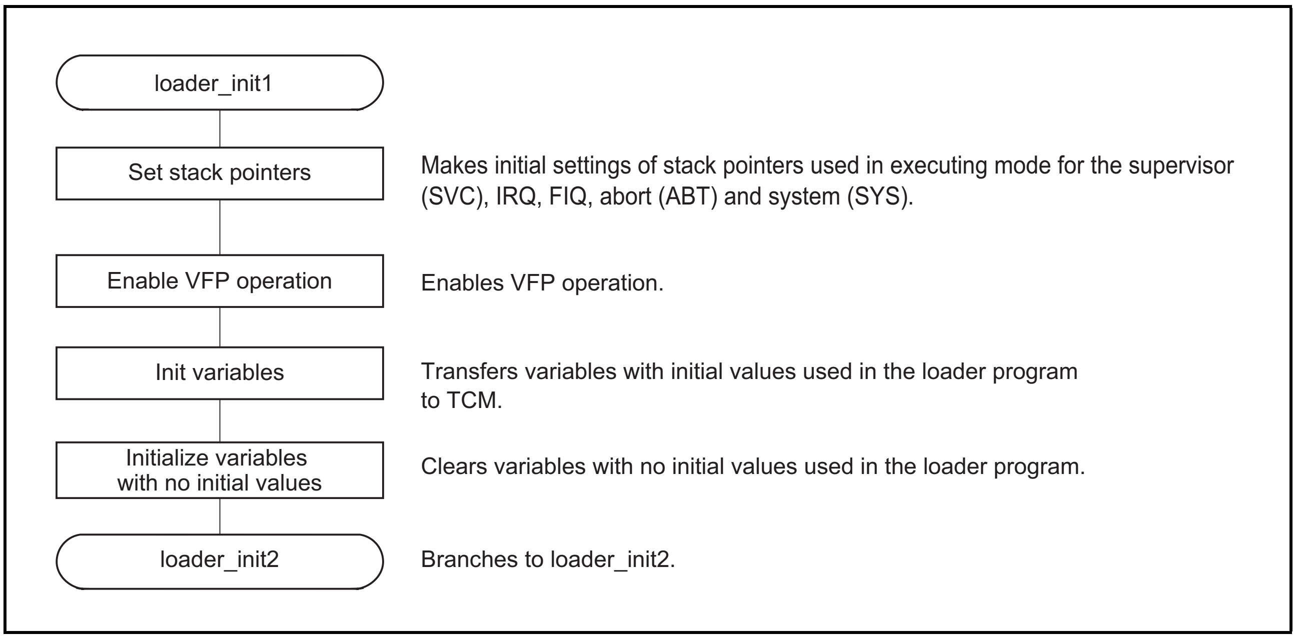
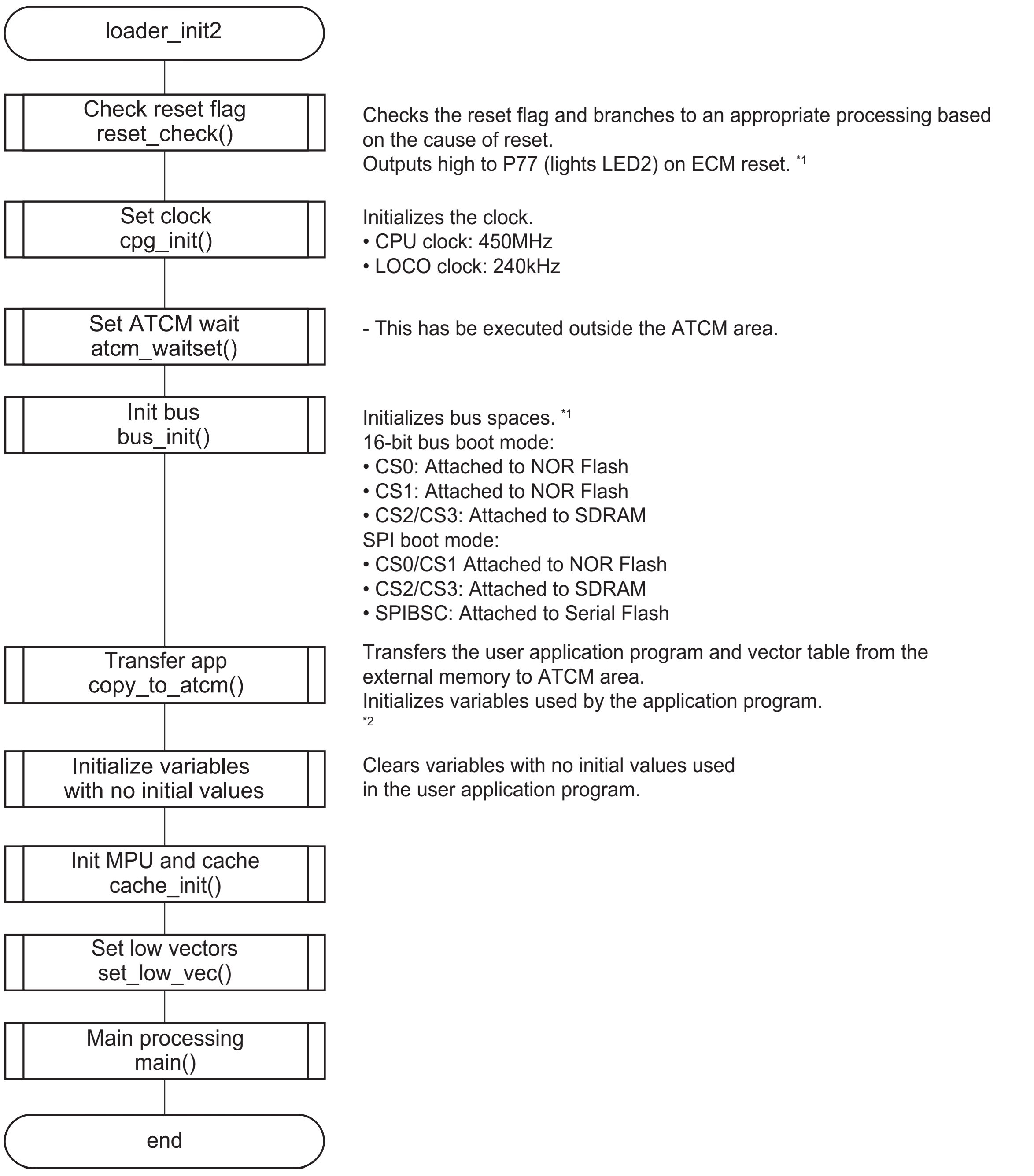

loader_init1
Set stack pointers
Makes initial settings of stack pointers used in executing mode for the supervisor (SVC), IRQ, FIQ, abort (ABT) and system (SYS).
Enable VFP operation
Enables VFP operation.
Init variables
Transfers variables with initial values used in the loader program to TCM.
Initialize variables with no initial values
Clears variables with no initial values used in the loader program.
loader_init2
Branches to loader_init2.

Figure 5.6 Loader Program 1 Processing

5.10.2 Loader Program 1 Processing

Figure 5.7 shows the loader program 2 flowchart.


loader_init2
Check reset flag
reset_check()
Checks the reset flag and branches to an appropriate processing based on the cause of reset.
Outputs high to P77 (lights LED2) on ECM reset. *1
Set clock
cpg_init()
Initializes the clock.
• CPU clock: 450MHz
• LOCO clock: 240kHz
Set ATCM wait
atcm_waitset()
- This has be executed outside the ATCM area.
Init bus
bus_init()
Initializes bus spaces. *1
16-bit bus boot mode:
• CS0: Attached to NOR Flash
• CS1: Attached to NOR Flash
• CS2/CS3: Attached to SDRAM
SPI boot mode:
• CS0/CS1 Attached to NOR Flash
• CS2/CS3: Attached to SDRAM
• SPIBSC: Attached to Serial Flash
Transfer app
copy_to_atcm()
Transfers the user application program and vector table from the external memory to ATCM area.
Initializes variables used by the application program. *2
Initialize variables
with no initial values
Clears variables with no initial values used in the user application program.
Init MPU and cache
cache_init()
Set low vectors
set_low_vec()
Main processing
main()
end

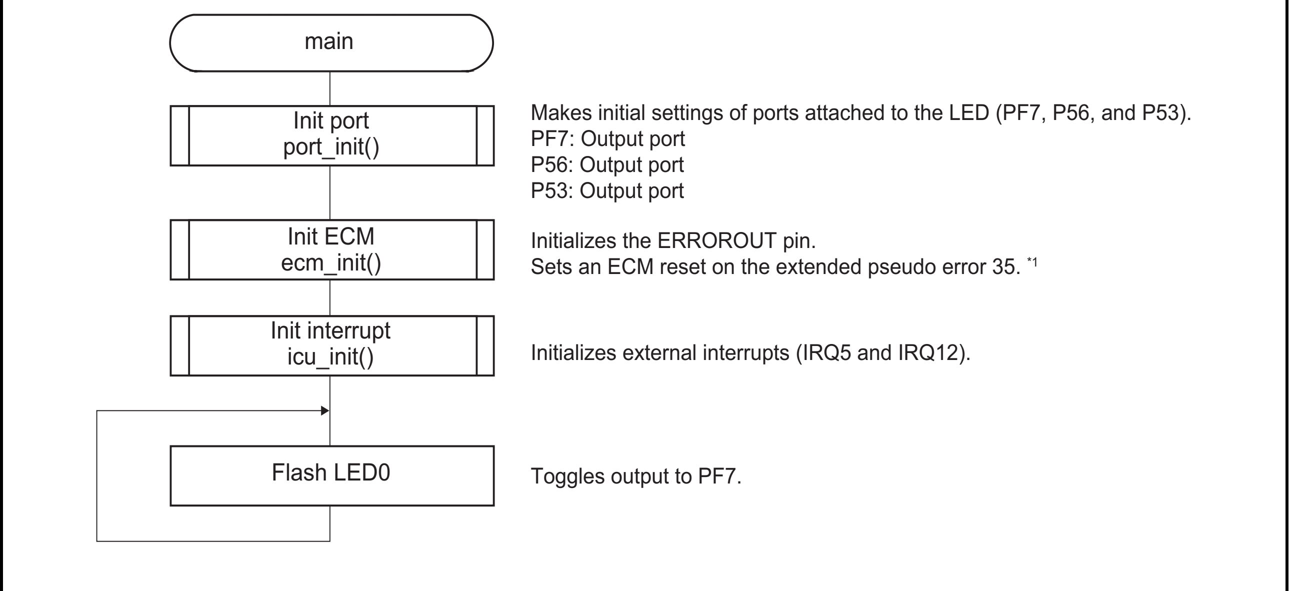
Note 1. Not executed in RAM version.

Note 2. In RAM version, only variable initialization is performed. The EWARM built-in function (__iar_data_init3()) is called to initialize the variables used by the application program.

Figure 5.7 Loader Program 2 Processing

5.10.3 Main Processing

Figure 5.8 shows the main processing flowchart.


main
Init port
port_init()
Makes initial settings of ports attached to the LED (PF7, P56, and P53).
PF7: Output port
P56: Output port
P53: Output port
Init ECM
ecm_init()
Initializes the ERROROUT pin.
Sets an ECM reset on the extended pseudo error 35. *1
Init interrupt
icu_init()
Initializes external interrupts (IRQ5 and IRQ12).
Flash LED0
Toggles output to PF7.

Note 1. This process is not performed in RAM execution version.

Figure 5.8 Main Processing

5.10.4 Reset Flag Detection Processing

Figure 5.9 and Figure 5.10 show the flowchart of reset flag detection processing.

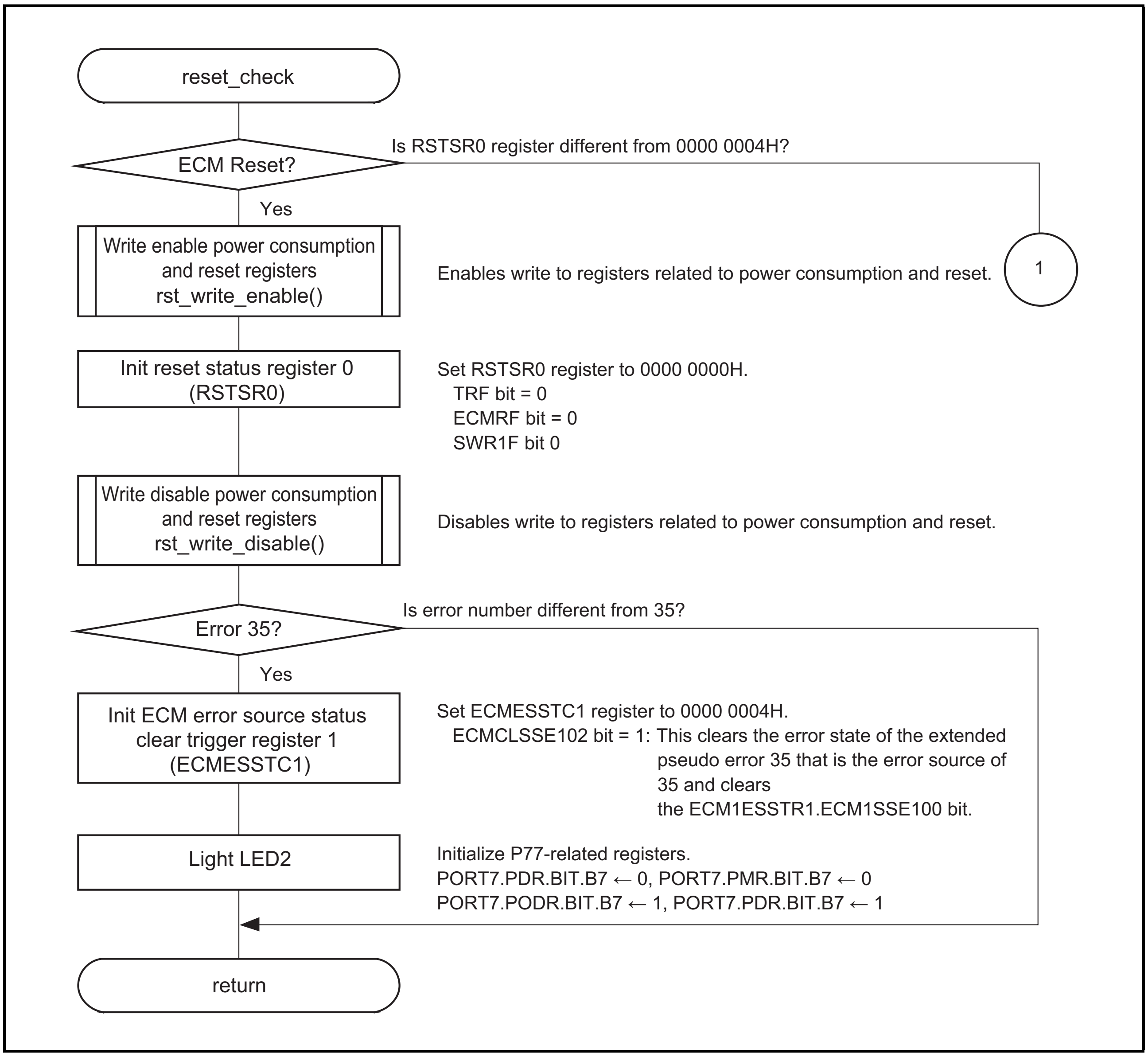

reset_check
ECM Reset?
Is RSTSR0 register different from 0000 0004H?
Yes
1
Write enable power consumption and reset registers rst_write_enable()
Enables write to registers related to power consumption and reset.
Init reset status register 0 (RSTSR0)
Set RSTSR0 register to 0000 0000H.
TRF bit = 0
ECMRF bit = 0
SWR1F bit 0
Write disable power consumption and reset registers rst_write_disable()
Disables write to registers related to power consumption and reset.
Error 35?
Is error number different from 35?
Yes
Init ECM error source status clear trigger register 1 (ECMESSTC1)
Set ECMESSTC1 register to 0000 0004H.
ECMCLSSE102 bit = 1: This clears the error state of the extended pseudo error 35 that is the error source of 35 and clears the ECM1ESSTR1.ECM1SSE100 bit.
Light LED2
Initialize P77-related registers.
PORT7.PDR.BIT.B7 ← 0, PORT7.PMR.BIT.B7 ← 0
PORT7.PODR.BIT.B7 ← 1, PORT7.PDR.BIT.B7 ← 1
return

Figure 5.9 Reset Flag Detection Processing

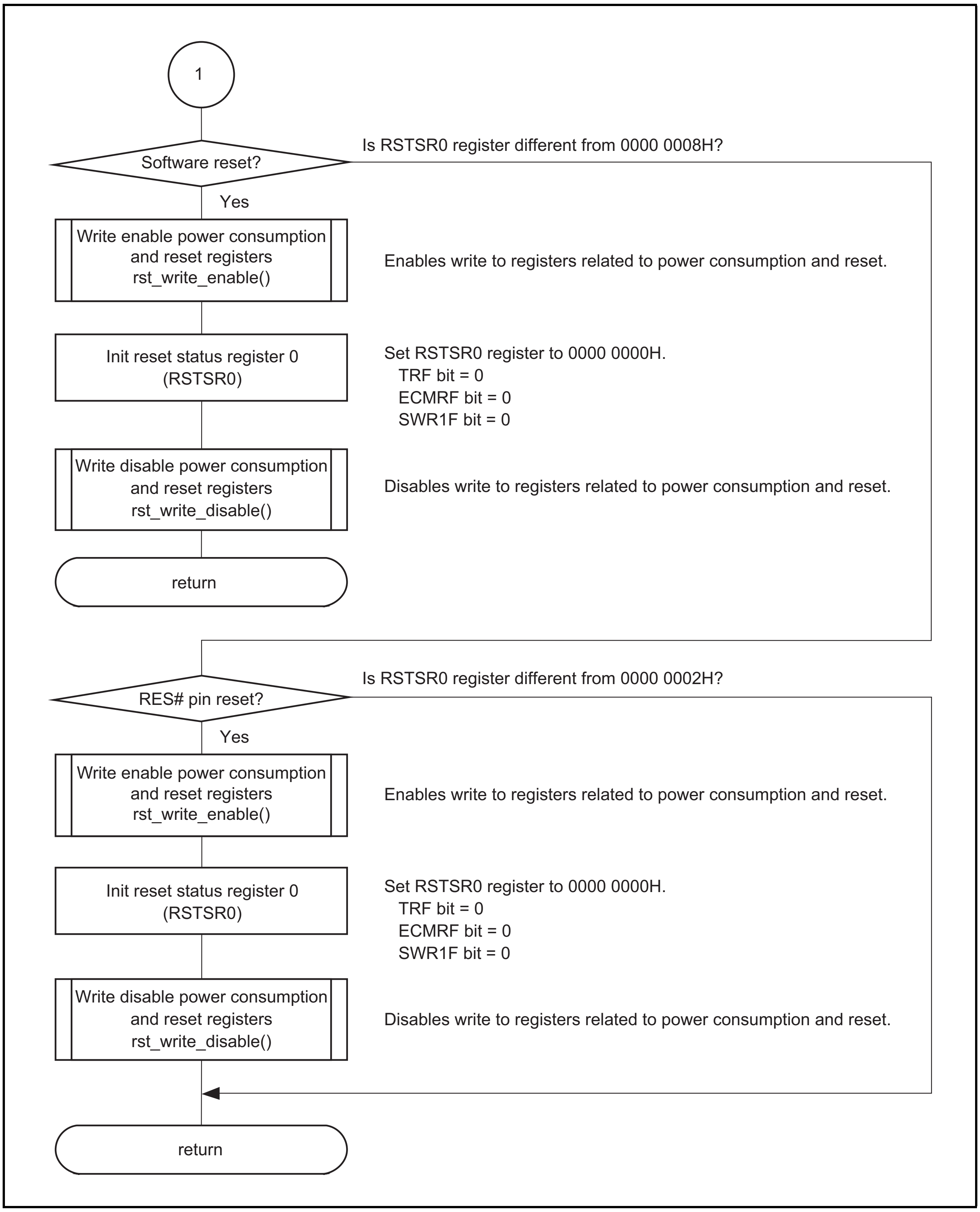
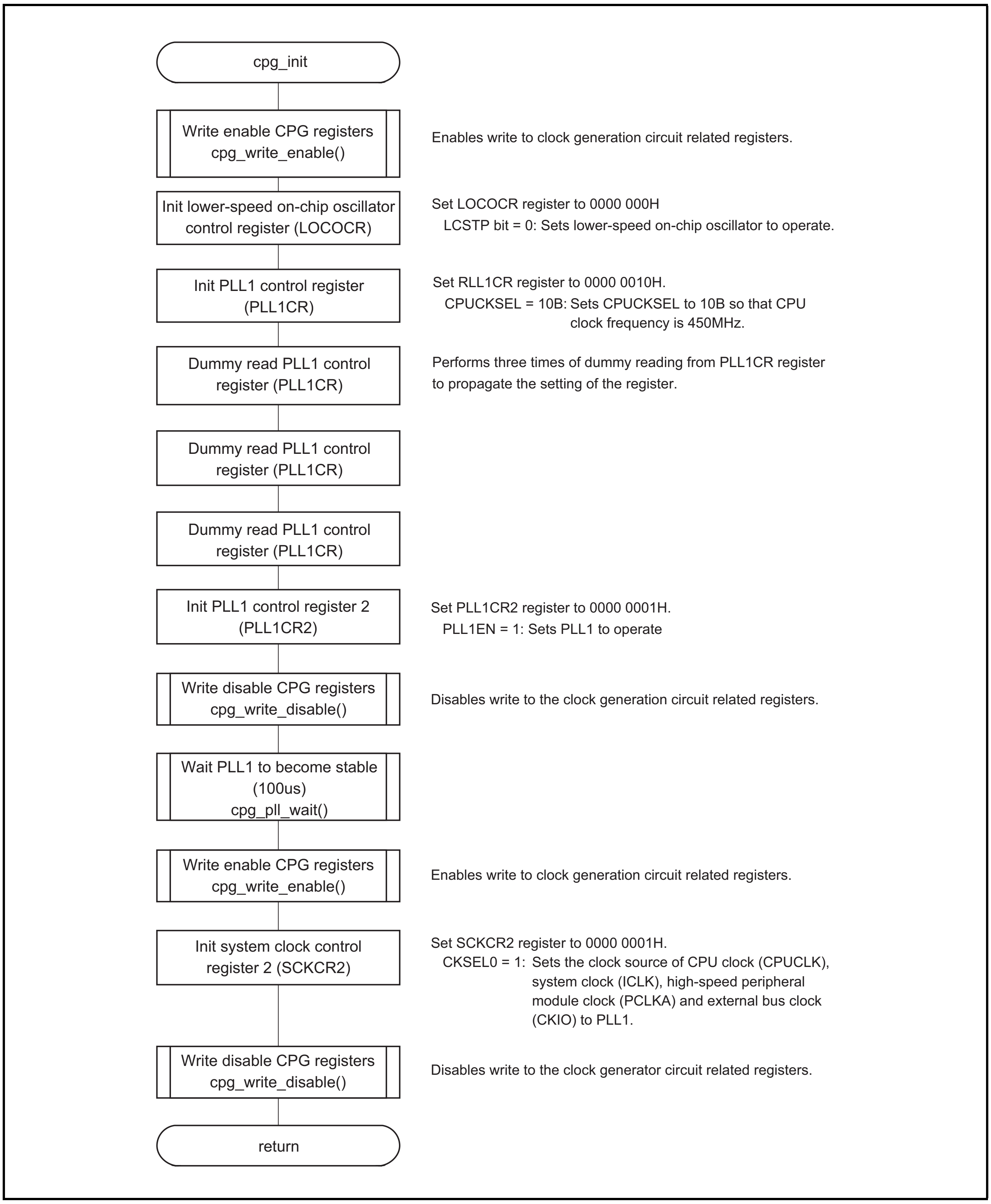

1
Software reset?
Is RSTSR0 register different from 0000 0008H?
Yes
Write enable power consumption and reset registers rst_write_enable()
Enables write to registers related to power consumption and reset.
Init reset status register 0 (RSTSR0)
Set RSTSR0 register to 0000 0000H.
TRF bit = 0
ECMRF bit = 0
SWR1F bit = 0
Write disable power consumption and reset registers rst_write_disable()
Disables write to registers related to power consumption and reset.
return
RES# pin reset?
Is RSTSR0 register different from 0000 0002H?
Yes
Write enable power consumption and reset registers rst_write_enable()
Enables write to registers related to power consumption and reset.
Init reset status register 0 (RSTSR0)
Set RSTSR0 register to 0000 0000H.
TRF bit = 0
ECMRF bit = 0
SWR1F bit = 0
Write disable power consumption and reset registers rst_write_disable()
Disables write to registers related to power consumption and reset.
return

Figure 5.10 Reset Flag Detection Processing

5.10.5 Clock Setting Processing

Figure 5.11 shows the clock setting processing flowchart.

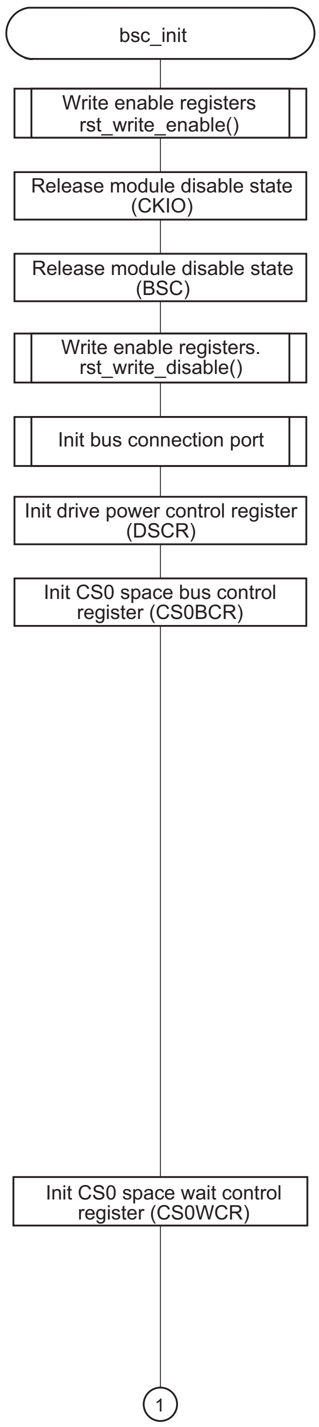


cpg_init
Write enable CPG registers cpg_write_enable()
Enables write to clock generation circuit related registers.
Init lower-speed on-chip oscillator control register (LOCOCR)
Set LOCOCR register to 0000 000H
LCSTP bit = 0: Sets lower-speed on-chip oscillator to operate.
Init PLL1 control register (PLL1CR)
Set RLL1CR register to 0000 0010H.
CPUCKSEL = 10B: Sets CPUCKSEL to 10B so that CPU clock frequency is 450MHz.
Dummy read PLL1 control register (PLL1CR)
Performs three times of dummy reading from PLL1CR register to propagate the setting of the register.
Dummy read PLL1 control register (PLL1CR)
Dummy read PLL1 control register (PLL1CR)
Init PLL1 control register 2 (PLL1CR2)
Set PLL1CR2 register to 0000 0001H.
PLL1EN = 1: Sets PLL1 to operate
Write disable CPG registers cpg_write_disable()
Disables write to the clock generation circuit related registers.
Wait PLL1 to become stable (100us) cpg_pll_wait()
Write enable CPG registers cpg_write_enable()
Enables write to clock generation circuit related registers.
Init system clock control register 2 (SCKCR2)
Set SCKCR2 register to 0000 0001H.
CKSEL0 = 1: Sets the clock source of CPU clock (CPUCLK), system clock (ICLK), high-speed peripheral module clock (PCLKA) and external bus clock (CKIO) to PLL1.
Write disable CPG registers cpg_write_disable()
Disables write to the clock generator circuit related registers.
return

Figure 5.11 Clock Setting Processing

5.10.6 Bus Setting Processing

Figure 5.12 to Figure 5.14 show the buss setting processing flowchart.


bsc_init
Write enable registers rst_write_enable()
Release module disable state (CKIO)
Release module disable state (BSC)
Write enable registers. rst_write_disable()
Init bus connection port
Init drive power control register (DSCR)
Init CS0 space bus control register (CS0BCR)
Init CS0 space wait control register (CS0WCR)
1

[1]

Enables write to power consumption and reset related registers.

Releases module disable state of CKIO.

Releases module disable state of BSC. [2]

Disables write to power consumption and reset related registers.

Different processing is required based on the mode:
set_bus_port_nor_boot(): 16-bit bus boot mode
set_bus_port_serial_boot(): SPI boot mode

Set DSCR register to 0001H.
Initializes CKIO to output high drive power to make SDRAM attachment.

Sets the following bits of the CS0 space bus control register: [2]

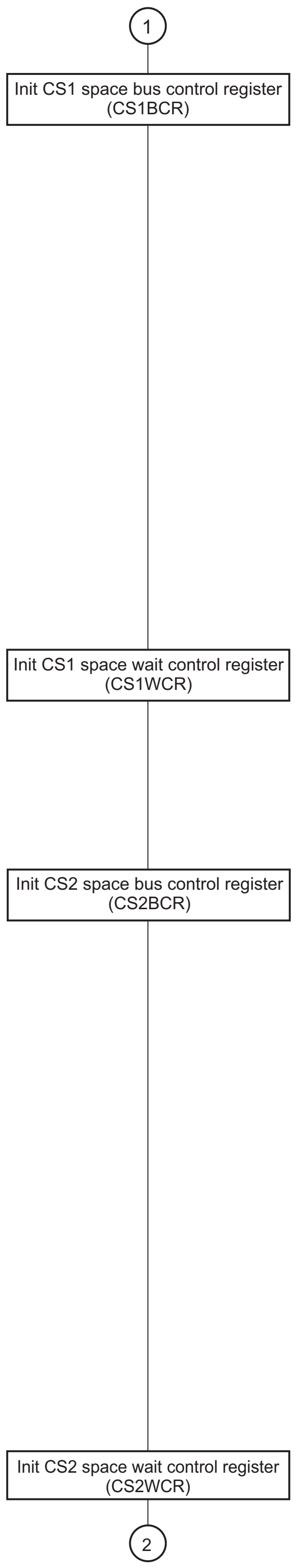
IWW = 001B:	Sets the number of idle cycles to one which is inserted after external memory access attached to the CS space. This applies to write-read and write-write cycles.
IWRWD = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-write cycles.
IWRWS = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-write cycles for repeated access to the same CS space.
IWRRD = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-read cycles for repeated access to a different CS space.
IWRRS = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-read cycles for repeated access to the same CS space.
TYPE = 000B:	Sets the type of memory attached to the CS space SRAM interface.
BSZ = 10B:	Sets the data bus width of the CS space to 16 bits.

Sets the following bits of the CS0 wait control register: [2]

SW = 10B:	Sets the delay cycle to 2.5 state between active state of address and CS# and active state of RD# and WE#.
WR = 0110B:	Set the number of waits to six cycles required for read and write access.
WM = 1:	This is required to disable external weight input.
HW = 00B:	Sets the delay state to 0.5 state from inactive state of RD# and WE# to inactive state of address and CS0#.

Note 1. This process is not performed in RAM execution version.
Note 2. This process is performed only in SPI boot mode.

Figure 5.12 Bus Setting Processing (1)


1
Init CS1 space bus control register (CS1BCR)
Init CS1 space wait control register (CS1WCR)
Init CS2 space bus control register (CS2BCR)
Init CS2 space wait control register (CS2WCR)
2

Sets the following bits of the CS1 space bus control register:

IWW = 001B:	Sets the number of idle cycles to one which is inserted after external memory access attached to the CS space. This applies to write-read and write-write cycles.
IWRWD = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-write cycles.
IWRWS = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-write cycles for repeated access to the same CS space.
IWRRD = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-read cycles for repeated access to a different CS space.
IWRRS = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-read cycles for repeated access to the same CS space.
TYPE = 000B:	Sets the type of memory attached to the CS space SRAM interface.
BSZ = 10B:	Sets the data bus width of the CS space to 16 bits.

Sets the following bits of the CS1 space wait control register:

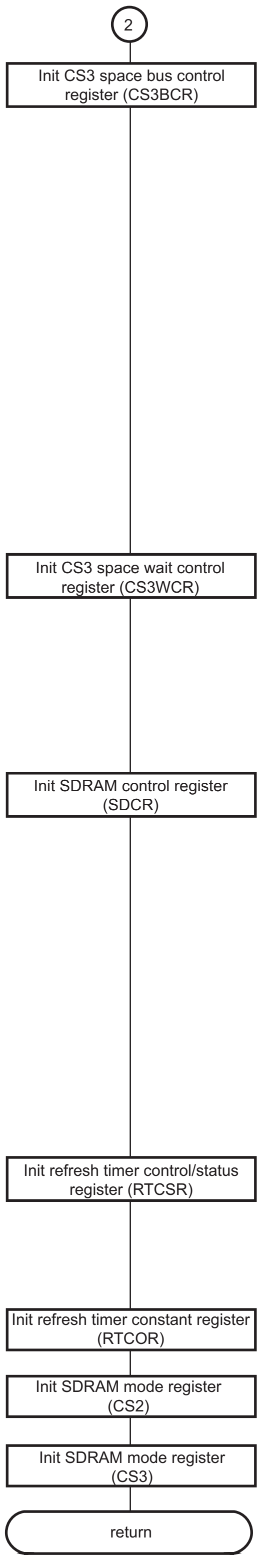
SW = 10B:	Sets the delay cycle to 2.5 state between active state of address and CS# and active state of RD# and WE#.
RW = 0110B:	Set the number of waits to six cycles required for read and write access.
WM = 1:	This is required to disable external weight input.
HW = 00B:	Sets the delay state to 0.5 state from inactive state of RD# and WE# to inactive state of address and CS0#.

Sets the following bits of the CS2 space bus control register:

IWW = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to write-read and write-write cycles.
IWRWD = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-write cycles.
IWRWS = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-write cycles for repeated access to the same CS space.
IWRRD = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-read cycles for repeated access to a different CS space.
IWRRS = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-read cycles for repeated access to the same CS space.
TYPE = 100B:	Sets the type of memory attached to the CS space SDRAM interface.
BSZ = 10B:	Sets the data bus width of the CS space to 16 bits.

Sets the following bits of the CS2 space wait control register:
A2CL = 01B: Set the CAS latency of area 2 to two.

Figure 5.13 Bus Setting Processing (2)


2
Init CS3 space bus control register (CS3BCR)
Init CS3 space wait control register (CS3WCR)
Init SDRAM control register (SDCR)
Init refresh timer control/status register (RTCSR)
Init refresh timer constant register (RTCOR)
Init SDRAM mode register (CS2)
Init SDRAM mode register (CS3)
return

Sets the following bits of the CS3 space bus control register:

IWW = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to write-read and write-write cycles.
IWRWD = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-write cycles.
IWRWS = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-write cycles for repeated access to the same CS space.
IWRRD = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-read cycles for repeated access to a different CS space.
IWRRS = 000B:	Sets the number of idle cycles to zero which is inserted after external memory access attached to the CS space. This applies to read-read cycles for repeated access to the same CS space.
TYPE = 100B:	Sets the type of memory attached to the CS space SDRAM interface.
BSZ = 10B:	Sets the data bus width of the CS space to 16 bits.

Sets the following bits of the CS3 space wait control register:

WTRP = 01B:	Sets the minimum number of states before precharge completion to one.
WTRCD = 01B:	Sets the minimum number of states from issuance of ACTV command to READ(A)/WRITE(A) command to one.
TRWL = 10B:	sets the minimum wait cycles before precharge activation to two.
WTRC = 10B:	Sets the minimum idle states to five.

Sets the following bits of the SDRAM control register:

A2ROW = 10B:	Sets the number of row address bits of SDRAM attached to area 2 to 13 bits.
A2COL = 10B:	Sets the number of column address bits of SDRAM attached to area 2 to 10 bits.
DEEP 0:	Configures so that self refresh will occur in deep power down mode.
RFSH = 1:	Configures SDRAM refresh control to refresh.
RMODE = 0:	Configures so that auto refresh will occur when RFSH bit set to one.
PDOWN = 0:	Configures SDRAM not to power down when access to SDRAM ends
BACTV = 0:	Configures so that access is done in auto precharge mode.
A3ROW = 10B:	Sets the number of row address bits of SDRAM attached to area 3 to 13 bits.
A3COL = 10B:	Sets the number of column address bits of SDRAM attached to area 3 to 10 bits.

Sets RTCSR register to A55A0008H.

RPC = 000B:	Sets the repetition count of refreshing on refresh request to one.
CSK = 010B:	Sets the clock for counting up the refresh timer counter to CKIO/16.
CMIE = 0:	Configures so that interrupt request is disabled when CMF bit is set to one.

Sets RTCOR register to A55A0030H

Set A0003040H to 000H.

Set A0004040H to 000H.

Figure 5.14 Bus Setting Processing (3)

5.10.7 Application Program Transfer Processing

Figure 5.15 shows the flowchart of application program transfer processing.

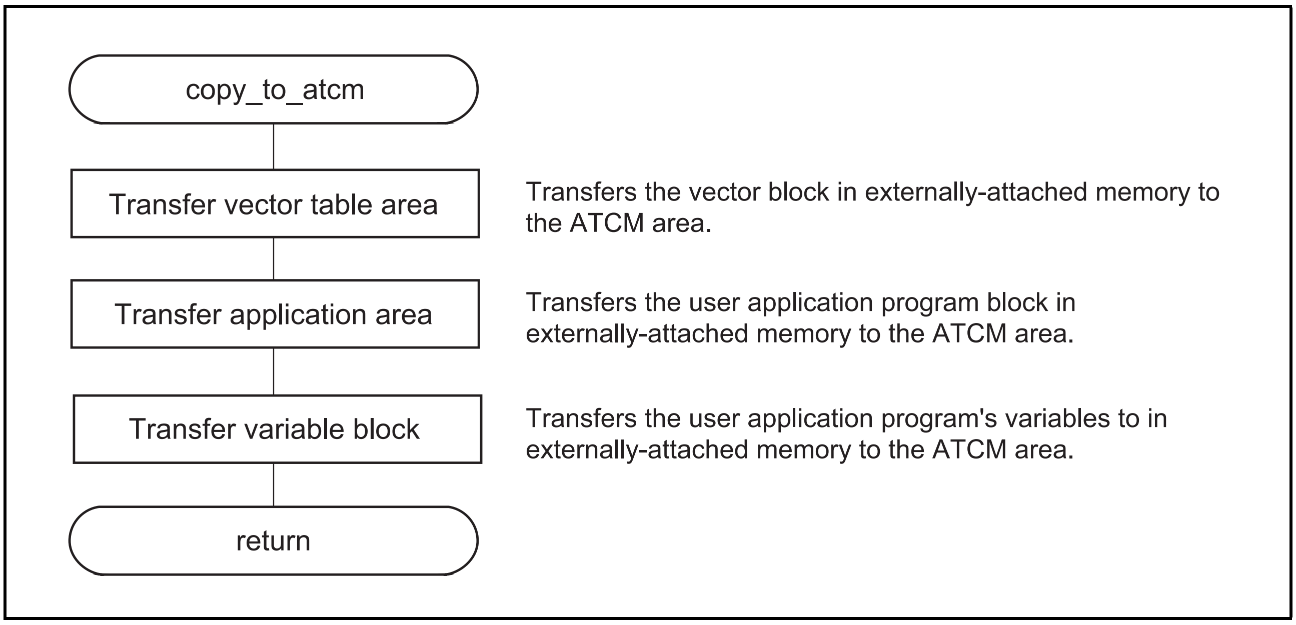

copy_to_atcm
Transfer vector table area
Transfers the vector block in externally-attached memory to the ATCM area.
Transfer application area
Transfers the user application program block in externally-attached memory to the ATCM area.
Transfer variable block
Transfers the user application program's variables to in externally-attached memory to the ATCM area.
return

Figure 5.15 Application Program Transfer Processing

5.10.8 Setting Processing for MPU and Cache

Figure 5.16 shows the flowchart of setting processing for MPU and cache.

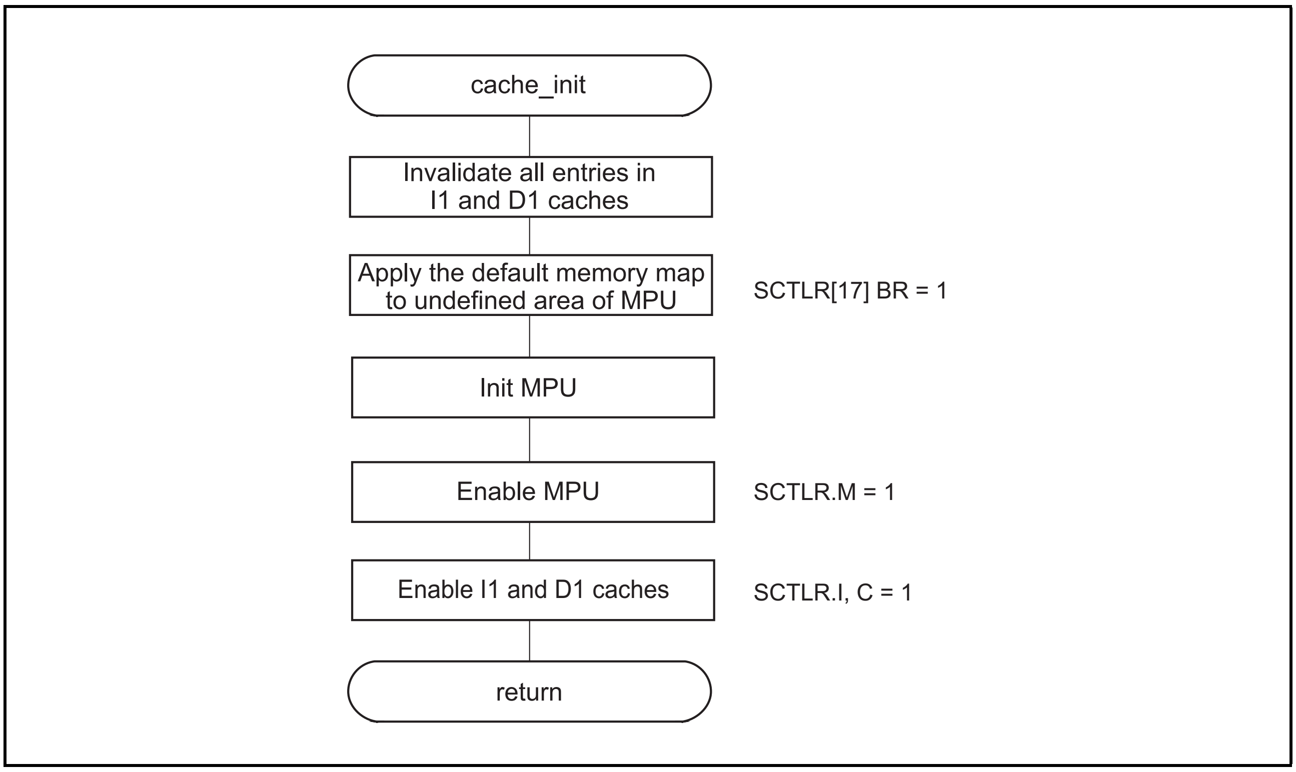


Figure 5.16 Setting Processing for MPU and Cache

5.10.9 Low Vector Setting Processing

Figure 5.17 shows the low vector setting processing flowchart.

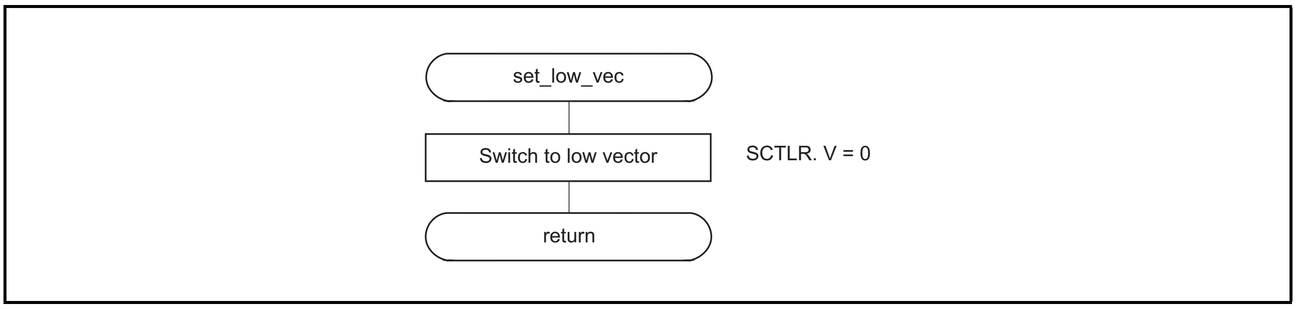


Figure 5.17 Low Vector Setting Processing

5.10.10 Port Setting Processing

Figure 5.18 shows the port setting processing flowchart.

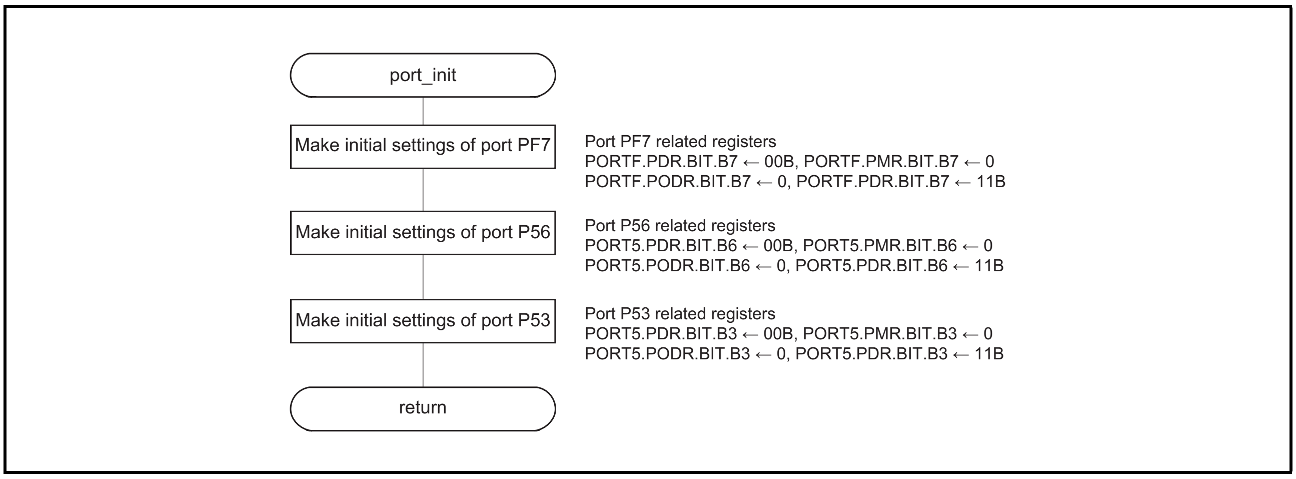

port_init
Make initial settings of port PF7
Port PF7 related registers
PORTF.PDR.BIT.B7 ← 00B, PORTF.PMR.BIT.B7 ← 0
PORTF.PODR.BIT.B7 ← 0, PORTF.PDR.BIT.B7 ← 11B
Make initial settings of port P56
Port P56 related registers
PORT5.PDR.BIT.B6 ← 00B, PORT5.PMR.BIT.B6 ← 0
PORT5.PODR.BIT.B6 ← 0, PORT5.PDR.BIT.B6 ← 11B
Make initial settings of port P53
Port P53 related registers
PORT5.PDR.BIT.B3 ← 00B, PORT5.PMR.BIT.B3 ← 0
PORT5.PODR.BIT.B3 ← 0, PORT5.PDR.BIT.B3 ← 11B
return

Figure 5.18 Port Setting Processing

5.10.11 ECM Setting Processing

Figure 5.19 shows the ECM setting processing flowchart.

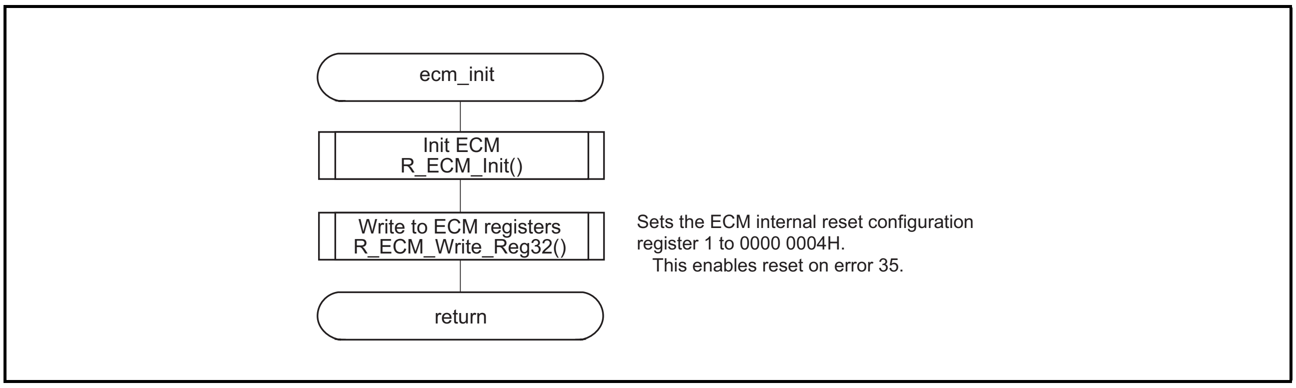

ecm_init
Init ECM
R_ECM_Init()
Write to ECM registers
R_ECM_Write_Reg32()
Sets the ECM internal reset configuration register 1 to 0000 0004H.
This enables reset on error 35.
return

Figure 5.19 ECM Setting Processing

5.10.12 ICU Setting Processing

Figure 5.20 shows the ICU setting processing flowchart.

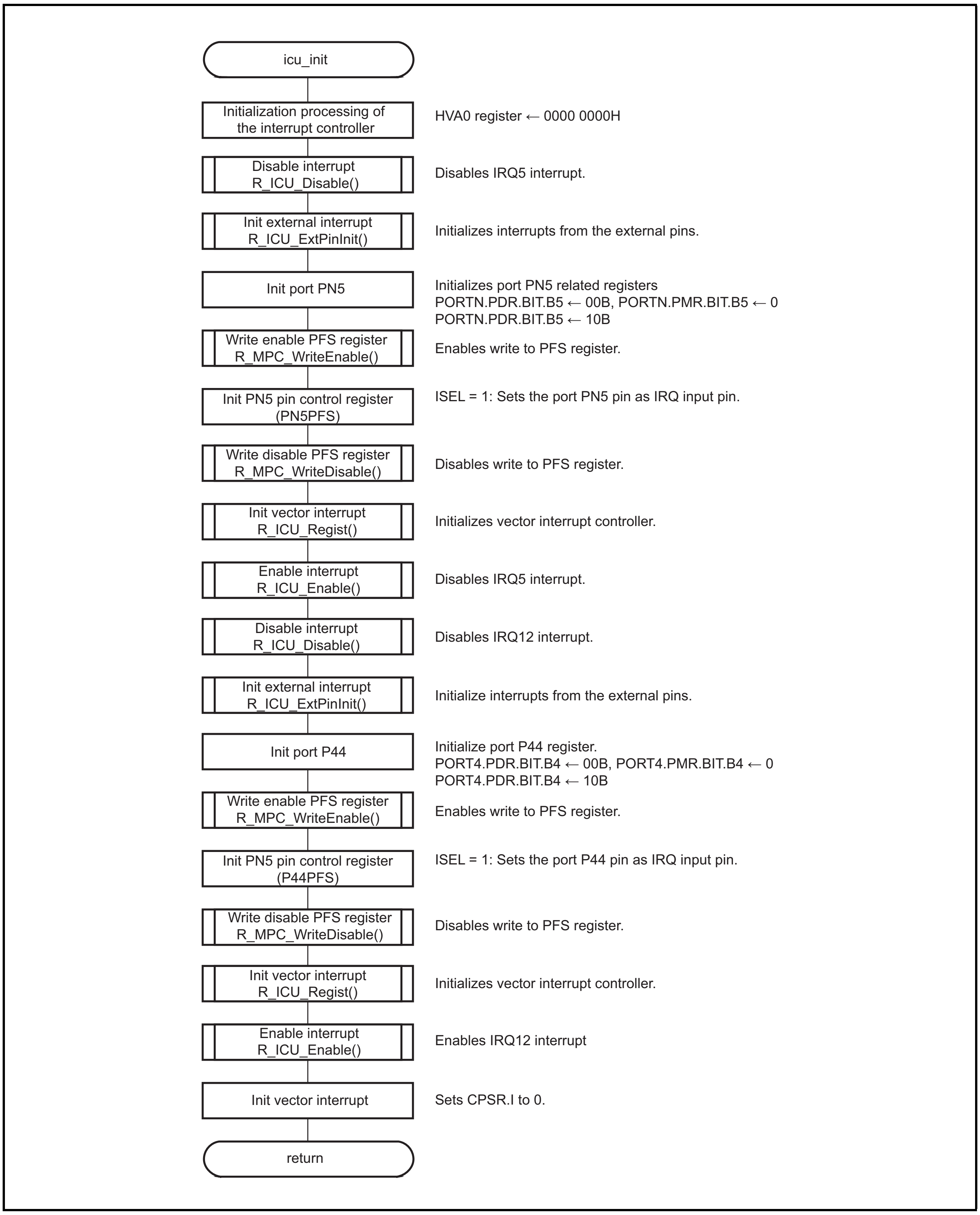

icu_init
Initialization processing of the interrupt controller
HVA0 register ← 0000 0000H
Disable interrupt R_ICU_Disable()
Disables IRQ5 interrupt.
Init external interrupt R_ICU_ExtPinInit()
Initializes interrupts from the external pins.
Init port PN5
Initializes port PN5 related registers
PORTN.PDR.BIT.B5 ← 00B, PORTN.PMR.BIT.B5 ← 0
PORTN.PDR.BIT.B5 ← 10B
Write enable PFS register R_MPC_WriteEnable()
Enables write to PFS register.
Init PN5 pin control register (PN5PFS)
ISEL = 1: Sets the port PN5 pin as IRQ input pin.
Write disable PFS register R_MPC_WriteDisable()
Disables write to PFS register.
Init vector interrupt R_ICU_Regist()
Initializes vector interrupt controller.
Enable interrupt R_ICU_Enable()
Disables IRQ5 interrupt.
Disable interrupt R_ICU_Disable()
Disables IRQ12 interrupt.
Init external interrupt R_ICU_ExtPinInit()
Initialize interrupts from the external pins.
Init port P44
Initialize port P44 register.
PORT4.PDR.BIT.B4 ← 00B, PORT4.PMR.BIT.B4 ← 0
PORT4.PDR.BIT.B4 ← 10B
Write enable PFS register R_MPC_WriteEnable()
Enables write to PFS register.
Init PN5 pin control register (P44PFS)
ISEL = 1: Sets the port P44 pin as IRQ input pin.
Write disable PFS register R_MPC_WriteDisable()
Disables write to PFS register.
Init vector interrupt R_ICU_Regist()
Initializes vector interrupt controller.
Enable interrupt R_ICU_Enable()
Enables IRQ12 interrupt
Init vector interrupt
Sets CPSR.I to 0.
return

Figure 5.20 ICU Setting Processing

5.10.13 ATCM Access Wait Setting Processing

Figure 5.21 shows the ATCM access wait setting processing.

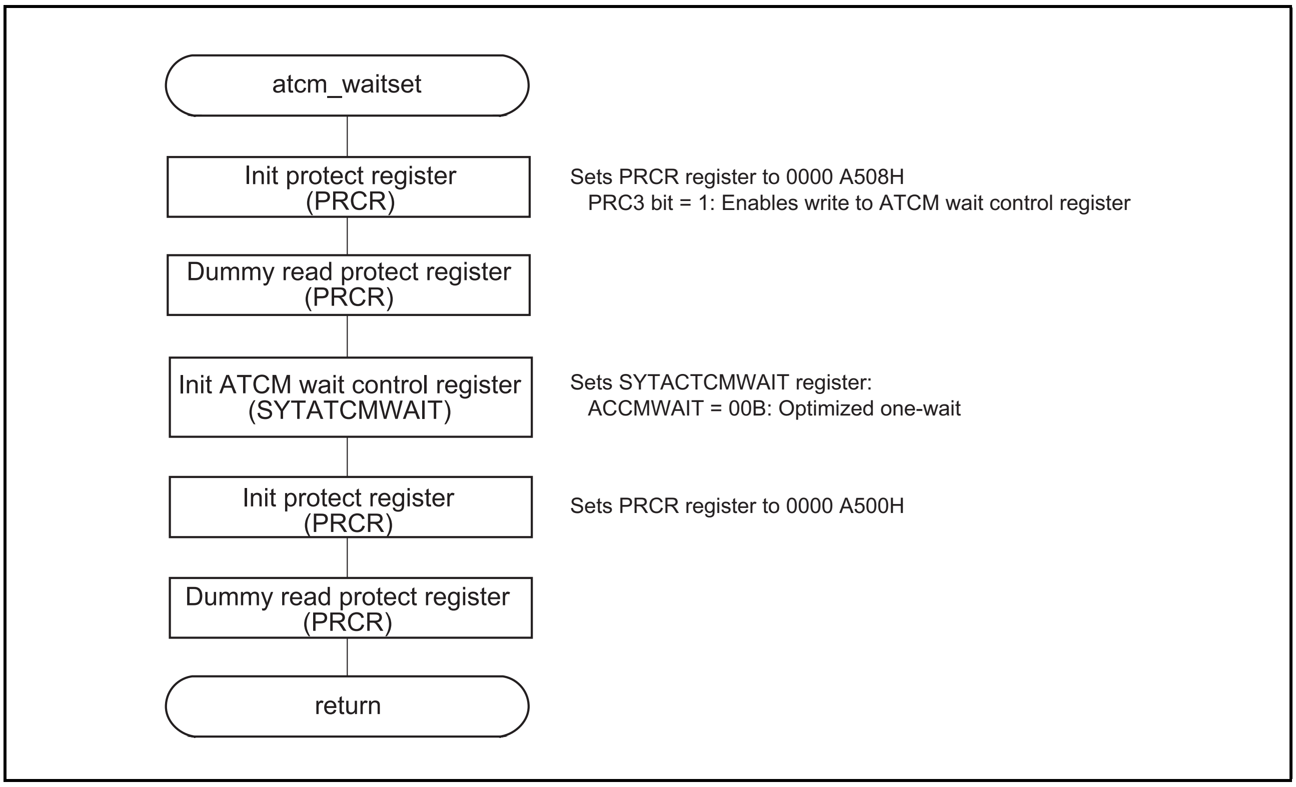

atcm_waitset
Init protect register (PRCR)
Sets PRCR register to 0000 A508H
PRC3 bit = 1: Enables write to ATCM wait control register
Dummy read protect register (PRCR)
Init ATCM wait control register (SYTATCMWAIT)
Sets SYTACTCMWAIT register:
ACCMWAIT = 00B: Optimized one-wait
Init protect register (PRCR)
Sets PRCR register to 0000 A500H
Dummy read protect register (PRCR)
return

Figure 5.21 ATCM Access Wait Setting Processing

5.10.14 PLL Stability Waiting (100us) Processing

Figure 5.22 shows the flowchart of PLL stability waiting (100us) processing.

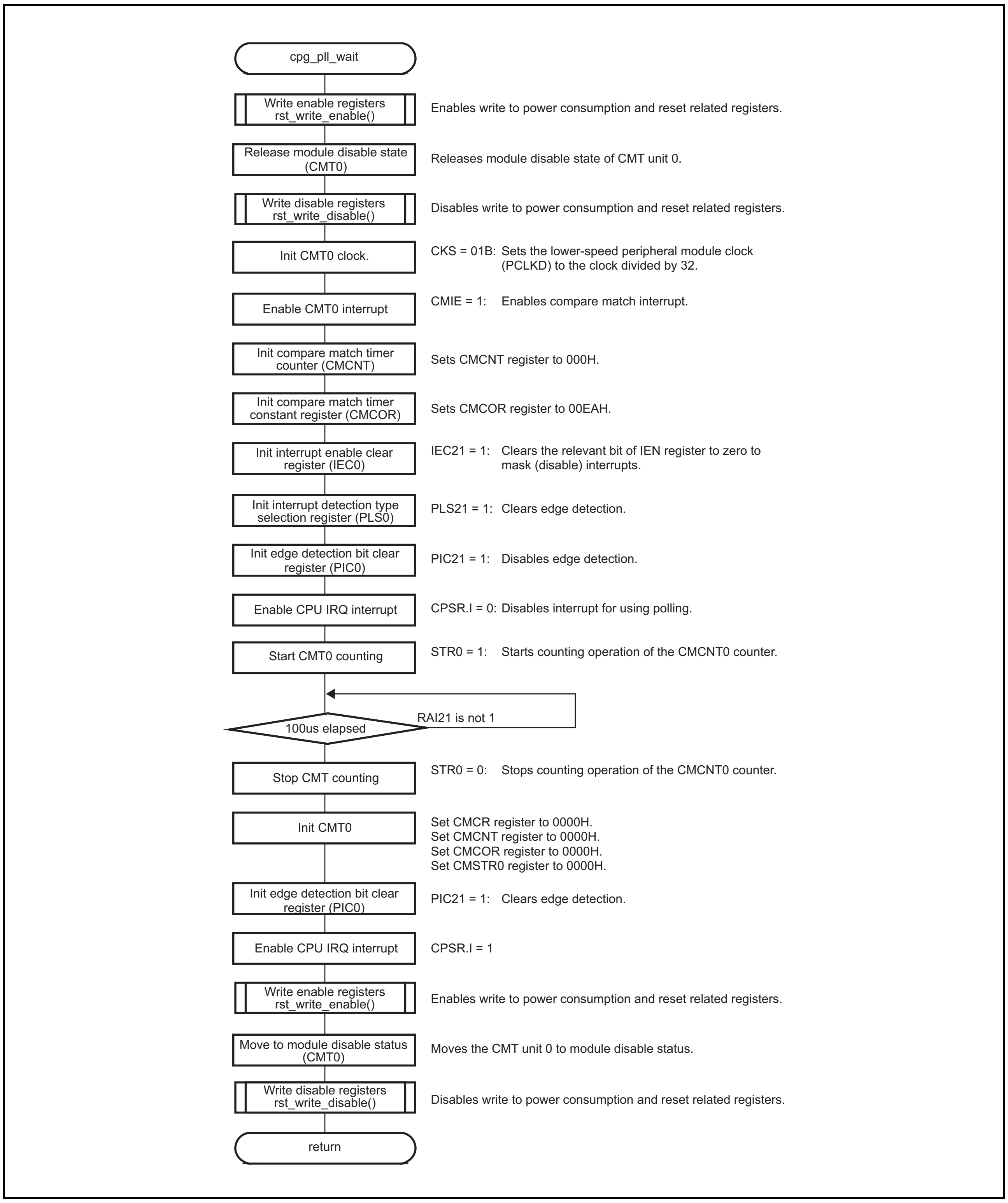

cpg_pll_wait
Write enable registers rst_write_enable()
Enables write to power consumption and reset related registers.
Release module disable state (CMT0)
Releases module disable state of CMT unit 0.
Write disable registers rst_write_disable()
Disables write to power consumption and reset related registers.
Init CMT0 clock.
CKS = 01B: Sets the lower-speed peripheral module clock (PCLKD) to the clock divided by 32.
Enable CMT0 interrupt
CMIE = 1: Enables compare match interrupt.
Init compare match timer counter (CMCNT)
Sets CMCNT register to 000H.
Init compare match timer constant register (CMCOR)
Sets CMCOR register to 00EAH.
Init interrupt enable clear register (IEC0)
IEC21 = 1: Clears the relevant bit of IEN register to zero to mask (disable) interrupts.
Init interrupt detection type selection register (PLS0)
PLS21 = 1: Clears edge detection.
Init edge detection bit clear register (PIC0)
PIC21 = 1: Disables edge detection.
Enable CPU IRQ interrupt
CPSR.I = 0: Disables interrupt for using polling.
Start CMT0 counting
STR0 = 1: Starts counting operation of the CMCNT0 counter.
100us elapsed
RAI21 is not 1
Stop CMT counting
STR0 = 0: Stops counting operation of the CMCNT0 counter.
Init CMT0
Set CMCR register to 0000H.
Set CMCNT register to 0000H.
Set CMCOR register to 0000H.
Set CMSTR0 register to 0000H.
Init edge detection bit clear register (PIC0)
PIC21 = 1: Clears edge detection.
Enable CPU IRQ interrupt
CPSR.I = 1
Write enable registers rst_write_enable()
Enables write to power consumption and reset related registers.
Move to module disable status (CMT0)
Moves the CMT unit 0 to module disable status.
Write disable registers rst_write_disable()
Disables write to power consumption and reset related registers.
return

Figure 5.22 PLL Stability Waiting (100us) Processing

5.10.15 ECM Initial Settings

Figure 5.23 shows the ECM initial setting flowchart.

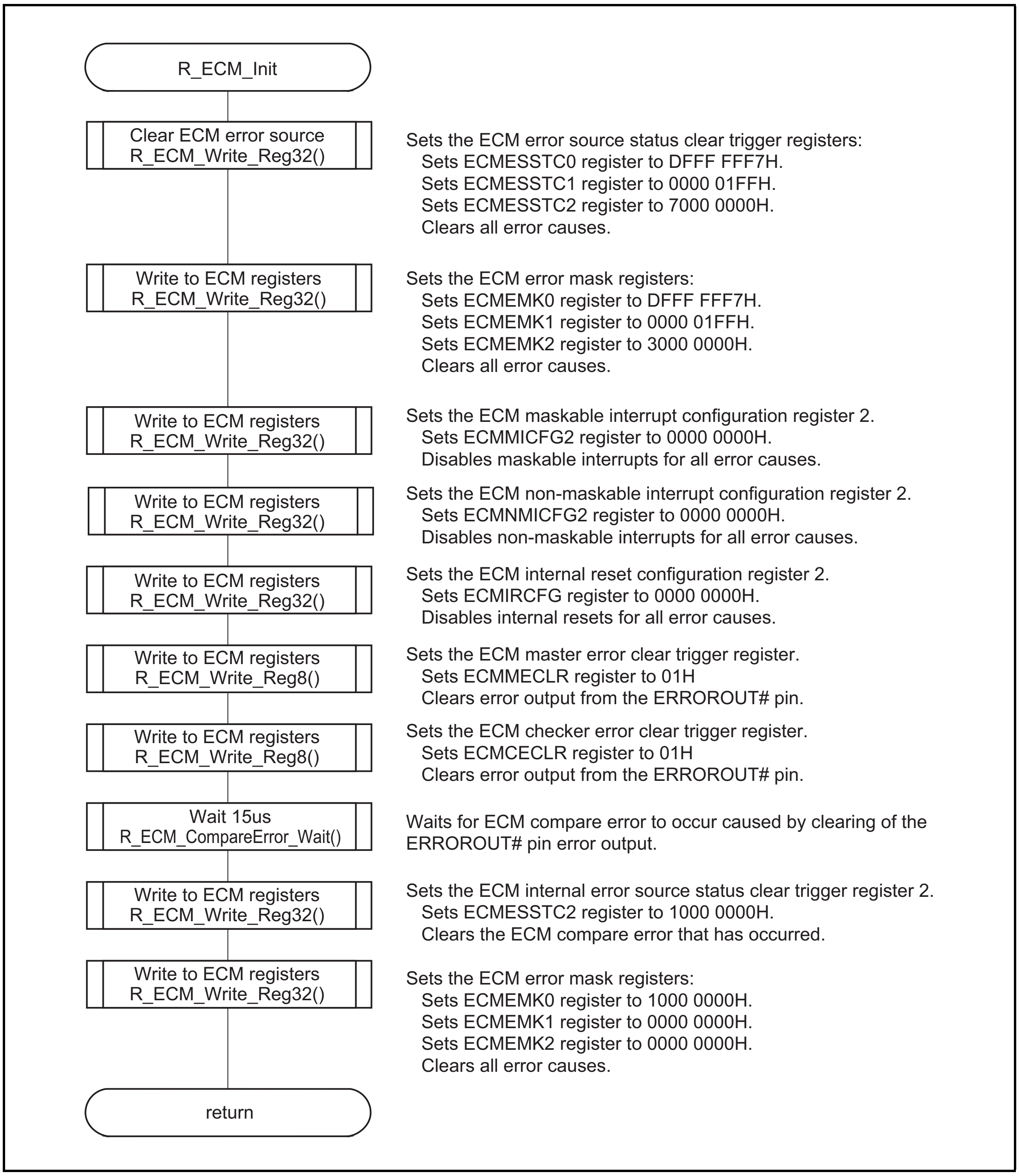

R_ECM_Init
Clear ECM error source
R_ECM_Write_Reg32()
Sets the ECM error source status clear trigger registers:
Sets ECMESSTC0 register to DFFF FFF7H.
Sets ECMESSTC1 register to 0000 01FFH.
Sets ECMESSTC2 register to 7000 0000H.
Clears all error causes.
Write to ECM registers
R_ECM_Write_Reg32()
Sets the ECM error mask registers:
Sets ECMEMK0 register to DFFF FFF7H.
Sets ECMEMK1 register to 0000 01FFH.
Sets ECMEMK2 register to 3000 0000H.
Clears all error causes.
Write to ECM registers
R_ECM_Write_Reg32()
Sets the ECM maskable interrupt configuration register 2.
Sets ECMMICFG2 register to 0000 0000H.
Disables maskable interrupts for all error causes.
Write to ECM registers
R_ECM_Write_Reg32()
Sets the ECM non-maskable interrupt configuration register 2.
Sets ECMNMICFG2 register to 0000 0000H.
Disables non-maskable interrupts for all error causes.
Write to ECM registers
R_ECM_Write_Reg32()
Sets the ECM internal reset configuration register 2.
Sets ECMIRCFG register to 0000 0000H.
Disables internal resets for all error causes.
Write to ECM registers
R_ECM_Write_Reg8()
Sets the ECM master error clear trigger register.
Sets ECMMECLR register to 01H
Clears error output from the ERROROUT# pin.
Write to ECM registers
R_ECM_Write_Reg8()
Sets the ECM checker error clear trigger register.
Sets ECMCECLR register to 01H
Clears error output from the ERROROUT# pin.
Wait 15us
R_ECM_CompareError_Wait()
Waits for ECM compare error to occur caused by clearing of the ERROROUT# pin error output.
Write to ECM registers
R_ECM_Write_Reg32()
Sets the ECM internal error source status clear trigger register 2.
Sets ECMESSTC2 register to 1000 0000H.
Clears the ECM compare error that has occurred.
Write to ECM registers
R_ECM_Write_Reg32()
Sets the ECM error mask registers:
Sets ECMEMK0 register to 1000 0000H.
Sets ECMEMK1 register to 0000 0000H.
Sets ECMEMK2 register to 0000 0000H.
Clears all error causes.
return

Figure 5.23 ECM Initial Settings

5.10.16 Interrupt Enabling Setting Processing

Figure 5.24 shows the interrupt enabling setting processing flowchart.

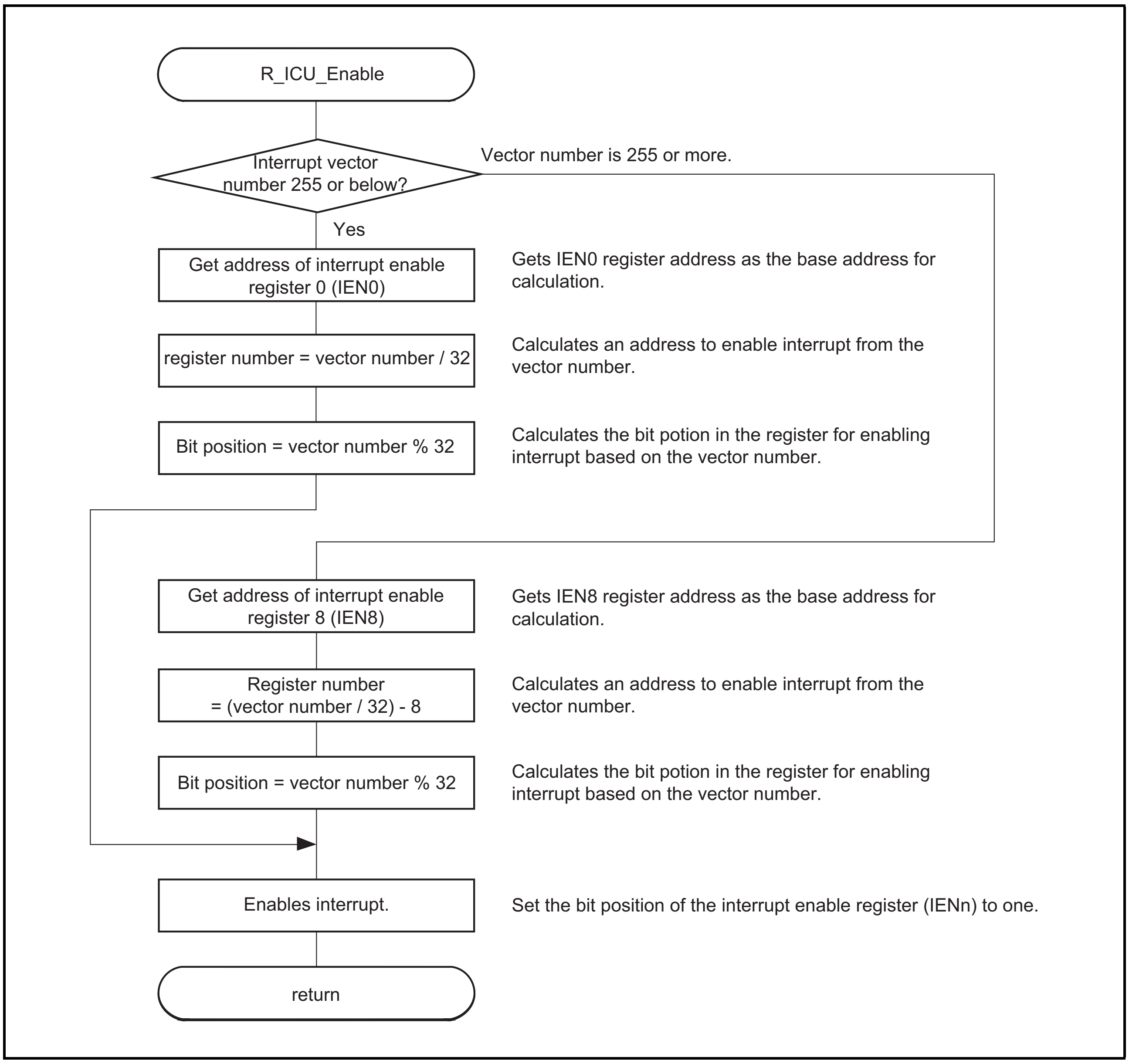

R_ICU_Enable
Interrupt vector number 255 or below?
Vector number is 255 or more.
Yes
Get address of interrupt enable register 0 (IEN0)
Gets IEN0 register address as the base address for calculation.
register number = vector number / 32
Calculates an address to enable interrupt from the vector number.
Bit position = vector number % 32
Calculates the bit potion in the register for enabling interrupt based on the vector number.
Get address of interrupt enable register 8 (IEN8)
Gets IEN8 register address as the base address for calculation.
Register number = (vector number / 32) - 8
Calculates an address to enable interrupt from the vector number.
Bit position = vector number % 32
Calculates the bit potion in the register for enabling interrupt based on the vector number.
Enables interrupt.
Set the bit position of the interrupt enable register (IENn) to one.
return

Figure 5.24 Interrupt Enabling Setting Processing

5.10.17 Interrupt Disabling Setting Processing

Figure 5.25 shows the interrupt disabling setting processing flowchart.

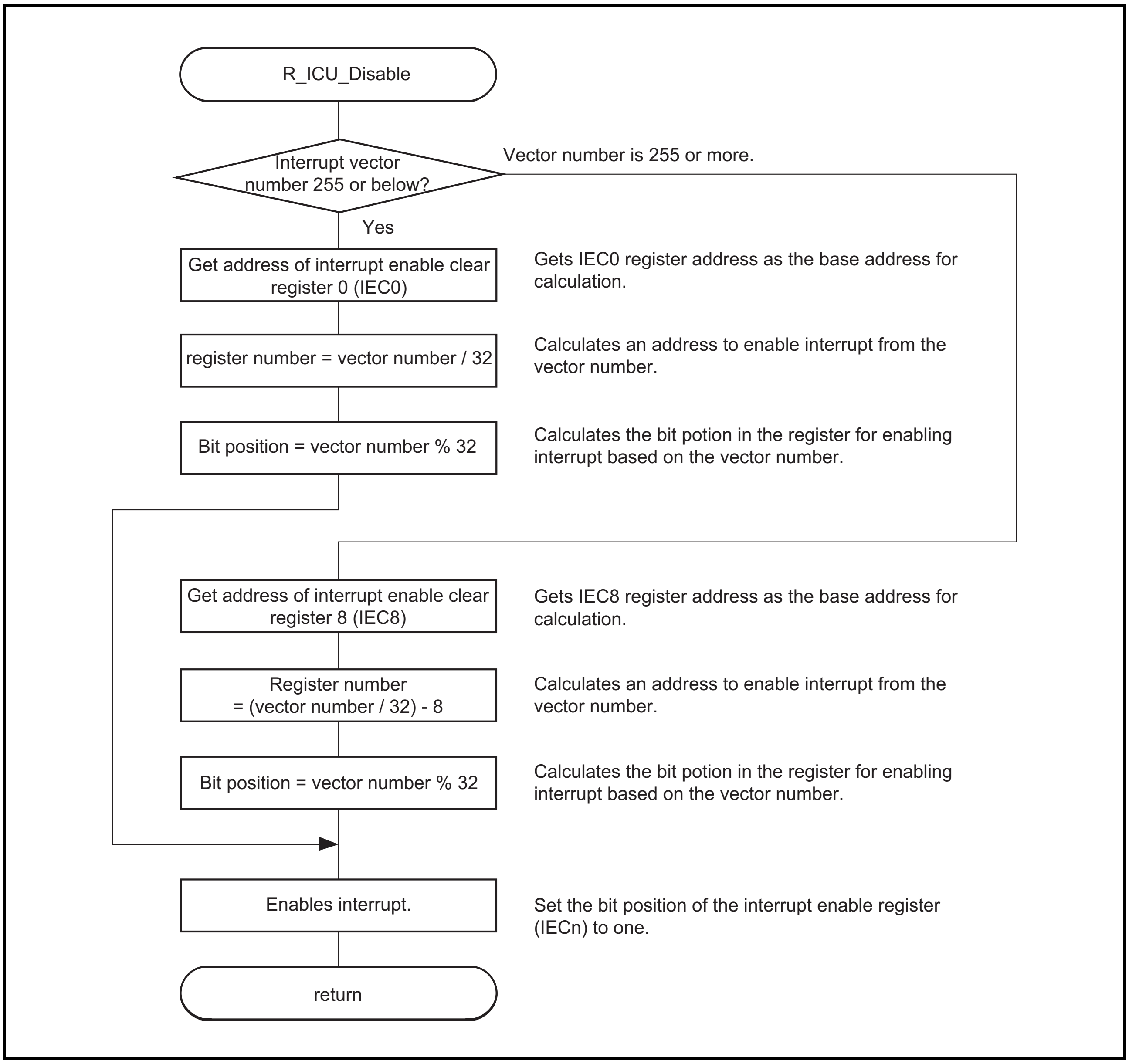

R_ICU_Disable
Interrupt vector number 255 or below?
Vector number is 255 or more.
Yes
Get address of interrupt enable clear register 0 (IEC0)
Gets IEC0 register address as the base address for calculation.
register number = vector number / 32
Calculates an address to enable interrupt from the vector number.
Bit position = vector number % 32
Calculates the bit potion in the register for enabling interrupt based on the vector number.
Get address of interrupt enable clear register 8 (IEC8)
Gets IEC8 register address as the base address for calculation.
Register number = (vector number / 32) - 8
Calculates an address to enable interrupt from the vector number.
Bit position = vector number % 32
Calculates the bit potion in the register for enabling interrupt based on the vector number.
Enables interrupt.
Set the bit position of the interrupt enable register (IECn) to one.
return

Figure 5.25 Interrupt Disabling Setting Processing

5.10.18 External Interrupt Setting Processing

Figure 5.26 shows the flowchart of external interrupt setting processing.

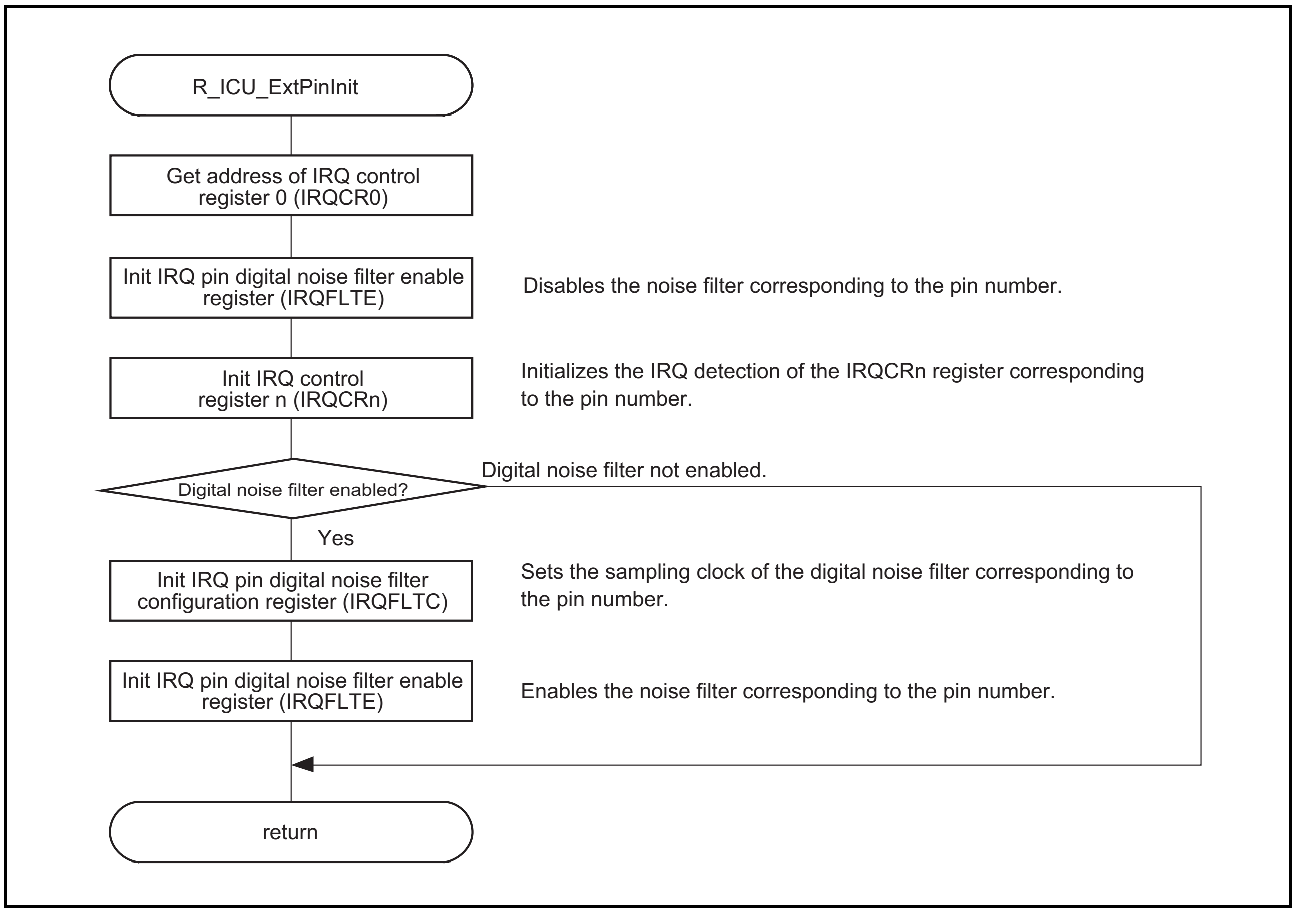

R_ICU_ExtPinInit
Get address of IRQ control register 0 (IRQCR0)
Init IRQ pin digital noise filter enable register (IRQFLTE)
Disables the noise filter corresponding to the pin number.
Init IRQ control register n (IRQCRn)
Initializes the IRQ detection of the IRQCRn register corresponding to the pin number.
Digital noise filter enabled?
Digital noise filter not enabled.
Yes
Init IRQ pin digital noise filter configuration register (IRQFLTC)
Sets the sampling clock of the digital noise filter corresponding to the pin number.
Init IRQ pin digital noise filter enable register (IRQFLTE)
Enables the noise filter corresponding to the pin number.
return

Figure 5.26 External Interrupt Setting Processing

5.10.19 Interrupt Registration Setting Processing

Figure 5.27 shows the flowchart of interrupt registration setting processing.

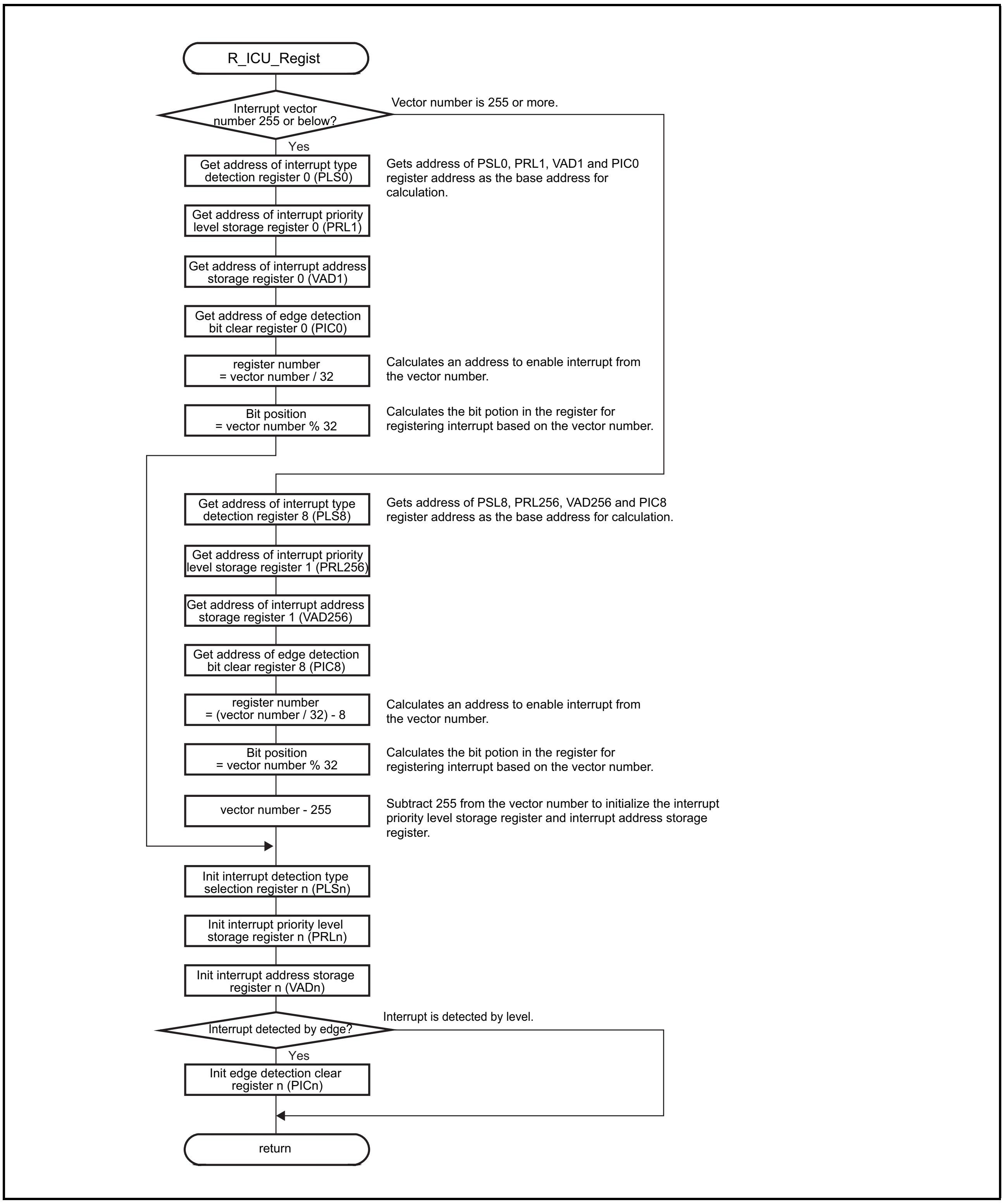

R_ICU_Regist
Interrupt vector number 255 or below?
Vector number is 255 or more.
Yes
Get address of interrupt type detection register 0 (PLS0)
Gets address of PSL0, PRL1, VAD1 and PIC0 register address as the base address for calculation.
Get address of interrupt priority level storage register 0 (PRL1)
Get address of interrupt address storage register 0 (VAD1)
Get address of edge detection bit clear register 0 (PIC0)
register number = vector number / 32
Calculates an address to enable interrupt from the vector number.
Bit position = vector number % 32
Calculates the bit potion in the register for registering interrupt based on the vector number.
Get address of interrupt type detection register 8 (PLS8)
Gets address of PSL8, PRL256, VAD256 and PIC8 register address as the base address for calculation.
Get address of interrupt priority level storage register 1 (PRL256)
Get address of interrupt address storage register 1 (VAD256)
Get address of edge detection bit clear register 8 (PIC8)
register number = (vector number / 32) - 8
Calculates an address to enable interrupt from the vector number.
Bit position = vector number % 32
Calculates the bit potion in the register for registering interrupt based on the vector number.
vector number - 255
Subtract 255 from the vector number to initialize the interrupt priority level storage register and interrupt address storage register.
Init interrupt detection type selection register n (PLSn)
Init interrupt priority level storage register n (PRLn)
Init interrupt address storage register n (VADn)
Interrupt detected by edge?
Interrupt is detected by level.
Yes
Init edge detection clear register n (PICn)
return

Figure 5.27 Interrupt Registration Setting Processing

5.10.20 Register Setting on Serial Flash Memory (User Defined) Processing

Figure 5.28 shows the flowchart of register setting on serial flash memory (user defined) processing.

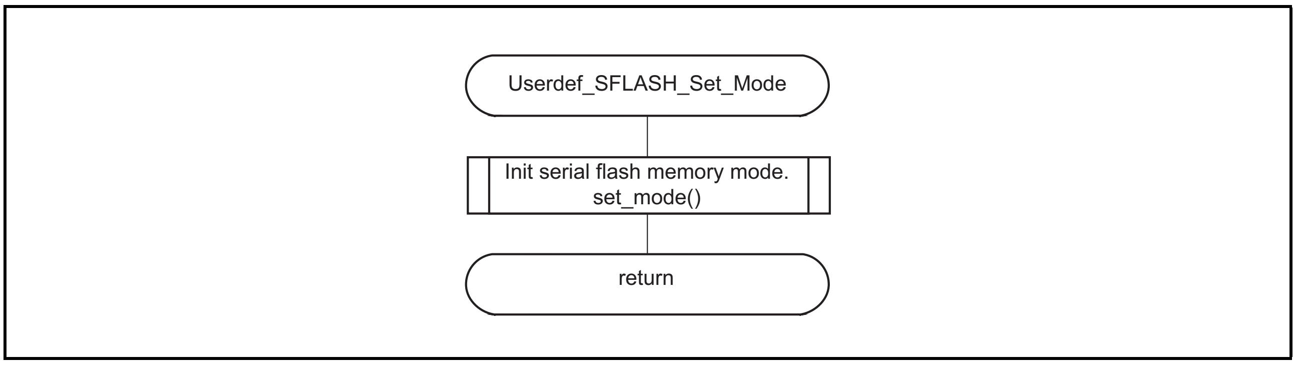


Figure 5.28 Register Setting on Serial Flash Memory (User Defined) Processing

5.10.21 Enabling Serial Flash Memory Write (User Defined) Processing

Figure 5.29 shows the flowchart of enabling serial flash memory write (user defined) processing.

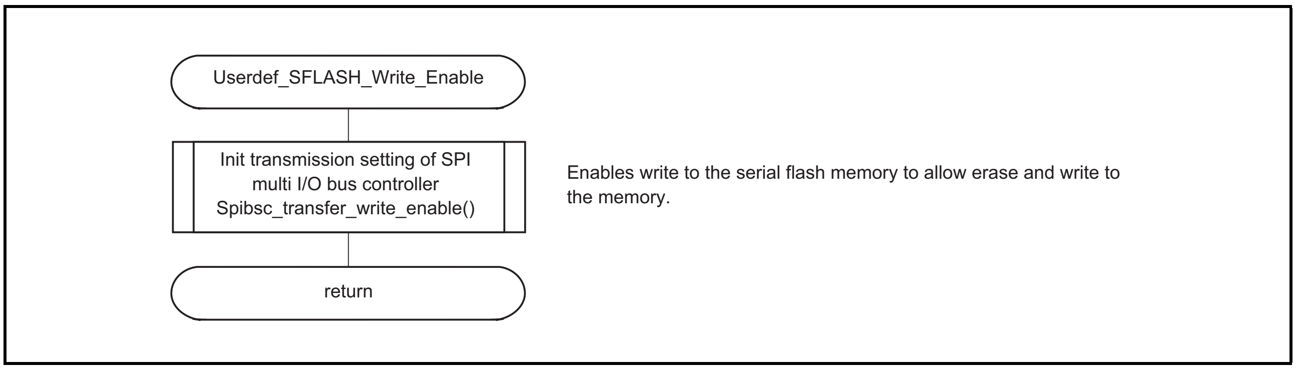

Userdef_SFLASH_Write_Enable
Init transmission setting of SPI multi I/O bus controller
Spibsc_transfer_write_enable()
Enables write to the serial flash memory to allow erase and write to the memory.
return

Figure 5.29 Enabling Serial Flash Memory Write (User Defined) Processing

5.10.22 Processing for Waiting for Ready of Serial Flash Memory (User Defined)

Figure 5.30 shows processing for waiting for ready of serial flash memory (user defined)

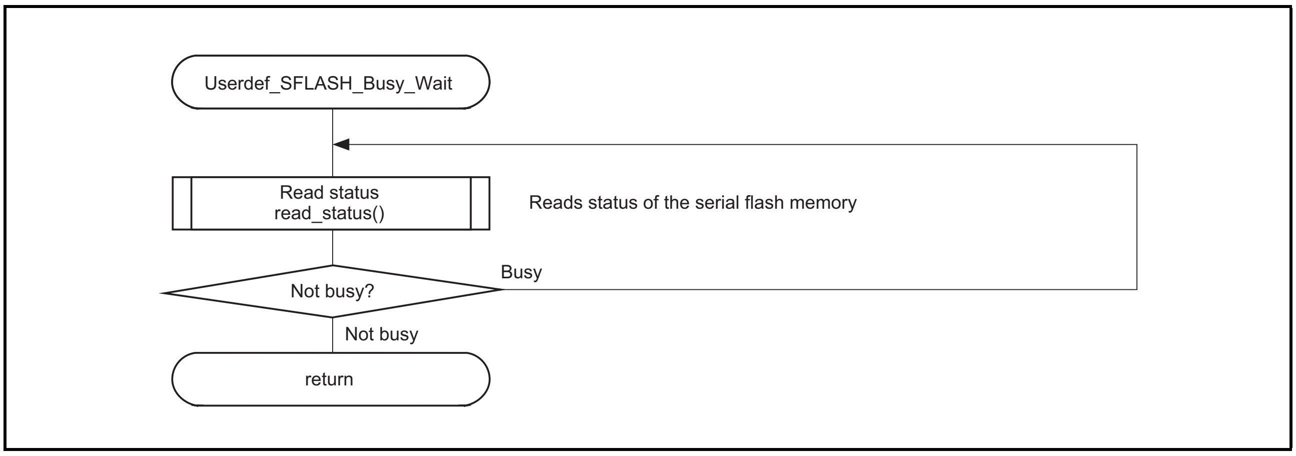


Figure 5.30 Processing for Waiting for Ready of Serial Flash Memory (User Defined)

5.10.23 SPIBSC External Address Mode Initial Settings

Figure 5.31 shows the flowchart of SPIBSC external address mode initial settings.

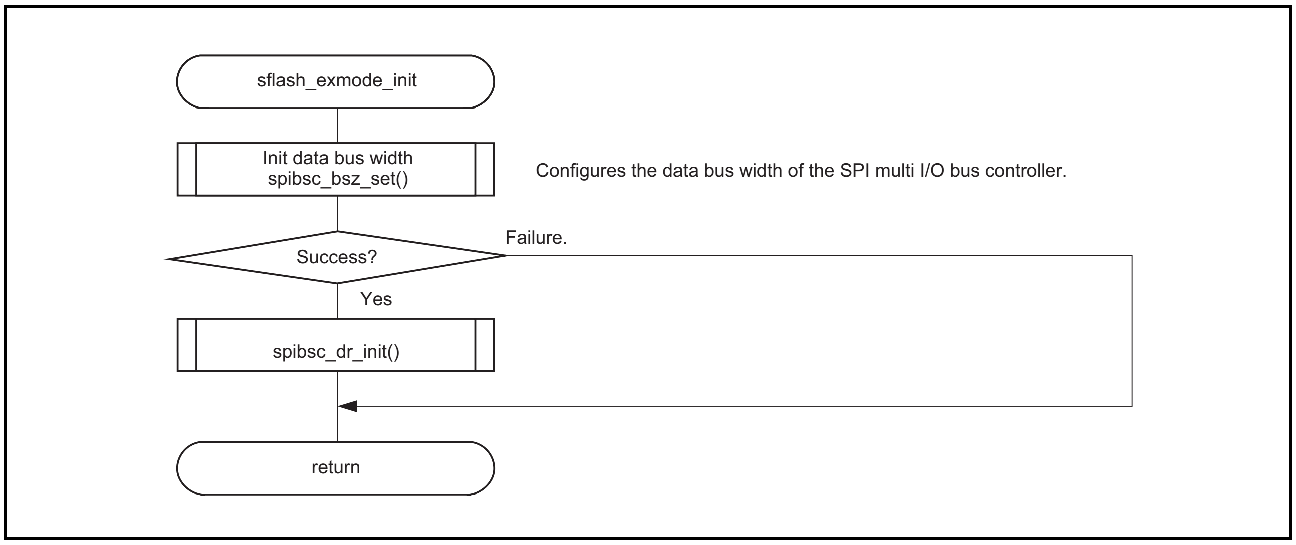


Figure 5.31 SPIBSC External Address Mode Initial Settings

5.10.24 SPIBSC Initial Settings

Figure 5.32 shows the SPIBSC initial setting flowchart.

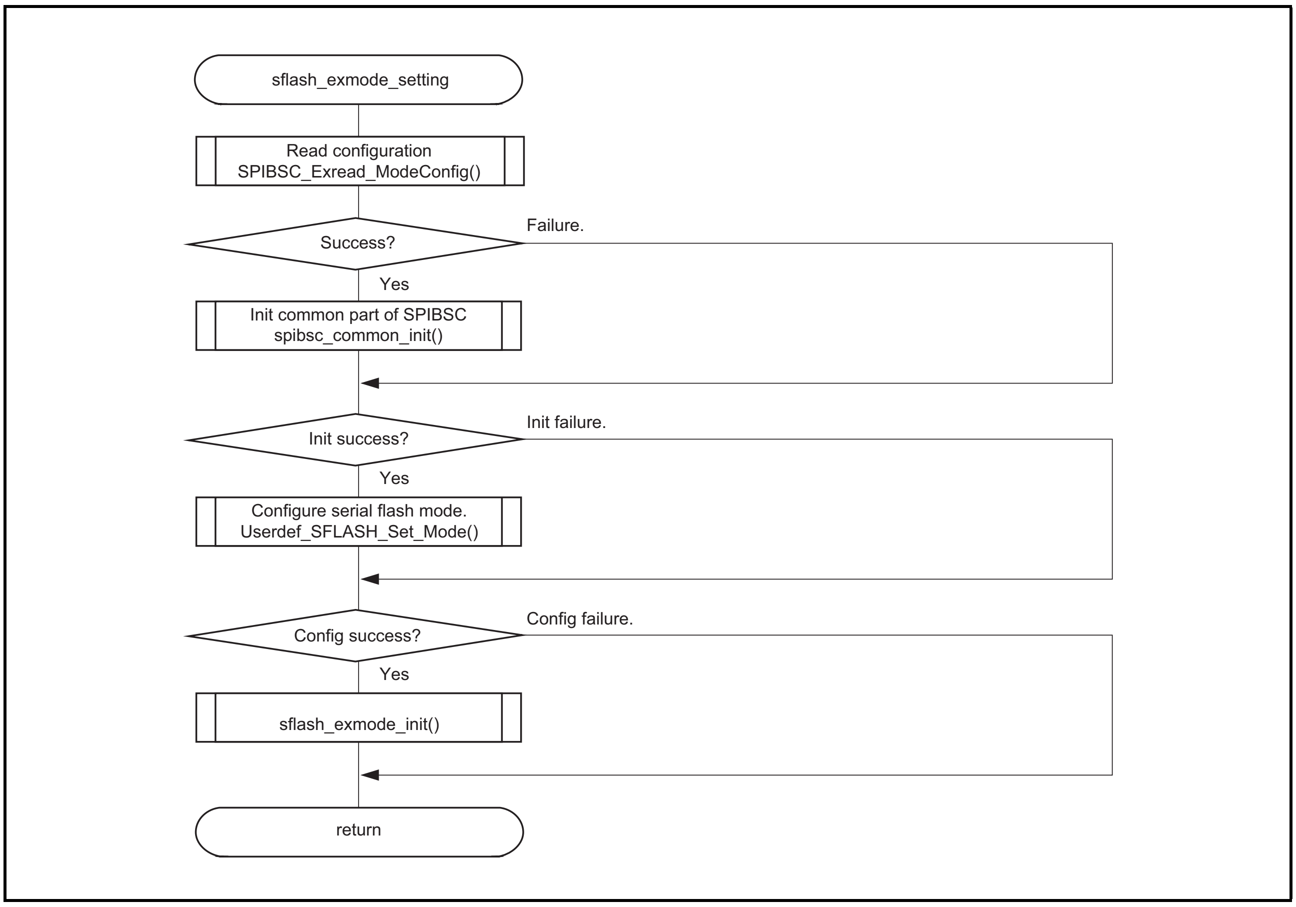


Figure 5.32 SPIBSC Initial Settings

5.10.25 SPIBSC Data Transfer Waiting Processing

Figure 5.33 shows the SPIBSC data transfer waiting processing flowchart.

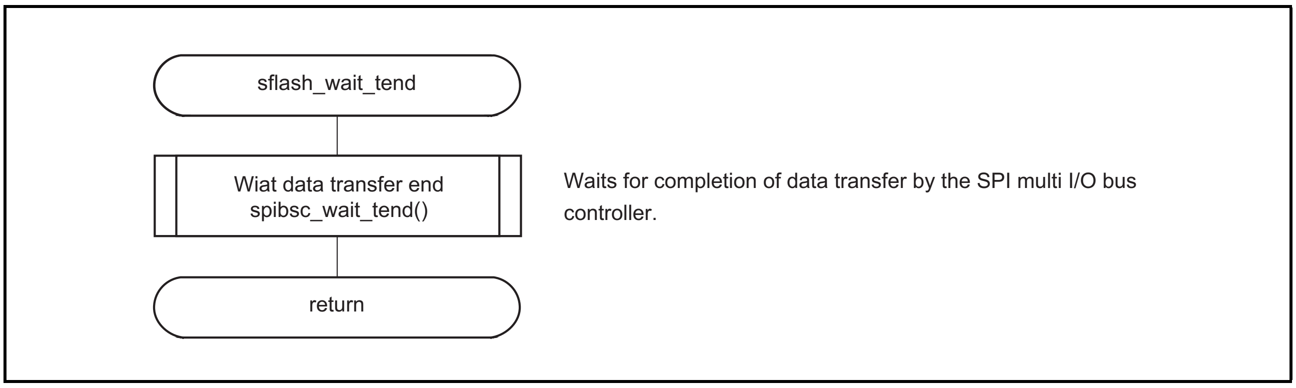

sflash_wait_tend
Wiat data transfer end
spibsc_wait_tend()
return
Waits for completion of data transfer by the SPI multi I/O bus controller.

Figure 5.33 SPIBSC Data Transfer Waiting Processing

5.10.26 SPIBSC External Address Mode Setting Processing

Figure 5.34 shows the SPIBSC external address mode setting processing flowchart.

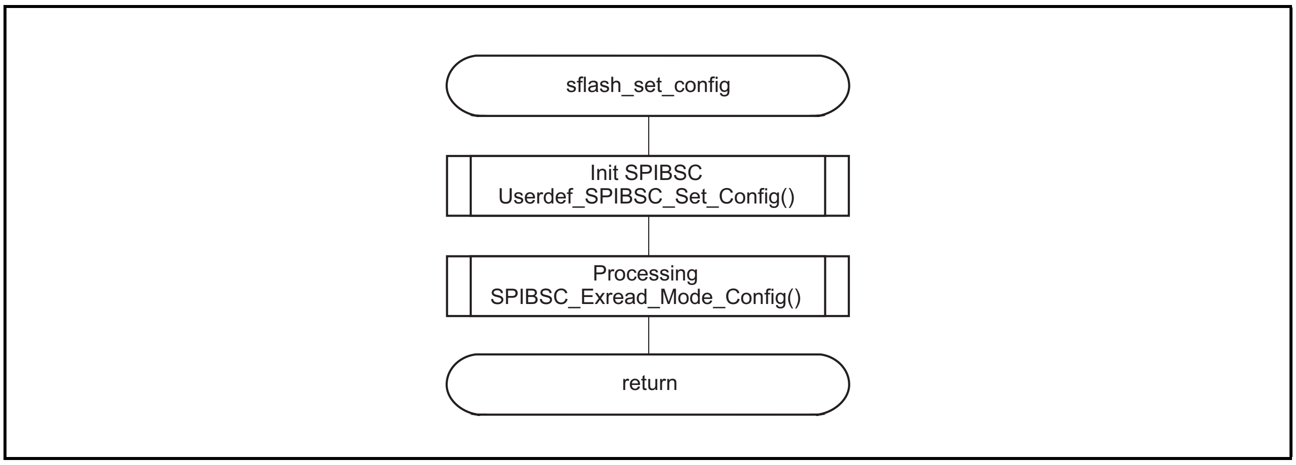


Figure 5.34 SPIBSC External Address Mode Setting Processing

5.10.27 SPIBSC Configuration Processing

Figure 5.35 shows the SPIBSC configuration processing flowchart.

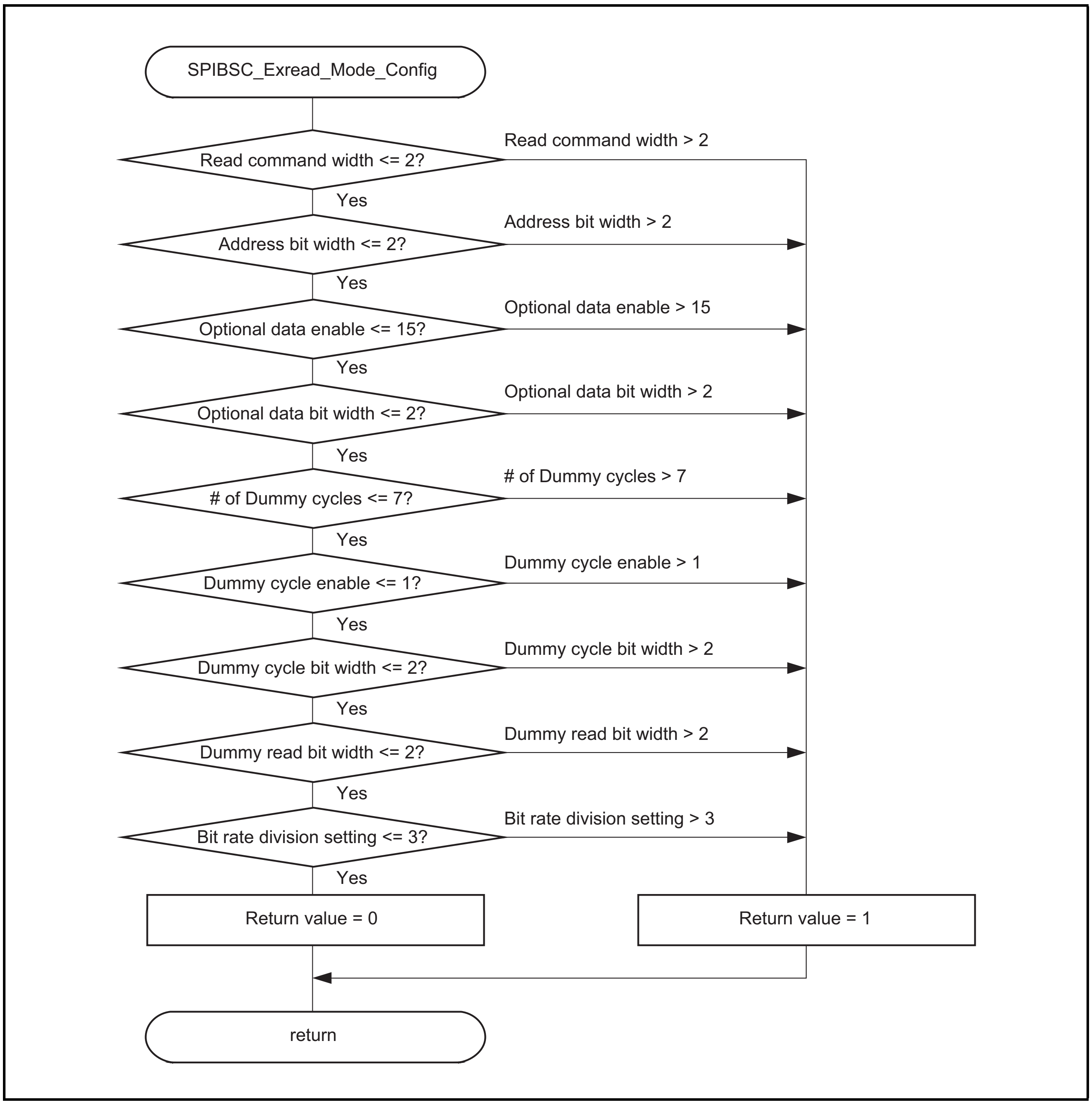

SPIBSC_Exread_Mode_Config
Read command width <= 2?
Read command width > 2
Yes
Address bit width <= 2?
Address bit width > 2
Yes
Optional data enable <= 15?
Optional data enable > 15
Yes
Optional data bit width <= 2?
Optional data bit width > 2
Yes
# of Dummy cycles <= 7?
# of Dummy cycles > 7
Yes
Dummy cycle enable <= 1?
Dummy cycle enable > 1
Yes
Dummy cycle bit width <= 2?
Dummy cycle bit width > 2
Yes
Dummy read bit width <= 2?
Dummy read bit width > 2
Yes
Bit rate division setting <= 3?
Bit rate division setting > 3
Yes
Return value = 0
Return value = 1
return

Figure 5.35 SPIBSC Configuration Processing

5.10.28 SPIBSC External Address Space Read Setting Processing (User Defined)

Figure 5.36 shows the flowchart of SPIBSC external address space read setting processing (user defined).

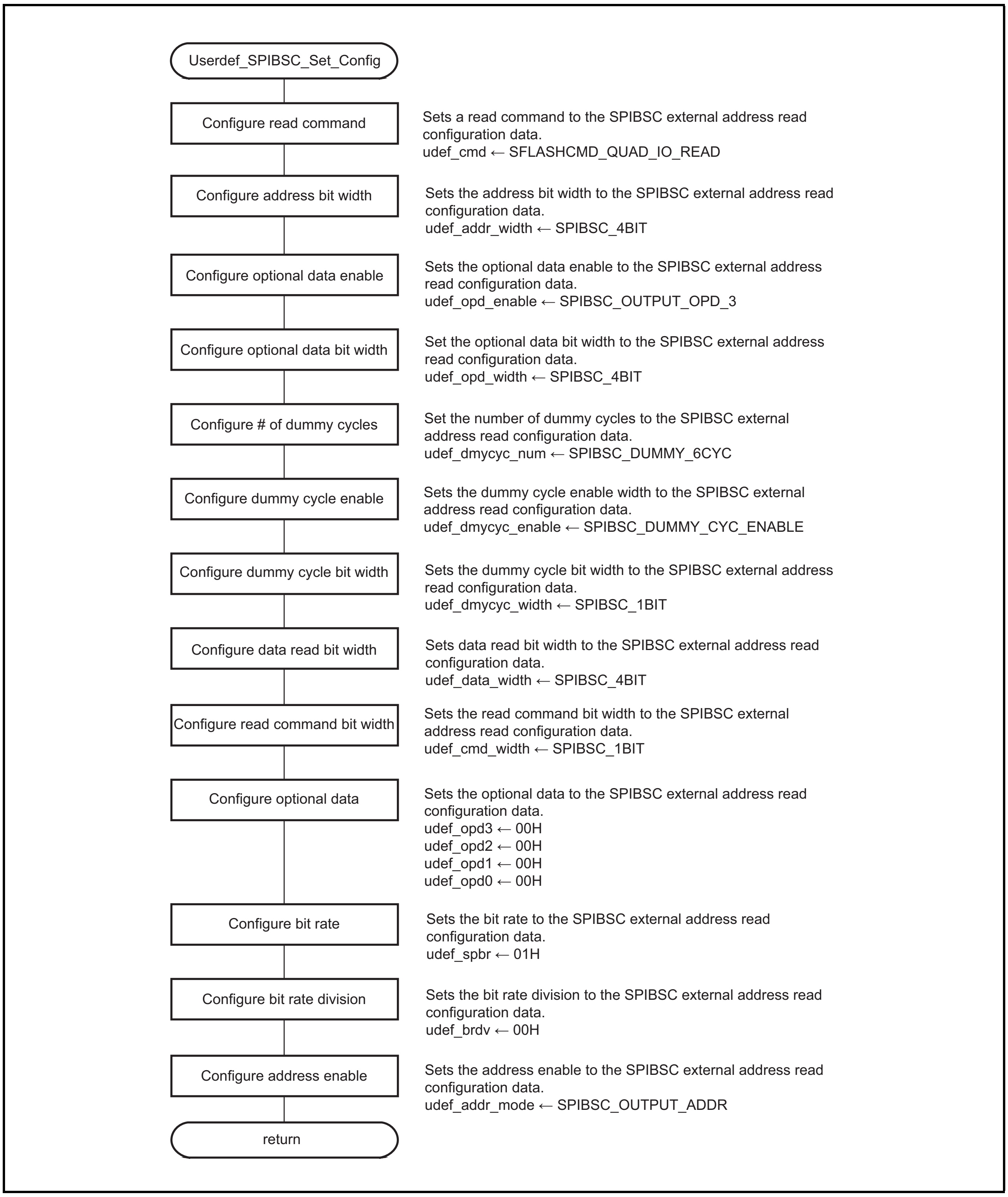

Userdef_SPIBSC_Set_Config
Configure read command
Sets a read command to the SPIBSC external address read configuration data.
udef_cmd ← SFLASHCMD_QUAD_IO_READ
Configure address bit width
Sets the address bit width to the SPIBSC external address read configuration data.
udef_addr_width ← SPIBSC_4BIT
Configure optional data enable
Sets the optional data enable to the SPIBSC external address read configuration data.
udef_opd_enable ← SPIBSC_OUTPUT_OPD_3
Configure optional data bit width
Set the optional data bit width to the SPIBSC external address read configuration data.
udef_opd_width ← SPIBSC_4BIT
Configure # of dummy cycles
Set the number of dummy cycles to the SPIBSC external address read configuration data.
udef_dmycyc_num ← SPIBSC_DUMMY_6CYC
Configure dummy cycle enable
Sets the dummy cycle enable width to the SPIBSC external address read configuration data.
udef_dmycyc_enable ← SPIBSC_DUMMY_CYC_ENABLE
Configure dummy cycle bit width
Sets the dummy cycle bit width to the SPIBSC external address read configuration data.
udef_dmycyc_width ← SPIBSC_1BIT
Configure data read bit width
Sets data read bit width to the SPIBSC external address read configuration data.
udef_data_width ← SPIBSC_4BIT
Configure read command bit width
Sets the read command bit width to the SPIBSC external address read configuration data.
udef_cmd_width ← SPIBSC_1BIT
Configure optional data
Sets the optional data to the SPIBSC external address read configuration data.
udef_opd3 ← 00H
udef_opd2 ← 00H
udef_opd1 ← 00H
udef_opd0 ← 00H
Configure bit rate
Sets the bit rate to the SPIBSC external address read configuration data.
udef_spbr ← 01H
Configure bit rate division
Sets the bit rate division to the SPIBSC external address read configuration data.
udef_brdv ← 00H
Configure address enable
Sets the address enable to the SPIBSC external address read configuration data.
udef_addr_mode ← SPIBSC_OUTPUT_ADDR
return

Figure 5.36 SPIBSC External Address Space Read Setting Processing (User Defined)

5.10.29 IRQ9 Interrupt (IRQ Pin Interrupt 5) Processing

Figure 5.37 shows the IRQ9 interrupt (IRQ pin interrupt 5) processing flowchart.

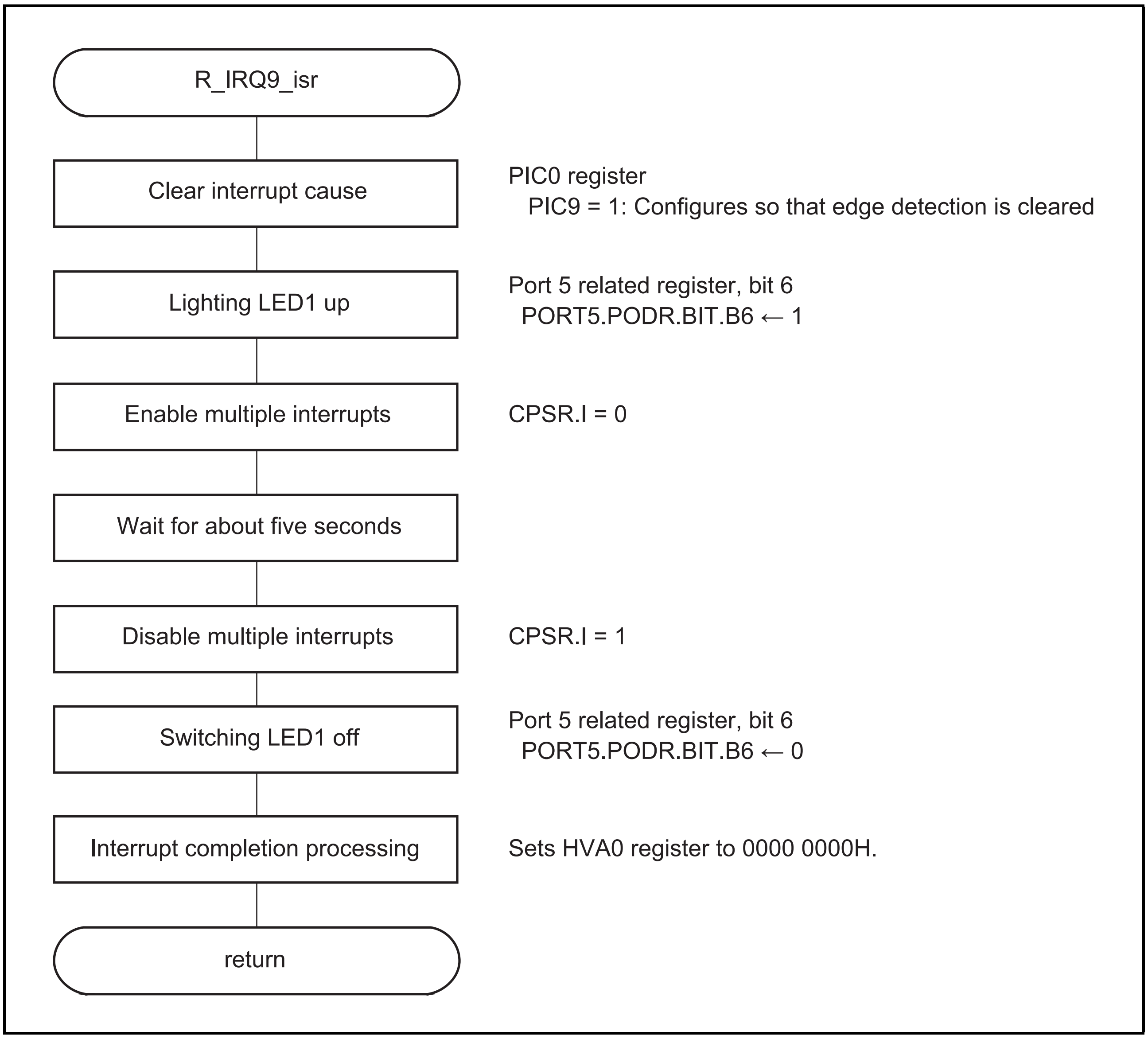
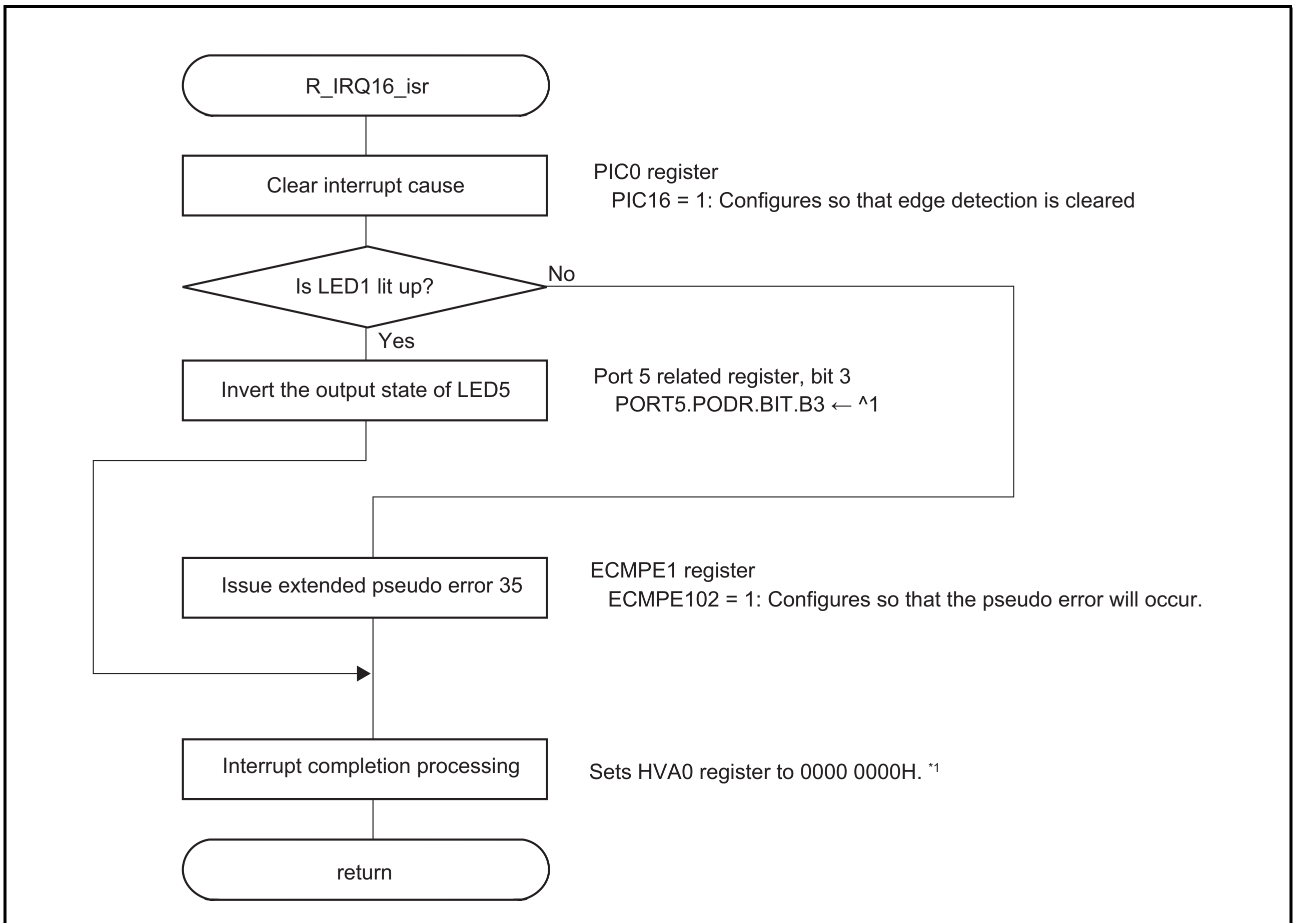

R_IRQ9_isr
Clear interrupt cause
PIC0 register
PIC9 = 1: Configures so that edge detection is cleared
Lighting LED1 up
Port 5 related register, bit 6
PORT5.PODR.BIT.B6 ← 1
Enable multiple interrupts
CPSR.I = 0
Wait for about five seconds
Disable multiple interrupts
CPSR.I = 1
Switching LED1 off
Port 5 related register, bit 6
PORT5.PODR.BIT.B6 ← 0
Interrupt completion processing
Sets HVA0 register to 0000 0000H.
return

Figure 5.37 IRQ9 Interrupt (IRQ Pin Interrupt 5) Processing

5.10.30 IRQ16 Interrupt (IRQ Pin Interrupt 12) Processing

Figure 5.38 shows the IRQ16 interrupt (IRQ pin interrupt 12) processing flowchart.



R_IRQ16_isr
Clear interrupt cause
PIC0 register
PIC16 = 1: Configures so that edge detection is cleared
Is LED1 lit up?
No
Yes
Invert the output state of LED5
Port 5 related register, bit 3
PORT5.PODR.BIT.B3 ← ^1
Issue extended pseudo error 35
ECMPE1 register
ECMPE102 = 1: Configures so that the pseudo error will occur.
Interrupt completion processing
Sets HVA0 register to 0000 0000H. *1
return

Note 1. This processing is not executed when extended pseudo error 35 is produced, because the error leads to an ECM reset immediately beforehand.

Figure 5.38 IRQ16 Interrupt (IRQ Pin Interrupt 12) Processing

6. Sample Program

Download the sample program from the Renesas Electronics website.

7. Related Documents

- User's Manual: Hardware
  RZ/T1 Group User's Manual: Hardware
  Download the latest version from the Renesas Electronics website.

  RZ/T1 Evaluation Board RTK7910022C00000BR User's Manual
  Download the latest version from the Renesas Electronics website.

- Technical Update/Technical News
  Download the latest version from the Renesas Electronics website.

- User's Manual: Development Environment
  The latest version for the IAR integrated development environment (IAR Embedded Workbench® for Arm) is available from the IAR Systems website.
  The latest version for the Arm integrated development environment (Development Studio 5TM) is available from the Arm website.
  The latest version for the Renesas Electronics integrated development environment (e2studio) is available from the Renesas Electronics website.

Appendix 1. Supplementary Notes on Development Environments

This section shows the steps up to the start of debugging of the sample program in each of the available development environments and includes notes on the porting of source code to other development environments.

< Steps up to debugging of the sample program (when booting is through the SPI)>

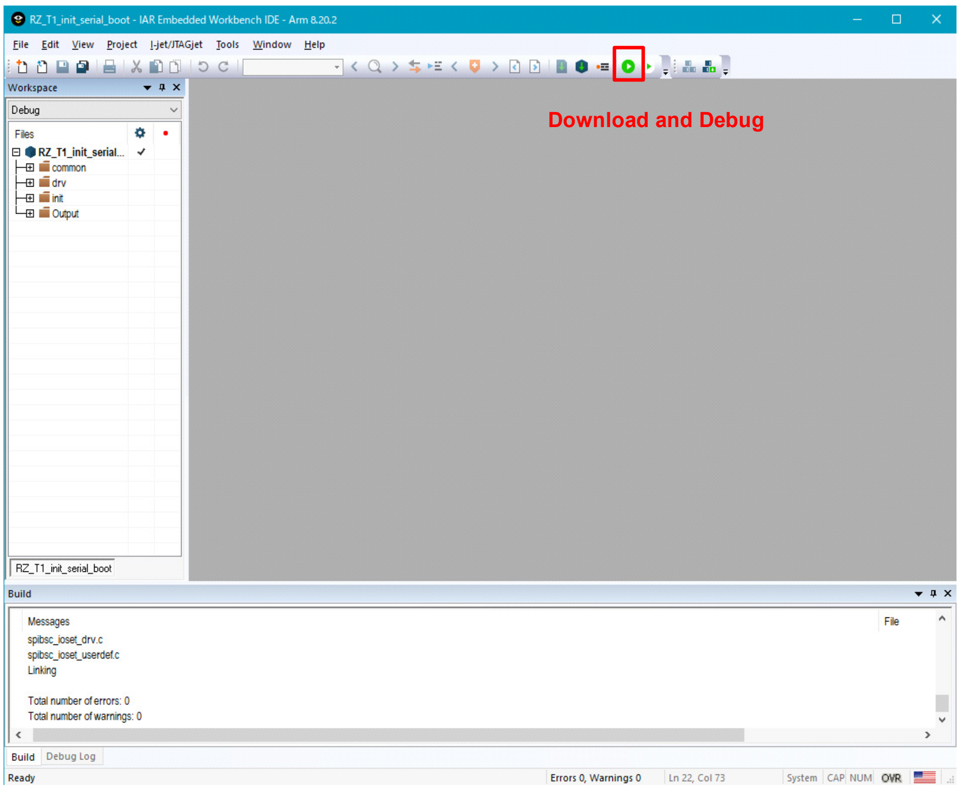
(EWARM from IAR systems)

(1) Start up the EWARM environment. Go to [File] > [Open] > [Workspace] and then specify RZ_T1_init_serial_boot.eww.

(2) Select the [Rebuild All] item from the [Project] menu to rebuild the project.

(3) While the RZ/T1 evaluation board and I-jet are connected, click on the download and debug button in the [Project] toolbar.

(4) After the emulator is connected, the program will be written to the external serial flash memory by the dedicated flash downloader. Debugging can then start.



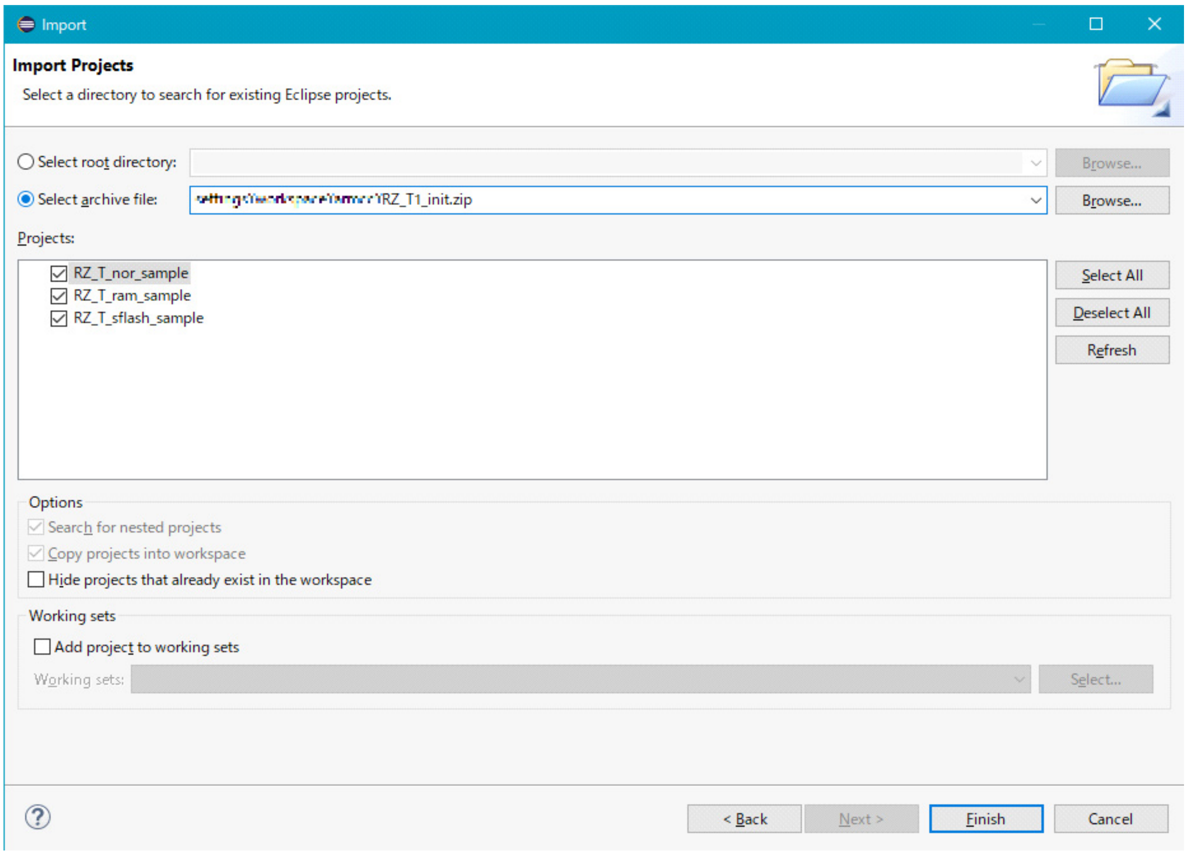
```
Messages
spibsc_ioset_drv.c
spibsc_ioset_userdef.c
Linking

Total number of errors: 0
Total number of warnings: 0
```

Note: When using the sample program to be run from RAM, the program is directly downloaded from your PC to the device's tightly coupled memory (TCM), so the dedicated flash downloader is not used.

(DS-5 from Arm)

(1) Startup the DS-5 environment. Go to [File] > [Import]. On the [Import] window, select [Existing Projects into Workspace] in the [General] folder and click the [Next] button.

(2) Select the [Select archive file:] radio button and click on the [Browse…] button. Select the compressed program file “RZ_T1_initial.zip” on the list in the window and click on the [Finish] button.



Import Projects

Select a directory to search for existing Eclipse projects.

( ) Select root directory:

(●) Select archive file: [illegible]\RZ_T1_init.zip

Projects:

- ☑ RZ_T_nor_sample
- ☑ RZ_T_ram_sample
- ☑ RZ_T_sflash_sample

Options

- ☑ Search for nested projects
- ☑ Copy projects into workspace
- ☐ Hide projects that already exist in the workspace

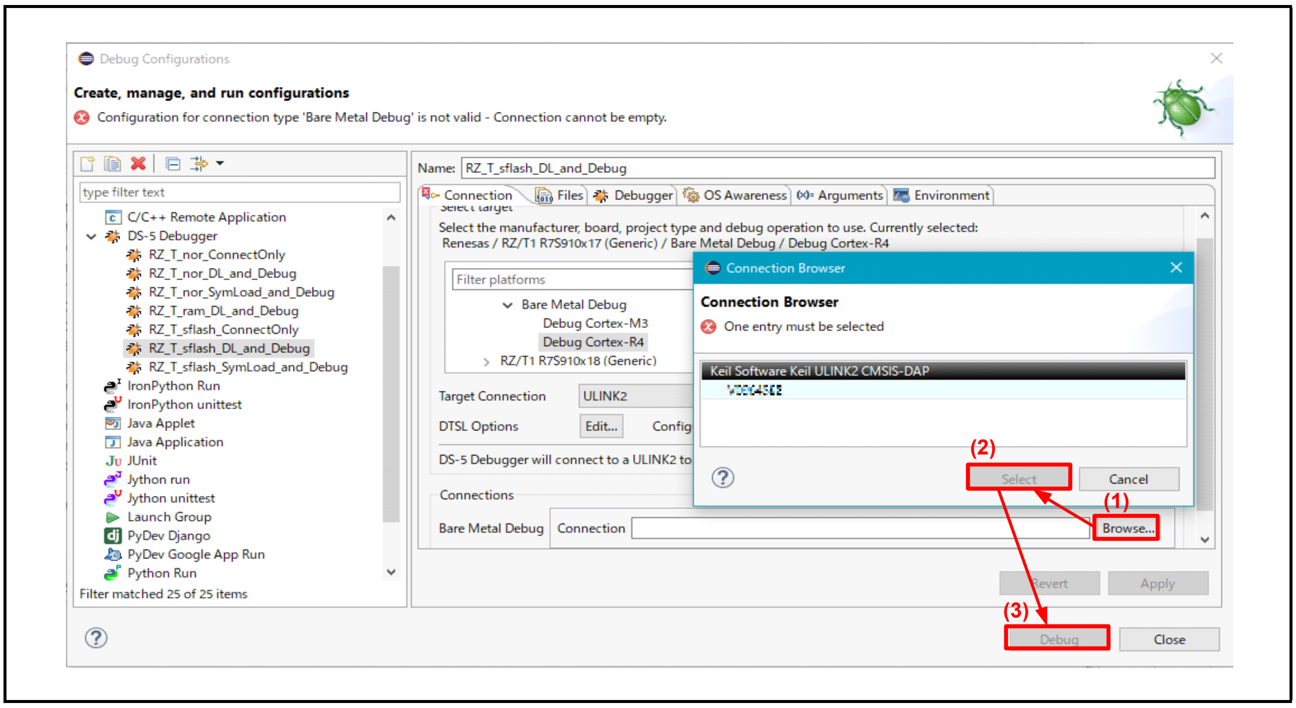
Working sets

- ☐ Add project to working sets

Working sets:

(3) Select [Build All] from the [Project] menu.

(4) Select the configuration window for "RZ_T_sflash_DL_and_Debug". While the RZ/T1 evaluation board and ULINK2 are connected, click on [Browse] and open the connection browser window as shown below. Select the target connection from the list in the window. Click on [Debug] in the debug configurations window and start debugging.


Debug Configurations

Create, manage, and run configurations

Configuration for connection type 'Bare Metal Debug' is not valid - Connection cannot be empty.

Name: RZ_T_sflash_DL_and_Debug

Select the manufacturer, board, project type and debug operation to use. Currently selected: Renesas / RZ/T1 R7S910x17 (Generic) / Bare Metal Debug / Debug Cortex-R4

Target Connection: ULINK2

DS-5 Debugger will connect to a ULINK2 to

Connection Browser

Connection Browser

One entry must be selected

Keil Software Keil ULINK2 CMSIS-DAP

[redacted]

Filter matched 25 of 25 items

(5) After the emulator is connected, RZ_T_sflash_sample.ds is executed and the program will be written to the external serial flash memory by using the "semihosting" functionality. Refer to the following application notes for details on writing to the flash memory by using the "semihosting" functionality.

- Example of Downloading to NOR Flash Memory by Using "Semihosting" of Arm® Development Studio 5 (DS-5TM)
- Example of Downloading to Serial Flash Memory by Using "Semihosting" of Arm® Development Studio 5 (DS-5TM)

(6) On completion of writing to the flash memory by the script, the message "Flash Programming Complete" appears in the application console window. Debugging can then start.

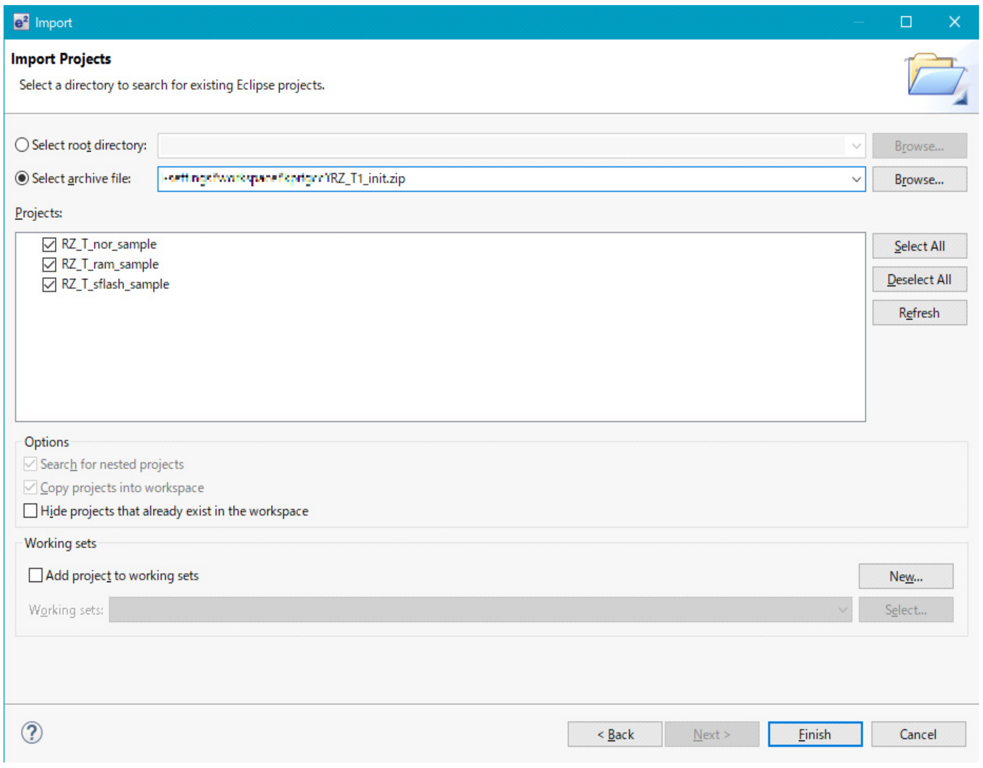
Note

Writing to the external serial flash can be skipped in the subsequent debugging connections if they do not involve change of the program. In this case, select "RZ_T_sflash_SymLoad_and_Debug" at step (4), so only download the symbol information and start debugging.

(e2 studio from RENESAS)

(1) Start up the e2 studio environment. In the workspace, go to [File] > [Import]. On the [Import] window, select [Existing Projects into Workspace] in the [General] folder and click the [Next] button.

(2) Select the [Select archive file:] radio button and click on the [Browse..] button. Select the compressed program file “RZ_T1_initial.zip” on the list in the window and click on the [Finish] button.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

○ Select root directory: [ ] Browse...

◉ Select archive file: [illegible]RZ_T1_init.zip Browse...

Projects:

- ☑ RZ_T_nor_sample
- ☑ RZ_T_ram_sample
- ☑ RZ_T_sflash_sample

Select All / Deselect All / Refresh

Options

- ☑ Search for nested projects
- ☑ Copy projects into workspace
- ☐ Hide projects that already exist in the workspace

Working sets

- ☐ Add project to working sets

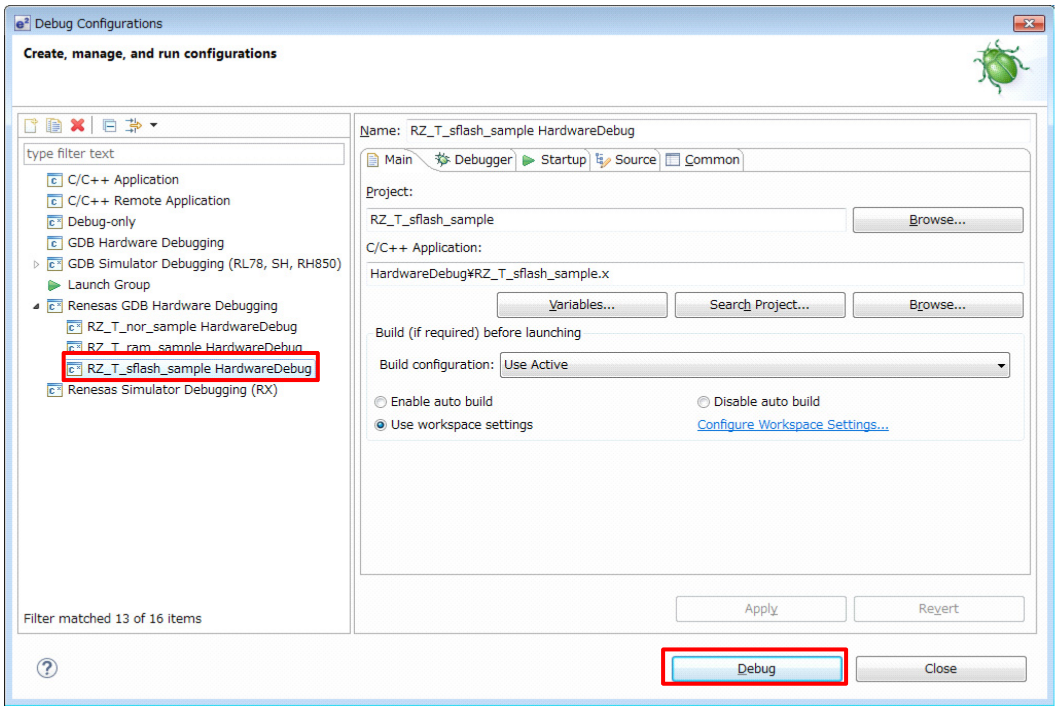
New...

Working sets: Select...

< Back | Next > | Finish | Cancel

(3) Select [Build All] from the [Project] menu.

(4) Select the configuration window for “RZ_T_sflash_sample_HardwareDebug” . While the RZ/T1 evaluation board and J-LINK are connected, click on [Debug] in the Debug Configurations window and start debugging.



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Remote Application
- Debug-only
- GDB Hardware Debugging
- GDB Simulator Debugging (RL78, SH, RH850)
- Launch Group
- Renesas GDB Hardware Debugging
  - RZ_T_nor_sample HardwareDebug
  - RZ_T_ram_sample HardwareDebug
  - RZ_T_sflash_sample HardwareDebug
- Renesas Simulator Debugging (RX)

Filter matched 13 of 16 items

Name: RZ_T_sflash_sample HardwareDebug

Main | Debugger | Startup | Source | Common

Project:
RZ_T_sflash_sample

C/C++ Application:
HardwareDebug¥RZ_T_sflash_sample.x

Build (if required) before launching

Build configuration: Use Active

Enable auto build
Disable auto build
Use workspace settings
Configure Workspace Settings...

(5) After the emulator is connected, the program will be written to the external serial flash memory by the dedicated flash downloader. Debugging can then start.

<Note on porting source code>

Points on porting source code from one development environment to another are given using statements from the sample program referred to in this document as examples.

Note: This is only an example. For detailed settings including extended settings, refer to the documents for the individual development environments.

(1) Specifying interrupt functions (specifying the IRQ interrupt function as an example)

EWARM from IAR Systems	DS-5 from Arm	e2 studio from RENESAS
#pragma type_attribute=__irq __arm void R_IRQ_isr (void);	__irq void R_IRQ_isr (void);	void R_IRQ_isr (void) __attribute__ ((interrupt));

(2) Inline assembler-language statements (taking the nop instruction as an example)

EWARM from IAR Systems	DS-5 from Arm	e2 studio from RENESAS
asm volatile ("nop");	__asm("nop");	asm("nop");

(3) Specifying a section in source code (example of specifying a parameter for the loader)

EWARM from IAR Systems	DS-5 from Arm	e2 studio from RENESAS
#pragma location="ldr_param" __root const uint32_t loader_param_SPI[19] = {	#pragma arm section rodata = "CONST_LOADER_TABLE" const stBOOT_SPIBSC spibsc_loader_table = {	const unsigned int Loader_Param[] __attribute__ ((section (".loader_param"))) = {

Note: Regarding assignment to sections, refer to the linker settings of the development environment you are using in addition to the examples given above.

Website and Support

Renesas Electronics website

http://www.renesas.com/

Inquiries

http://www.renesas.com/inquiry

Revision History	Application Note: Initial Settings

Rev.	Date	Description	
		Page	Summary
0.20	Mar. 27, 2015	—	First Edition issued
1.00	Apr. 10, 2015	—	Only the revision number was changed to be posted on a website.
1.10	Aug. 31, 2015	2. Operating Environment	
		6	Table 2.1 Operating Environment: Integrated Development Environment, partially amended and added
		5. Software	
		11	Description added
		12	5.1.1 Project Setting: Table5.1 amended
		13	5.1.2 Description added
		17	5.2.2 MPU Settings: Table5.4 amended
		19	5.2.4 Required Memory Size: Description and reference added
		19	Table 5.6: Table title was partially amended
		20	Table 5.7 added
		21	Table 5.8 added
		7. Related Documents	
		65	User's Manual: Development Environment, content added
		Appendix-1 Supplements on Development Environments	
		66 to 73	Appendix 1. Supplements on Development Environments added
1.20	Dec.04, 2015	2. Operating Environment	
		6	Table 2.1 Operating Environment: Integrated Development Environment, information partially amended
		5. Software	
		50	5.10.12 ICU Setting Processing: Figure 5.20, contents added
		Appendix-1 Supplements on Development Environments	
		67 to 70	<Steps up to debugging of the sample program (when booting is through the SPI>: Contents of DS-5 from ARM and e2 studio from Renesas are partially amended.
1.30	Jun.06, 2017	2. Operating Environment	
		6	Table 2.1 Operating Environment: The description of the integrated development environment, modified
		4. Hardware	
		10	Table 4.1 Pins and Functions: P53 pin, added
		5. Software	
		11	5.1 Operation Overview: The description, modified
		11	Figure 5.1 Operation overview after boot process: The processing, modified. Note 3, added.
		—	5.2.4 Required Memory Size, deleted
		19	Table 5.6 Interrupts Used in the Sample Program: The process outline of IRQ pin interrupt 5 (IRQ9), modified. The priority and process outline of IRQ pin interrupt 12 (IRQ16), modified.
		24	Table 5.10 Functions: The names of functions, modified (atcm_waitset, cpg_pll_wait, sflash_exmode_init, sflash_exmode_setting, sflash_wait_tend, sflash_set_config)
		27	5.9.10 port_init: The description, modified
		28	5.9.13 atcm_waitset: The name of the function, modified
		29	5.9.14 cpg_pll_wait: The name of the function, modified
		32	5.9.23 sflash_exmode_init: The name of the function, modified
		33	5.9.24 sflash_exmode_setting: The name of the function, modified
		33	5.9.25 sflash_wait_tend: The name of the function, modified
		34	5.9.26 sflash_set_config: The name of the function, modified
		35	5.9.29 R_IRQ9_isr: The description, modified
		35	5.9.30 R_IRQ16_isr: The description and remarks, modified
		38	Figure 5.8 Main Processing: P53 added
		39	Figure 5.9 Reset Flag Detection Processing: The name of the function, modified

Rev.	Date	Description	
		Page	Summary
1.30	Jun.06, 2017	40	Figure 5.10 Reset Flag Detection Processing: The name of the function, modified
		41	Figure 5.11 Clock Setting Processing: The name of the function, modified
		42	Figure 5.12 Bus Setting Processing (1): The name of the function, modified
		46	Figure 5.18 Port Setting Processing: The processing, added. The defined processing, modified.
		48	Figure 5.21 ATCM Access Wait Setting Processing: The name of the function, modified
		49	Figure 5.22 PLL Stability Waiting (100us) Processing: The name of the function, modified
		50	Figure 5.23 ECM Initial Settings: The setting of the ECM error mask register, modified
		56	Figure 5.31 SPIBSC External Address Mode Initial Settings: The name of the function, modified
		56	Figure 5.32 SPIBSC Initial Settings: The name of the function, modified
		57	Figure 5.33 SPIBSC Data Transfer Waiting Processing: The name of the function, modified
		57	Figure 5.34 SPIBSC External Address Mode Setting Processing: The name of the function, modified
		60	Figure 5.37 IRQ9 Interrupt (IRQ Pin Interrupt 5) Processing: The processing, added
		61	Figure 5.38 IRQ16 Interrupt (IRQ Pin Interrupt 12) Processing: The judgment and processing, added. Note, added.
1.40	Apr.27, 2018	All	"ARM" changed to "Arm"
		Introduction	
		1	The error, corrected
		2. Operating Environment	
		6	Table 2.1 Operating Environment: The description and version numbers of the integrated development environments from IAR Systems, Arm, and Renesas, modified
		5. Software	
		11	Figure 5.1 Operation overview after boot process: The errors, corrected
		12	5.1.1 Project Setting: The error, corrected
		13	5.1.2 Preparation: The description, modified
		15	Table 5.3 Sections to Be Used: The errors in Note 1 and Note 2, corrected
		16	Figure 5.3 Section Assignment (for 16-bit bus boot version): The error in Note 2, corrected
		18	Table 5.5 Exception Processing Vector Table: The errors in the table, corrected; the description of "Interrupts" under the table, modified
		19	5.4 Fixed-Width Integer Types; The description, modified
		19	Table 5.7 Fixed-width Integers Used in the Sample Program: The figure title, modified
		26	5.9.5 bsc_init: The description, modified
		27	5.9.7 cache_init: The description, modified
		28	5.9.12 icu_init: The description, modified
		28	5.9.13 atcm_waitset: The description, modified
		29	5.9.16 R_ICU_Enable: The description modified
		30	5.9.17 R_ICU_Disable: The description, modified
		30	5.9.18 R_ICU_ExtPinInit: The description and argument uint32_t dnf_set, modified
		31	5.9.19 R_ICU_Regist: The description, modified
		31	5.9.20 Userdef_SFLASH_Set_Mode: The synopsis, argument uint32_t addr_mode, and remarks, modified
		32	5.9.21 Userdef_SFLASH_Write_Enable: The remarks, modified
		32	5.9.22 Userdef_SFLASH_Busy_Wait: The remarks, modified
		32	5.9.23 sflash_exmode_init: The remarks, modified
		33	5.9.24 sflash_exmode_setting: The remarks, modified
		33	5.9.25 sflash_wait_tend: The remarks, modified
		34	5.9.26 sflash_set_config: The remarks, modified
		34	5.9.27 SPIBSC_Exread_Mode_Config: The remarks, modified
		34	5.9.28 Userdef_SPIBSC_Set_Config: The remarks, modified

Revision History	Application Note: Initial Settings

Rev.	Date	Description	
		Page	**Summary**
1.40	Apr.27, 2018	6. Sample Program	
		62	The description, modified
		7. Related Documents	
		63	The description, modified
		Appendix 1. Supplementary Notes on Development Environments	
		64	The title, modified
		64	< Steps up to debugging of the sample program (when booting is through the SPI)>, (EWARM from IAR systems) (4): The figure, changed
		65	< Steps up to debugging of the sample program (when booting is through the SPI)>, (DS-5 from Arm) (2): The figure, changed
		66	< Steps up to debugging of the sample program (when booting is through the SPI)>, (DS-5 from Arm) (4): The figure, changed
		67	< Steps up to debugging of the sample program (when booting is through the SPI)>, (e2 studio from RENESAS) (2): The figure, changed
		68	< Steps up to debugging of the sample program (when booting is through the SPI)>, (e2 studio from RENESAS) (4): The figure, changed
		69	<Note on porting source code> (2): The inline assembler of EWARM from IAR Systems, modified
1.41	Jul. 13, 2018	5. Software	
		36	Figure 5.6 Loader Program 1 Processing: Processing for initialization of variables with no initial values, added
		37	Figure 5.7 Loader Program 2 Processing: Processing for initialization of variables with no initial values, added

General Precautions in the Handling of MPU/MCU Products

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

1. 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 LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.
   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the 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.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338