RL78/G13

EEPROM Emulation Library Pack01

R01AN1361EJ0110
Rev. 1.10
June 01, 2016

Introduction

This application note explains how to write and read data to and from data flash memory using the EEPROM Emulation Library Pack01 (Flash Data Library (FDL) and EEPROM Emulation Library (EEL)).

Target Device

RL78/G13

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 of the modified program.

Contents

1. Specifications

This application note explains how to use the EEPROM Emulation Library Pack01.

The sample program covered in this document displays the write value, the target write data ID, and the current value of each data ID on the LCD. The sample program can be manipulated using three switches which are used to set the write value, set the target write data ID, and program into the data flash memory, respectively. Data in the data flash memory is erased by pressing the two switches for setting the write value and target write data ID at the same time for one second. The display on the LCD is updated every time the write value and the target write data ID are changed. The current value of the data ID is updated after power is turned on, data is programmed, or it is erased.

Table 1.1 lists the peripheral functions to be used and their uses.

Table 1.1 Peripheral Functions to be Used and their Uses

Peripheral Function	Use
Port I/O	Displays text on the LCD. Turns on and off LED0.
Interval timer	Generates the wait time for avoiding chatters.
External interrupt input (INTP1)	Sets the write data. Erases the data flash memory (pressed long together with INTP2).
External interrupt input (INTP2)	Sets the write data ID. Erases the data flash memory (pressed long together with INTP1).
External interrupt input (INTP4)	Executes programming.

Figure 1.1 shows the outline of the operation to display the value on the LCD.

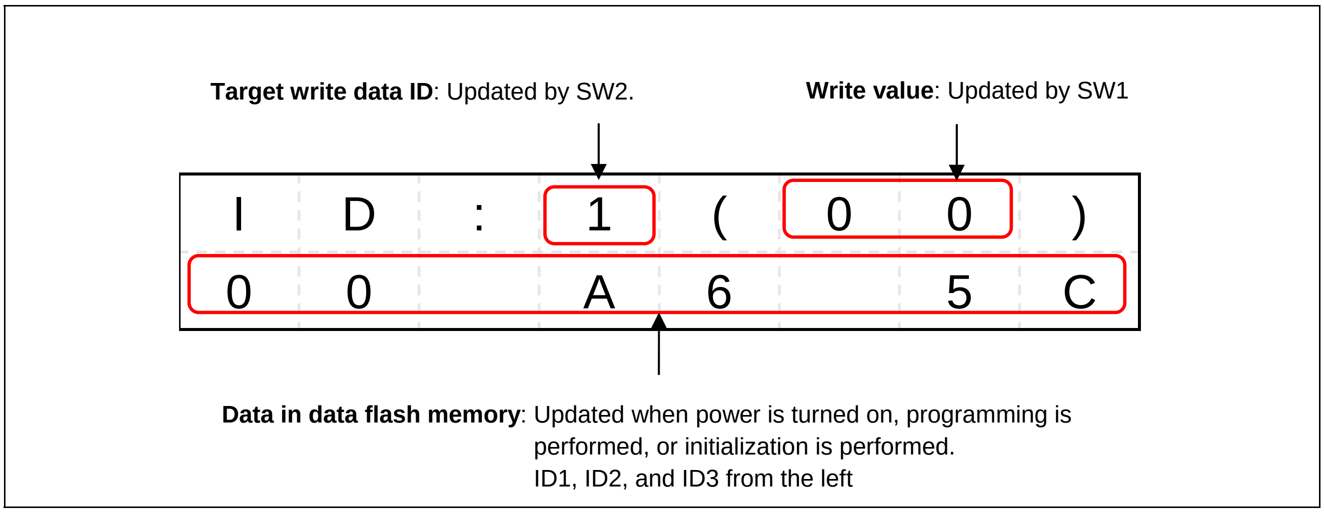

Target write data ID: Updated by SW2.
Write value: Updated by SW1
I D : 1 (0 0)
0 0 A 6 5 C
Data in data flash memory: Updated when power is turned on, programming is performed, or initialization is performed.
ID1, ID2, and ID3 from the left

Figure 1.1 Operation to Display Value on the LCD

1.1 Outline of EEPROM Emulation

EEPROM emulation is a function to store data in the flash memory that is installed as if it were EEPROM. EEPROM emulation uses FDL and EEL to carry out the write and read functions to and from data flash memory.

FDL is a software library for manipulating data flash memory. EEL is a software library for a user program to execute the EEPROM emulation function.

The user can use the EEPROM emulation function without being aware of any manipulations on data flash memory by calling the functions that EEL provides.

The EEPROM Emulation Library Pack01 allows the user to assign a 1-byte identifier (data ID: 1 to 255) to each block of data and to perform read or write operations in units of 1 to 255 bytes for each ID that is assigned (a maximum of 255 data items can be assigned to an ID).

Four or more contiguous data flash memory blocks are used to store data. This is called the EEPROM emulation block.

The data to be programmed by EEPROM emulation is divided into the reference data that is used for reference purposes and the user data that is specified by the user. The programming of the reference data starts at the lowest address of the emulation block and that of the user data at the highest address of the block.

A sample mapping of user data that is programmed into data flash memory is shown below. The assumed programming order is: user data A → user data B → user data A → user data C.

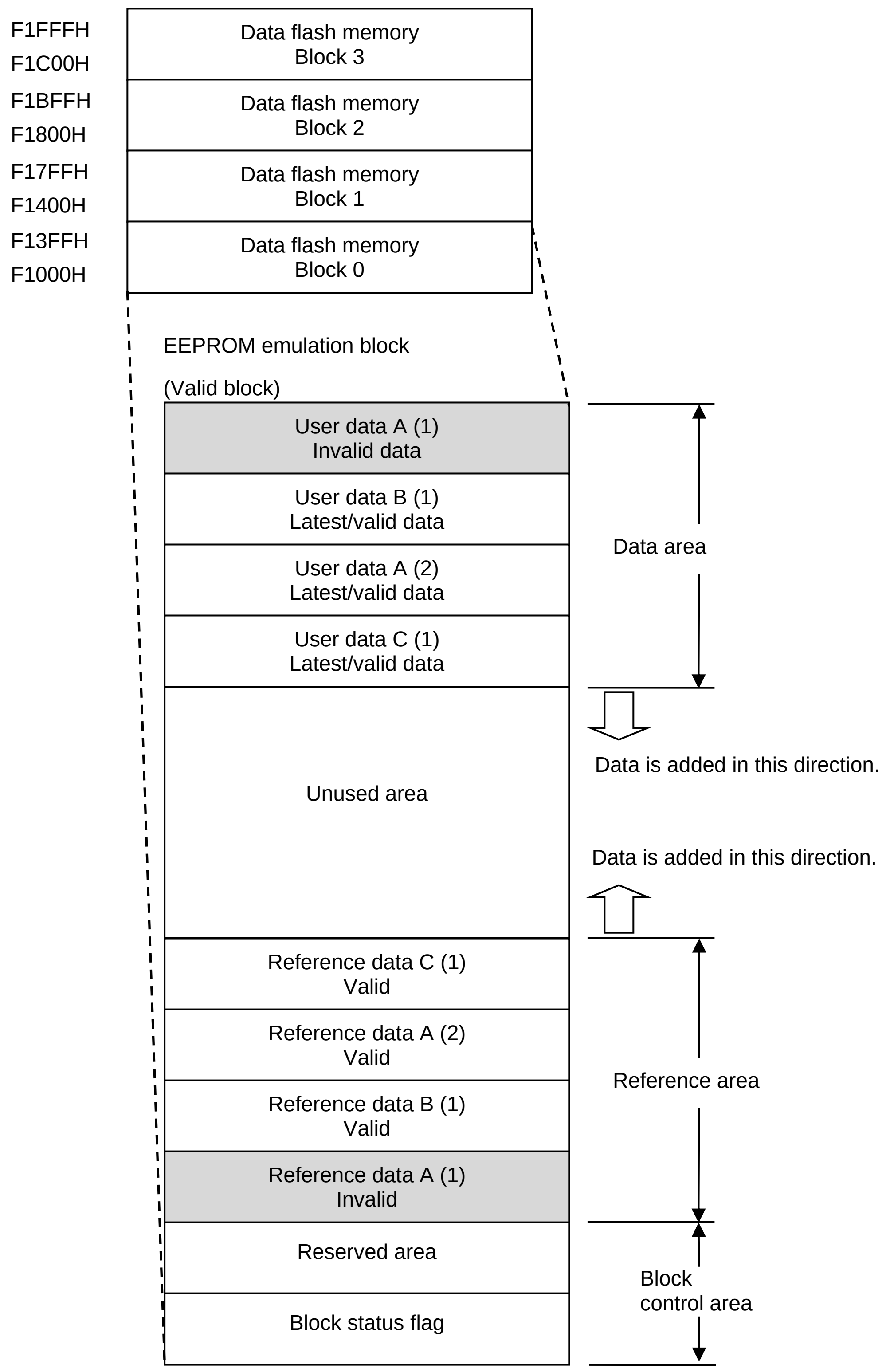

F1FFFH
F1C00H
Data flash memory Block 3
F1BFFH
F1800H
Data flash memory Block 2
F17FFH
F1400H
Data flash memory Block 1
F13FFH
F1000H
Data flash memory Block 0
EEPROM emulation block
(Valid block)
User data A (1) Invalid data
User data B (1) Latest/valid data
User data A (2) Latest/valid data
User data C (1) Latest/valid data
Data area
Data is added in this direction.
Unused area
Data is added in this direction.
Reference data C (1) Valid
Reference data A (2) Valid
Reference data B (1) Valid
Reference data A (1) Invalid
Reference area
Reserved area
Block status flag
Block control area

Figure 1.2 Example of Data Programming Map

If it is found that there is not enough free space in the valid block that is currently in use when programming the specified data, that valild block is expanded to program the specified data.

EEPROM emulation expands data flash memory blocks 0 to 3 in that order as valid and standby blocks. When the last block 3 is reached, it loops through the blocks by designating the next sequential block as the initial block 0.

If programming is executed consecutively, the valid blocks are expanded unitl a certain number of blocks (the number of remaining standby blocks is two or less) is reached. If a block expansion occurs when the number of remaining standby blocks is two or less, the valid data in the oldest valid block is copied into the latest valid block and the oldest valid block is erased.

The outline of valid and standby block expansion is shown in the figure below.

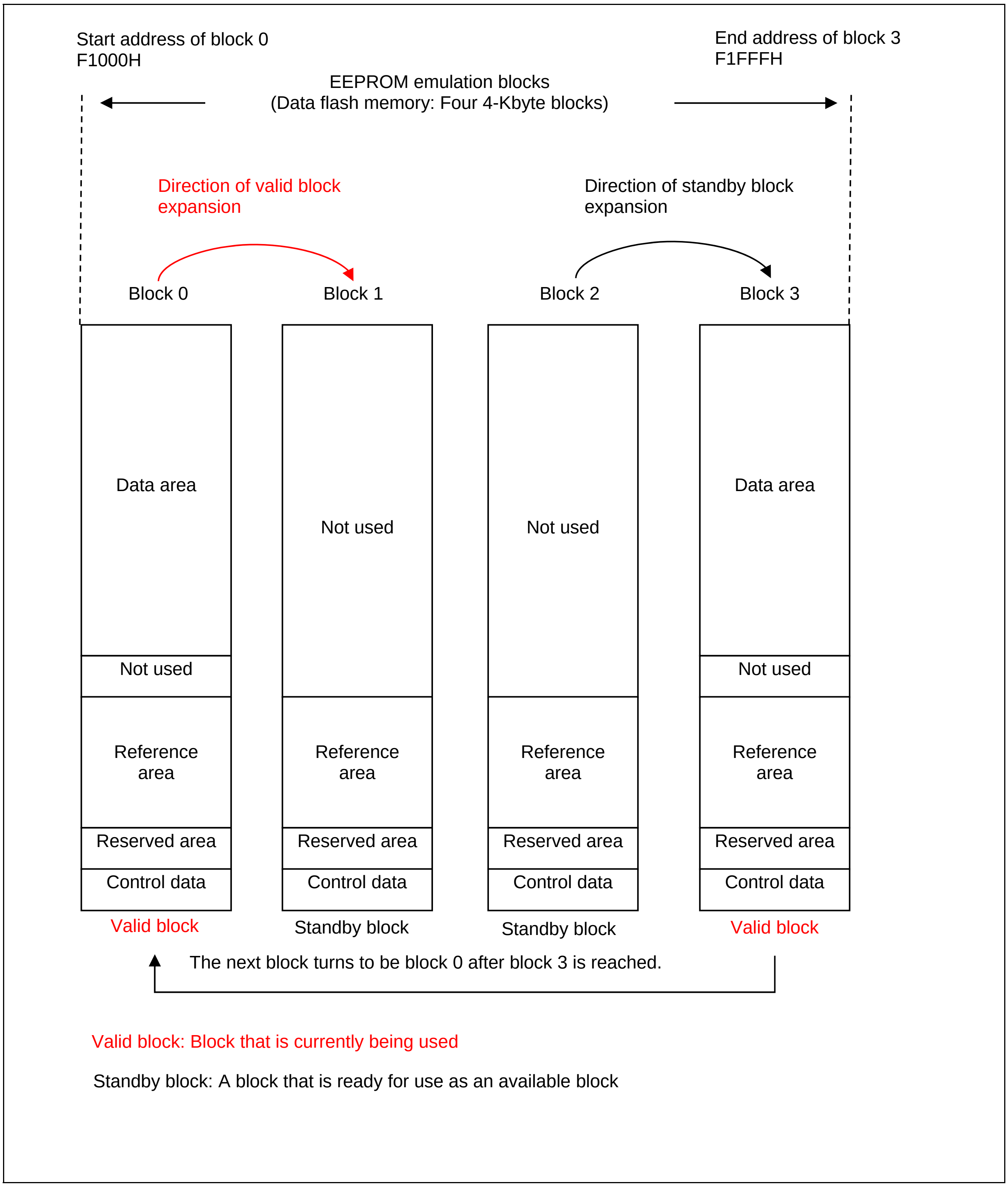


Figure 1.3 Outline of Expansion of Valid and Standby Blocks

The table below summarizes the settings for the data to be programmed into data flash memory and their value ranges.

Setting	Value Range	Remarks
User data length	1 to 255	-
Count of user data to be stored Note 1	1 to 255	Number of data types
Data ID range	1 to 255	-
Number of EEPROM emulation blocks Note 2	4 to 8	-
Recommended size of user data Note 1	988 × Total number of blocks × 1/4 – 988/2 bytes	Including the size of the control reference data to be appended during programming

Notes: 1. The count of user data should be determined so that the total size of user data required during programming into the EEPROM emulation blocks falls within a qurter of the total number of blocks. Accordingly, the value range of the user data count varies according to the size of the user data to store. In addition, the size of the reference data to be appended to each data item as control data need be taken into consideration when determining the total size.

2. This must not exceed the maximum number of blocks of the data flash memory incorporated.

1.2 Initialization of the EEPROM Emulation Blocks

To use data flash memory with EEL, it is necessary to once initialize it as the EEPROM emulation blocks. When a block is in a state in which it should be initialized, in which case there is no valid blocks in the EEPROM emulation block area, an "EEL_ERR_POOL_INCONSISTENT" error is caused when the EEL_CMD_STARTUP command is executed. In such a case, it is necessary to initialize the blocks by executing the EEL_CMD_FORMAT command to make the EEPROM emulation blocks available.

In the case where it is not longer possible to continuously use the EEPROM emuation blocks in the current available state such as when the data flash memory area is corrupted or when it becomes necessary to change the initial settings as the result of adding user data after initialization, or when it becomes necessary to perform EEPROM emulation block initialization at an arbitrary timing, it is possible to initialize the EEPROM emulation blocks by executing the .EEL_CMD_FORMAT command at an arbitrary timing.

1.3 Rearrangement of the EEPROM Emulation Blocks

As programming into the EEPROM emulatiaon blocks proceeds, a standby block is consumed every time a valid block is expanded. If it becomes necessary to expand a valid block when there are less than three remaining standby blocks, the move of the valid data remaining in the old block and the erasure of the old block are carried out before the programming of the specified data is started. If programming occurs at the timing when this erasure processing needs to be performed, there occurs a separate processing time that is required to carry out the move and erasure processing in addition to the processing time required to perform programming.

If this additional processing time is not tolerated, it is possible to avoid the data move and erase operations from being executed at the same time at the timing when a need to perform programming of high urgency arises by maintaining the blocks at the timing when the system has margin.

The block maintenance can be carried out either by rearranging the blocks through the execution of the EEL_CMD_CLEANUP command or by performing maintenance by executing the maintenance mode processing of the EEL_Handler function.

1.4 Initial Settings for EEL

The items listed below need always be set as initial settings for EEL. Because the macros and their names that are used as initial settings are also used as parameter names that are common within EEL, they must not be changed except their numeric values. Their values mut not be changed after the EEPROM emulation blocks are initialized (after the EEL_CMD_FORMAT command is executed). If they are changed, reinitialize the EEPROM emulation blocks.

This application note assumes the initial settings listed in the following tables.

FDL user include file (fdl_descriptor.h)

Constant Name	Value	Remarks
FDL_SYSTEM_FREQUENCY	32000000	Operating frequency
FDL_WIDE_VOLTAGE_MODE	Undefined	Voltage mode of data flash memory Undefined: Full-speed mode Defined: Wide voltage mode
FAL_POOL_SIZE	4	Number of data flash memory blocks
EEL_POOL_SIZE	4	Number of data flash memory blocks

EEL user include file (eel_discriptor.h)

Constant Name	Value	Remarks
EEL_STORAGE_TYPE	‘D’	Flash type (D: Data flash)
EEL_VAR_NO	3	Number of data bytes stored
EEL_REFRESH_BLOCK_THRESHOLD	1	Number of valid blocks that serves as the reference when the maintenance mode is executed.

EEL user program file (eel_discriptor.c)

Variable Name	Value	Remarks
eel_descriptor[EEL_VAR_NO + 1][4]	Describbed outside the table.	ID (data ID) and size of data
eel_refresh_bth_u08	EEL_REFRESH_BLOCK_THRESHOLD	Number of valid blocks that serves as the reference when the maintenance mode is executed
eel_storage_type_u08	EEL_STORAGE_TYPE	Flash type (D: Data flash)
eel_var_number_u08	EEL_VAR_NO	Number of data bytes stored

The values to be set in eel_descriptor [EEL_VAR_NO + 1][4] are listed below.

Data ID	Word Size	Byte Size	RAM Reference Flag
0x01	0x01	0x01	0x01
0x02	0x01	0x01	0x01
0x03	0x01	0x01	0x01
0x00	0x00	0x00	0x00

Note: The RAM reference flag is available for reference configuration. Set 1 in the data.
Set 0x00 at the end of the data as its termination information.

1.5 How to Get the EEPROM Emulation Library

Before compiling the sample program, please download the latest EEPROM Emulation library and copy the library files to the following folder below “r01an3022_flash”.

incrl78 folder : eel.h, eel_type.h, fdl.h, fdl_types.h

librl78 folder : eel.lib, fdl.lib

descriptor folder : eel_descriptor.c, eel_descriptor.h, eel_user_type.h, fdl_descriptor.c, fdl_descriptor.h

The EEPROM Emulation library is available on the Renesas Electronics Website.

Please contact your local Renesas Electronics sales office or distributor for more information.

2. Operation Check Conditions

The sample code described in this application note has been checked under the conditions listed in the table below.

Table 2.1 Operation Check Conditions

Item	Description
Microcontroller used	RL78/G13 (R5F100LEA)
Operating frequency	• High-speed on-chip oscillator (HOCO) clock: 32 MHz • CPU/peripheral hardware clock: 32 MHz
Operating voltage	5.0 V (Operation is possible over a voltage range of 2.9 V to 5.5 V.) LVD operation (V_{LVD}): Reset mode which uses 2.81 V (2.76 V to 2.87 V)
Integrated development environment	CS+ V3.02.00 from Renesas Electronics Corp.
C compiler	CA78K0R V1.72 from Renesas Electronics Corp.
Board to be used	Renesas Starter Kit for RL78/G13 (R0K50100LS000BE)
Flash memory self-programming library (Type, Ver)	EELRL78 Pack01, Ver 1.12 Note

Note: Use and evaluate the latest version.

3. Related Application Notes

The application notes that are related to this application note are listed below for reference.

- RL78/G13 Initialization (R01AN0451E) Application Note
- RL78 Microcontroller EEPROM Emulation Library Pack01 (R01AN0351J) Application Note

4. Description of the Hardware

4.1 Hardware Configuration Example

Figure 4.1 shows an example of the hardware configuration used for this application note.

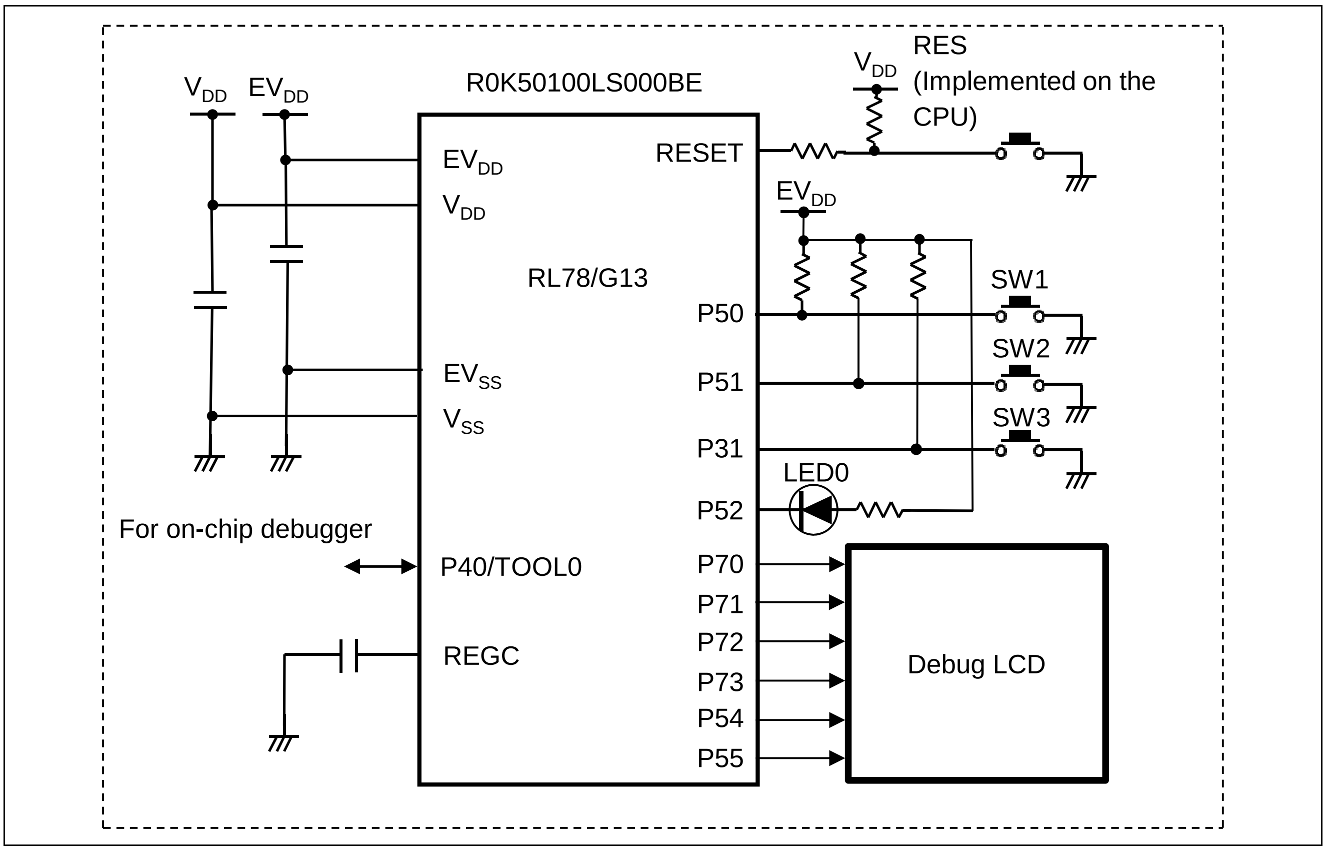


Figure 4.1 Hardware Configuration

Cautions:
1. The purpose of this circuit is only to provide the connection outline and the circuit is simplified accordingly. When designing and implementing an actual circuit, provide proper pin treatment and make sure that the hardware's electrical specifications are met (connect the input-only ports separately to V_{DD} or V_{SS} via a resistor).
2. V_{DD} must be held at not lower than the reset release voltage (V_{LVD}) that is specified as LVD.

4.2 List of Pins to be Used

Table 4.1 lists pins to be used and their functions.

Table 4.1 Pins to be Used and their Functions

Pin Name	I/O	Description
P31/TI03/TO03/INTP4	I/O	Executes writing.
P50/INTP1/SI11/SDA11	I/O	Sets "target write data ID." Erases data flash memory (pressed long together with INTP2).
P51/INTP2/SO11	I/O	Increments "write value." Erases data flash memory (pressed long together with INTP1).
P52	Output	LED (indicating flash memory access status) on/off control
P54	Output	Debug LCD control
P55	Output	Debug LCD control
P70/KR0/SCK21/SCL21	Output	Debug LCD control
P71/KR1/SI21/SDA21	Output	Debug LCD control
P72/KR2/SO21	Output	Debug LCD control
P73/KR3/SO01	Output	Debug LCD control

5. Description of Software

5.1 Operation Outline

This application note explains how to use the EEPROM Emulation Library Pack01.

The sample program covered in this document displays the write value, the target write data ID, and the current value of each data ID on the LCD. The sample program can be manipulated using three switches which are used to set the write value, set the target write data ID, and program into the data flash memory, respectively. Data in the data flash memory is erased by pressing the two switches for setting the write value and target write data ID at the same time for one second. The display on the LCD is updated every time the write value and the target write data ID are changed. The current value of the data ID is updated after power is turned on, data is programmed, or it is erased.

(1) Sets up the TAU0 channel 0.

<Setting conditions>

- Uses the TAU0 channel 0.
- Uses the 500-kHz operation clock.
- Enables only the software start trigger as the start trigger.
- Sets the enable edge to the falling edge.
- Sets the operation mode to interval timer mode.
- Sets the count start and interrupt settings to “Generates no timer interrupt at the beginning of counting.”
- Uses the timer interrupt (INTTM00).
- Sets the interrupt timing to 100 ms.

(2) Sets up the 12-bit interval timer.

<Setting conditions>

- Uses the 15-kHz operation clock.
- Sets the interrupt priority level to 2.
- Sets the interval time to 10 ms.

(3) Sets up the external interrupt input.

- Sets the interrupt priority level to 1.
- Sets the enable edge to the falling edge.

(4) Initializes the LCD.

(5) Starts the INTP.

- Enables edge detection interrupt processing of the INTP1, INTP2, and INTP4 pins.
- Enables interrupts of the INTP1, INTP2, and INTP4 pins.

(6) Initializes the EEL.

- Initializes the FDL.
- Initializes the RAM that the EEL is to use.
- Makes preparations for the EEL.
- Performs maintenance processing.
- If an error is caused during initialization, the sample program displays "ERROR!" on the LCD and suppresses the execution of the subsequent operations.

(7) Reads the contents of the data flash memory.

- Reads data of ID1 to ID3.

(8) Stops EEL and sets it up so that it is ready to switch into HALT mode.

(9) Clears the pressed status of the switch.

(10) Sends the write value, the target write data ID, and the current value of the data ID to the LCD.

(11) If none of the switches are pressed, switches into HALT mode and waits for a press of the switch.

(12) When a switch-triggered external interrupt occurs, exits the HALT mode and takes the following actions to avoid chatters:

- Starts the interval timer for counting on INTP1, INTP2, or INTP4 interrupt.
- Waits until an interval timer interrupt occurs.
- Has the interval timer interrupt handler test the switch status. More specifically, the interrupt handler checks the level of the input at P31, P50, and P51.
- If the level of P31, P50, or P51 is 0, sets the switch-pressed flag determining that a switch has been pressed.
- If the level of P31, P50, and P51 is 1, clears the switch-pressed flag and returns to step (8), determining that none of the switches have been pressed.

(13) Determine which switch has been pressed.

(14) Takes actions according to the pressed status of the switches.

- Increments the write value if only SW1 is pressed.
- Updates the target write data ID if only SW2 is pressed.
- If only SW3 is pressed, turns on LED0 to indicate that flash memory is being accessed and program the write value into the selected data ID location.
  - ➢ After programming the write value, the sample program reads it and compares the read value with the write value. If they match, the sample program turns off LED0.
  - ➢ If the write value and read value do not match, the sample program displays "ERROR" on the LCD and suppresses the execution of the subsequent operations.
- If SW1 and SW2 are pressed at the same time for one second, turns on LED0 to indicate that flash memory is being accessed and initialize the data flash memory.
  - ➢ If the initialization fails, the sample program displays "ERROR" on the LCD and suppresses the execution of the subsequent operations.
  - ➢ After the initialization is completed, the sample program turns off LED0 and reads data from ID locations ID1 to ID3.

(15) Returns to step (9).

Caution: For information about the precautions in using the device, refer to RL78/G13 User’s Manual: Hardware.

5.2 File Configuration

Table 5.1 lists the additional functions for files that are automatically generated in the integrated development environment and other additional files.

Table 5.1 List of Additional Functions and Files

File Name	Outline	Remarks
r_main.c	Main module	Additional functions: R_MAIN_INTCStart R_MAIN_ClearSwitchFlag R_MAIN_GetSwitchStatus R_MAIN_DetectLongPush R_MAIN_IncrementValue R_MAIN_ChangeDataID R_MAIN_SwitchProcess
r_eel.c	EEL execution	R_EEL_Initialize R_EEL_Maintenance R_EEL_ExecuteWrite R_EEL_ClearDataFlash R_EEL_Close

5.3 List of Option Byte Settings

Table 5.2 summarizes the settings of the option bytes.

Table 5.2 Option Byte Settings

Address	Setting	Description
000C0H/010C0H	11101111B	Disables the watchdog timer. (Stops counting after the release from the reset status.)
000C1H/010C1H	01111111B	LVD reset mode 2.81 V (2.76 V to 2.87 V)
000C2H/010C2H	11101000B	HS mode, HOCO: 32 MHz
000C3H/010C3H	10000100B	Enables the on-chip debugger Erases the data in the flash memory when on-chip debug security ID authentication fails.

The option bytes of the RL78/G13 comprise the user option bytes (000C0H to 000C2H) and on-chip debug option byte (000C3H).

The option bytes are automatically referenced and the specified settings are configured at power-on time or the reset is released.

5.4 Link Directive File

The link directive file is used to reserve the RAM area to be used by the flash self-programming library so that it is not available as the standard RAM area.

The outline of the link directive file that this sample program uses is shown below.

```
;************************************************************
; Redefined RAM area
;************************************************************
; --------------------------------------------------------
; Redefined default data segment RAM
; --------------------------------------------------------
MEMORY RAM          : ( 0FF30AH, 000B16H )
; --------------------------------------------------------
; Define new memory entry for saddr area
; --------------------------------------------------------
MEMORY RAM_SADDR : ( 0FFE20H, 0001E0H )
```

Set up a library-dedicated area so that it is not to be used as the standard RAM area.

5.5 List of Constants

Table 5.3 lists the constants for the sample program.

Table 5.3 Constants for the Sample Program

Constant	Setting	Description
SW_ON	0x01	Confirmation that a switch is pressed
SW_OFF	0x00	Switch-pressed status cleared.
ON_SW_1	0x01	Switch 1 pressed.
ON_SW_2	0x02	Switch 2 pressed.
ON_SW_3	0x04	Switch 3 pressed.
MAX_VALUE	0xFF	Maximum write value of data flash memory
MAX_ID	0x03	Maximum write data ID of data flash memory
EEL_NG	0x01	Abnormal EEL termination
EEL_MODE_ENFORCED	0xFF	Enforced mode
EEL_MODE_POLLING	0x00	Polling mode
LCD_SIZE	0x08	Maximum number of characters to be dieplayed on LCD

5.6 List of Variables

Table 5.4 lists the global variables that are used in this sample program.

Table 5.4 Global Variables for the Sample Program

Type	Variable Name	Contents	Function Used
uint8_t	g_sw_push	Confirmation that a switch is pressed	main R_MAIN_ ClearSwitchFlag r_it_interrupt
uint8_t	g_it_flag	Interval timer interrupt flag	main R_MAIN_ ClearSwitchFlag r_it_interrupt
uint8_t	g_write_value	Write value	main R_MAIN_ IncrementValue R_EEL_ExecuteWrite
uint8_t	g_write_id	Target write data ID	main R_MAIN_ ChangeDataID R_EEL_ExecuteWrite
uint8_t	g_read_value [EEL_VER_NO]	Read value	main R_EEL_ExecuteWrite
int8_t	g_upper_string [LCD_SIZE + 1]	String displayed on the LCD (upper column)	main
int8_t	g_downer_string [LCD_SIZE + 1]	String displayed on the LCD (lower column)	r_main R_EEL_ExecuteWrite R_EEL_ClearDataFlash

5.7 List of Functions

Table 5.5 lists the functions that are used in this sample program.

Table 5.5 List of Functions

Function Name	Outline
R_MAIN_INTCStart	Starts INTP.
R_INTC1_Start	Starts INTP1.
r_intc1_interrupt	INTP1 external interrupt
R_INTC2_Start	Starts INTP2.
r_intc2_interrupt	INTP2 external interrupt
R_INTC4_Start	Starts INTP4.
r_intc4_interrupt	INTP4 external interrupt
R_IT_Start	Starts interval timer.
r_it_interrupt	Interval timer interrupt
R_IT_Stop	Stops interval timer.
R_MAIN_ClearSwitchFlag	Clears switch-pressed status.
R_EEL_Initialize	Initializes EEL.
R_EEL_Maintenance	Processes maintenance mode.
R_MAIN_GetSwitchStatus	Gets switch status.
R_MAIN_SwitchProcess	Processes switch-pressed status.
R_MAIN_IncrementValue	Increments write value.
R_MAIN_ChangeDataID	Changes target write data ID.
R_EEL_ExecuteWrite	Executes writing.
R_MAIN_DetectLongPush	Detects long press.
R_MAIN_INTCStop	Stops INTP.
R_INTC1_Stop	Stops INTP1.
R_INTC2_Stop	Stops INTP2.
R_INTC4_Stop	Stops INTP4.
R_TAU0_Channel0_Start	Starts TAU0 channel 0.
R_TAU0_Channel0_Stop	Stops TAU0 channel 0.
R_EEL_ClearDataFlash	Initializes data flash memory.
R_EEL_Close	Stops EEL.

5.8 Function Specifications

This section describes the specifications for the functions that are used in the sample program.

[Function Name] R_ MAIN_INTCStart

Synopsis	Start INTP.
Header	r_cg_macrodriver.h
	r_cg_cgc.h
	r_cg_port.h
	r_cg_intc.h
	r_cg_timer.h
	r_cg_it.h
	fdl.h
	fdl_descriptor.h
	eel.h
	eel_descriptor.h
	eel_user_types.h
	lcd.h
	stdlib.h
	string.h
	r_cg_userdefine.h
Declaration	void R_MAIN_INTCStart(void)
Explanation	This function starts the INTP1, INTP2, and INTP4.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_INTC1_Start

Synopsis	Start INTP1.
Header	r_cg_macrodriver.h
	r_cg_intc.h
	r_cg_userdefine.h
Declaration	void R_INTC1_Start(void)
Explanation	This function starts the INTP1.
Arguments	None
Return value	None
Remarks	None

[Function Name] r_intc1_interrupt

Synopsis	INTP1 external interrupt
Header	r_cg_macrodriver.h
	r_cg_intc.h
	r_cg_it.h
	r_cg_userdefine.h
Declaration	__interrupt void r_intc1_interrupt(void)
Explanation	This function starts the interval timer.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_INTC2_Start

Synopsis	Start INTP2.
Header	r_cg_macrodriver.h r_cg_intc.h r_cg_userdefine.h
Declaration	void R_INTC2_Start(void)
Explanation	This function starts the INTP2.
Arguments	None
Return value	None
Remarks	None

[Function Name] r_intc2_interrupt

Synopsis	INTP2 external interrupt
Header	r_cg_macrodriver.h r_cg_intc.h r_cg_it.h r_cg_userdefine.h
Declaration	__interrupt void r_intc2_interrupt(void)
Explanation	This function starts the interval timer.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_INTC4_Start

Synopsis	Start INTP4.
Header	r_cg_macrodriver.h r_cg_intc.h r_cg_userdefine.h
Declaration	void R_INTC4_Start(void)
Explanation	This function starts the INTP4.
Arguments	None
Return value	None
Remarks	None

[Function Name] r_intc4_interrupt

Synopsis	INTP4 external interrupt
Header	r_cg_macrodriver.h r_cg_intc.h r_cg_it.h r_cg_userdefine.h
Declaration	__interrupt void r_intc4_interrupt(void)
Explanation	This function starts the interval timer.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_IT_Start

Synopsis	Start interval timer.
Header	r_cg_macrodriver.h r_cg_it.h r_cg_userdefine.h
Declaration	void R_IT_Start(void)
Explanation	This function starts the interval timer.
Arguments	None
Return value	None
Remarks	None

[Function Name] r_it_interrupt

Synopsis	Interval timer interrupt
Header	r_cg_macrodriver.h r_cg_it.h r_cg_userdefine.h
Declaration	__interrupt void r_it_interrupt(void)
Explanation	This function stops the interval timer. If any of switches SW1 to SW3 is pressed, the function sets the switch-pressed status flag (g_sw_push) to 1.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_IT_Stop

Synopsis	Stop interval timer.
Header	r_cg_macrodriver.h r_cg_it.h r_cg_userdefine.h
Declaration	void R_IT_Stop(void)
Explanation	This function stops the interval timer.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_MAIN_ClearSwitchFlag

Synopsis	Clear switch-pressed status.
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_timer.h r_cg_it.h fdl.h fdl_descriptor.h eel.h eel_descriptor.h eel_user_types.h lcd.h stdlib.h string.h r_cg_userdefine.h
Declaration	void R_MAIN_ClearSwitchFlag(void)
Explanation	This function clears the switch-pressed status.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_EEL_Initialize

Synopsis	Initializes EEL.	
Header	r_cg_macrodriver.h r_cg_userdefine.h fdl.h fdl_descriptor.h fdl_types.h eel.h eel_descripter.h eel_user_types.h led.h stdlib.h string.h	
Declaration	uint8_t R_EEL_Initialize(void)	
Explanation	This function initializes the EEL and enables transmitting/receiving data to/from the data flash memory.	
Arguments	sw_status	Switch-pressed status
Return value	• Normal termination: EEL_OK • Abnormal termination: EEL_NG	
Remarks	None	

[Function Name] R_EEL_Maintenance

Synopsis	Process maintenance mode.
Header	r_cg_macrodriver.h
	r_cg_userdefine.h
	fdl.h
	fdl_descriptor.h
	fdl_types.h
	eel.h
	eel_descripter.h
	eel_user_types.h
	led.h
	stdlib.h
	string.h
Declaration	void R_EEL_Maintenance(void)
Explanation	This function rearranges the blocks.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_MAIN_GetSwitchStatus

Synopsis	Get switch status.
Header	r_cg_macrodriver.h
	r_cg_cgc.h
	r_cg_port.h
	r_cg_intc.h
	r_cg_timer.h
	r_cg_it.h
	fdl.h
	fdl_descriptor.h
	eel.h
	eel_descriptor.h
	eel_user_types.h
	lcd.h
	stdlib.h
	string.h
	r_cg_userdefine.h
Declaration	uint8_t R_MAIN_GetSwitchStatus(void)
Explanation	This function gets the status of SW1, SW2, and SW3.
Arguments	None
Return value	Switch-pressed status: sw_status (Initial value = 0)
	• No SW is pressed: sw_status
	• SW1 is pressed: sw_status + ON_SW_1
	• SW2 is pressed: sw_status + ON_SW_2
	• SW3 is pressed: sw_status + ON_SW_3
Remarks	None

[Function Name] R_MAIN_SwitchProcess

Synopsis	Process switch-pressed status.	
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_timer.h r_cg_it.h fdl.h fdl_descriptor.h eel.h eel_descriptor.h eel_user_types.h lcd.h stdlib.h string.h r_cg_userdefine.h	
Declaration	uint8_t R_MAIN_SwitchProcess(uint8_t sw_status)	
Explanation	This function causes a branch according to the switch-pressed status.	
Arguments	sw_status	Switch-pressed status of SW1, SW2, and SW3
Return value	• Normal termination: EEL_OK • Abnormal termination: EEL_NG	
Remarks	None	

[Function Name] R_MAIN_IncrementValue

Synopsis	Increment write value.
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_timer.h r_cg_it.h fdl.h fdl_descriptor.h eel.h eel_descriptor.h eel_user_types.h lcd.h stdlib.h string.h r_cg_userdefine.h
Declaration	void R_MAIN_IncrementValue(void)
Explanation	This function increments the value to be written to the data flash memory.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_MAIN_ChangeDataID

Synopsis	Change target write data ID.
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_timer.h r_cg_it.h fdl.h fdl_descriptor.h eel.h eel_descriptor.h eel_user_types.h lcd.h stdlib.h string.h r_cg_userdefine.h
Declaration	void R_MAIN_ChangeDataID(void)
Explanation	This function changes the data ID to which the write value is to be written.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_EEL_ExecuteWrite

Synopsis	Execute writing.
Header	r_cg_macrodriver.h r_cg_userdefine.h fdl.h fdl_descriptor.h fdl_types.h eel.h eel_descripter.h eel_user_types.h led.h stdlib.h string.h
Declaration	uint8_t R_EEL_ExecuteWrite(void)
Explanation	This function writes the write value to the target write data ID in the data flash memory.
Arguments	None
Return value	• Normal termination: EEL_OK • Abnormal termination: EEL_NG
Remarks	None

[Function Name] R_MAIN_DetectLongPush

Synopsis	Detect long press.
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_timer.h r_cg_it.h fdl.h fdl_descriptor.h eel.h eel_descriptor.h eel_user_types.h lcd.h stdlib.h string.h r_cg_userdefine.h
Declaration	uint8_t R_MAIN_DetectLongPush(void)
Explanation	This function checks whether a switch is pressed long.
Arguments	None
Return value	• Long press is detected: SW_ON • Long press is not detected: SW_OFF
Remarks	None

[Function Name] R_ MAIN_INTCStop

Synopsis	Stop INTP.
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_timer.h r_cg_it.h fdl.h fdl_descriptor.h eel.h eel_descriptor.h eel_user_types.h lcd.h stdlib.h string.h r_cg_userdefine.h
Declaration	void R_MAIN_INTCStop(void)
Explanation	This function stops the INTP1, INTP2, and INTP4.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_INTC1_Stop

Synopsis	Stop INTP1.
Header	r_cg_macrodriver.h r_cg_intc.h r_cg_userdefine.h
Declaration	void R_INTC1_Stop(void)
Explanation	This function stops the INTP1.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_INTC2_Stop

Synopsis	Stop INTP2.
Header	r_cg_macrodriver.h r_cg_intc.h r_cg_userdefine.h
Declaration	void R_INTC2_Stop(void)
Explanation	This function stops the INTP2.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_INTC4_Stop

Synopsis	Stop INTP4.
Header	r_cg_macrodriver.h r_cg_intc.h r_cg_userdefine.h
Declaration	void R_INTC4_Stop(void)
Explanation	This function stops the INTP4.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_TAU0_Channel0_Start

Synopsis	Start TAU0 channel 0.
Header	r_cg_macrodriver.h r_cg_timer.h r_cg_userdefine.h
Declaration	void R_TAU0_Channel0_Start(void)
Explanation	This function starts channel 0 of the timer array unit 0.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_TAU0_Channel0_Stop

Synopsis	Stop TAU0 channel 0.
Header	r_cg_macrodriver.h r_cg_timer.h r_cg_userdefine.h
Declaration	void R_TAU0_Channel0_Stop(void)
Explanation	This function stops channel 0 of the timer array unit 0.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_EEL_ClearDataFlash

Synopsis	Initialize data flash memory.
Header	r_cg_macrodriver.h r_cg_userdefine.h fdl.h fdl_descriptor.h eel.h eel_descripter.h eel_user_types.h
Declaration	uint8_t R_EEL_ClearDataFlash(void)
Explanation	This function initializes the data flash memory.
Arguments	None
Return value	• Normal termination: EEL_OK • Abnormal termination: EEL_NG
Remarks	None

[Function Name] R_EEL_Close

Synopsis	Stop EEL.
Header	r_cg_macrodriver.h r_cg_userdefine.h fdl.h fdl_descriptor.h eel.h eel_descripter.h eel_user_types.h
Declaration	uint8_t R_EEL_Close(void)
Explanation	This function stops EEPROM emulation after placeing it into the stopped state.
Arguments	None
Return value	• Normal termination: EEL_OK • Abnormal termination: EEL_NG
Remarks	None

5.9 Flowcharts

Figure 5.1 shows the overall flow of the sample program described in this application note.

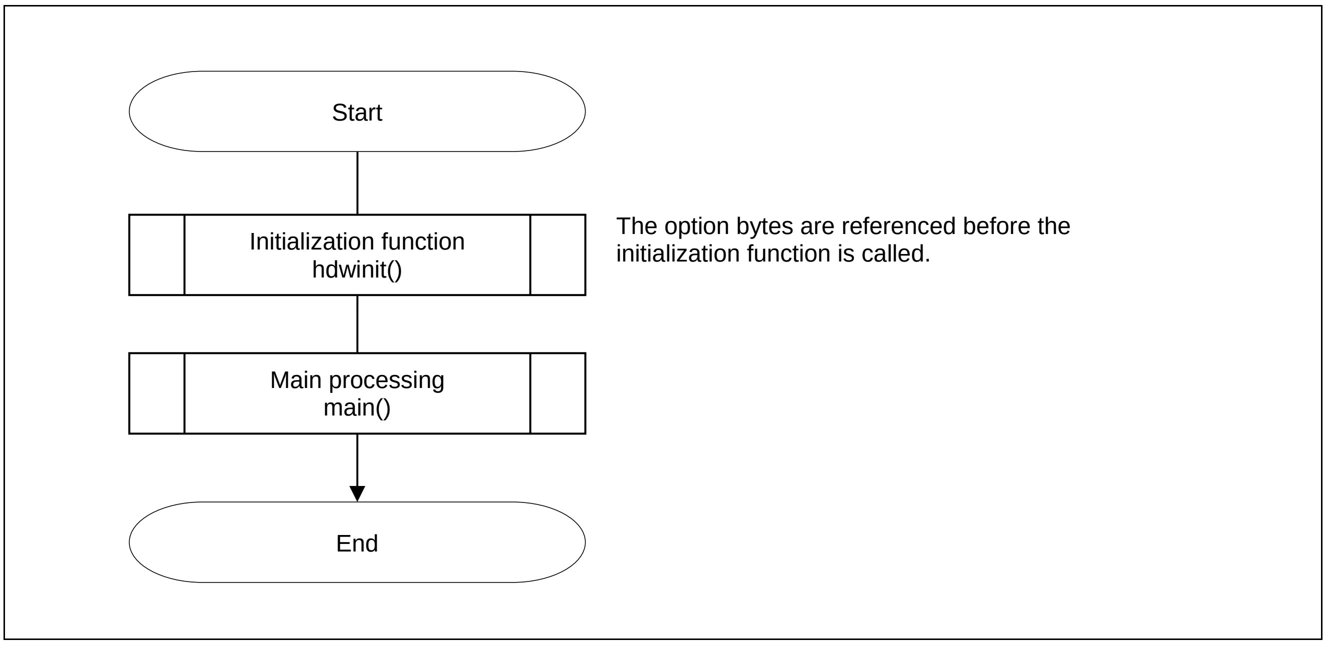

Start
Initialization function
hdwinit()
The option bytes are referenced before the initialization function is called.
Main processing
main()
End

Figure 5.1 Overall Flow

5.9.1 Initialization Function

Figure 5.2 shows the flowchart for the initialization function.

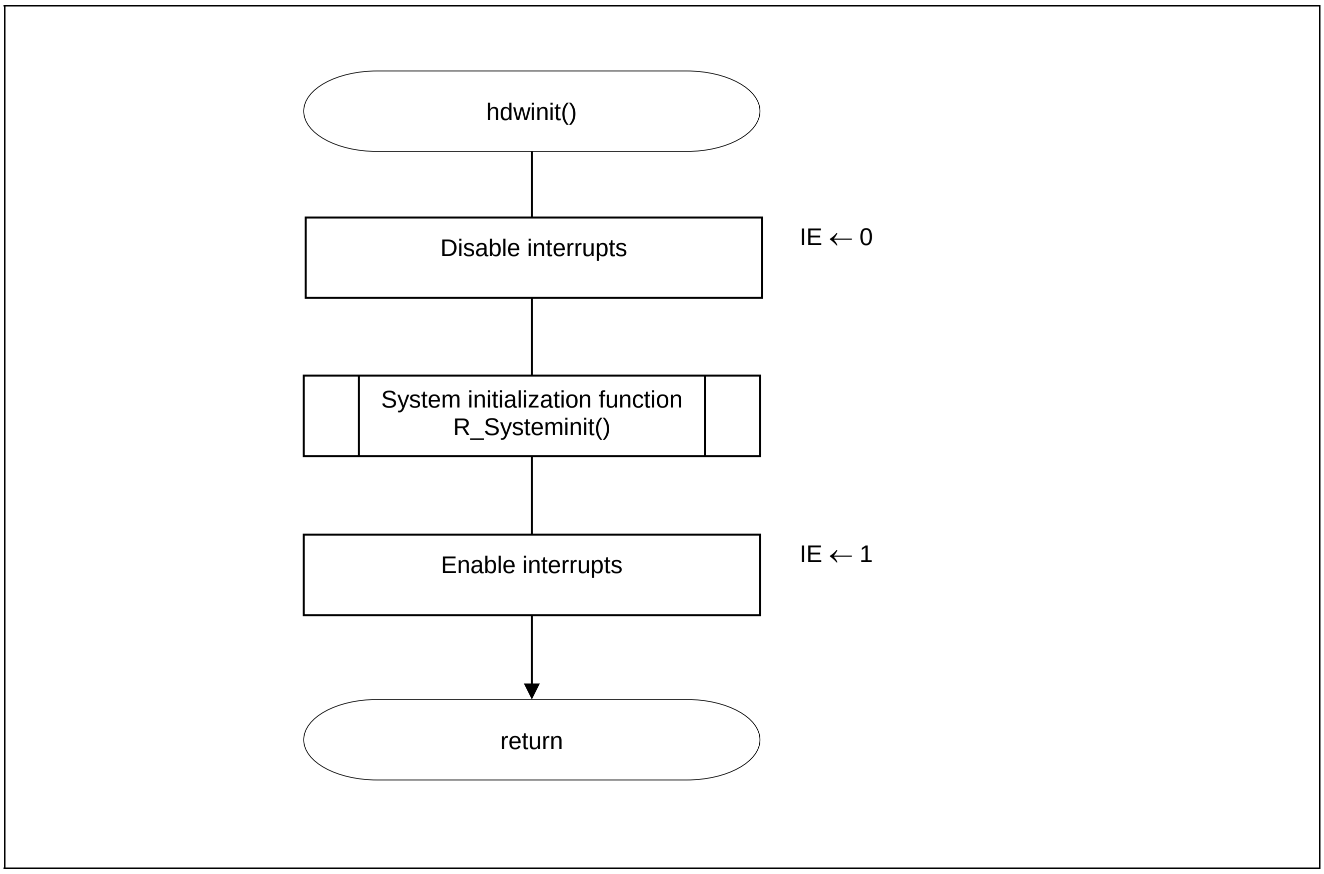


Figure 5.2 Initialization Function

5.9.2 System Initialization Function

Figure 5.3 shows the flowchart for the system initialization function.

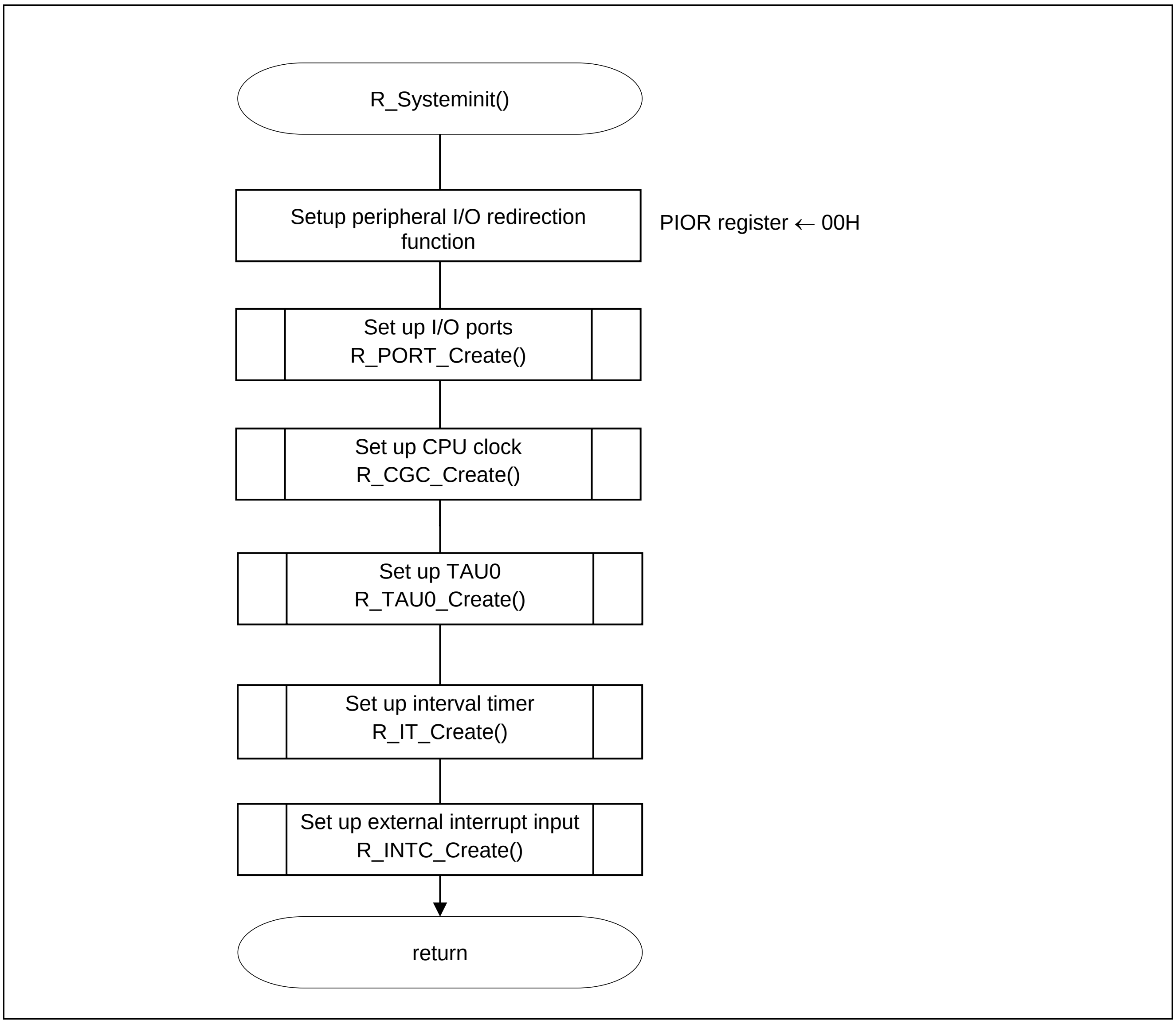


Figure 5.3 System Initialization Function

5.9.3 I/O Port Setup

Figure 5.4 the flowchart for I/O port setup.

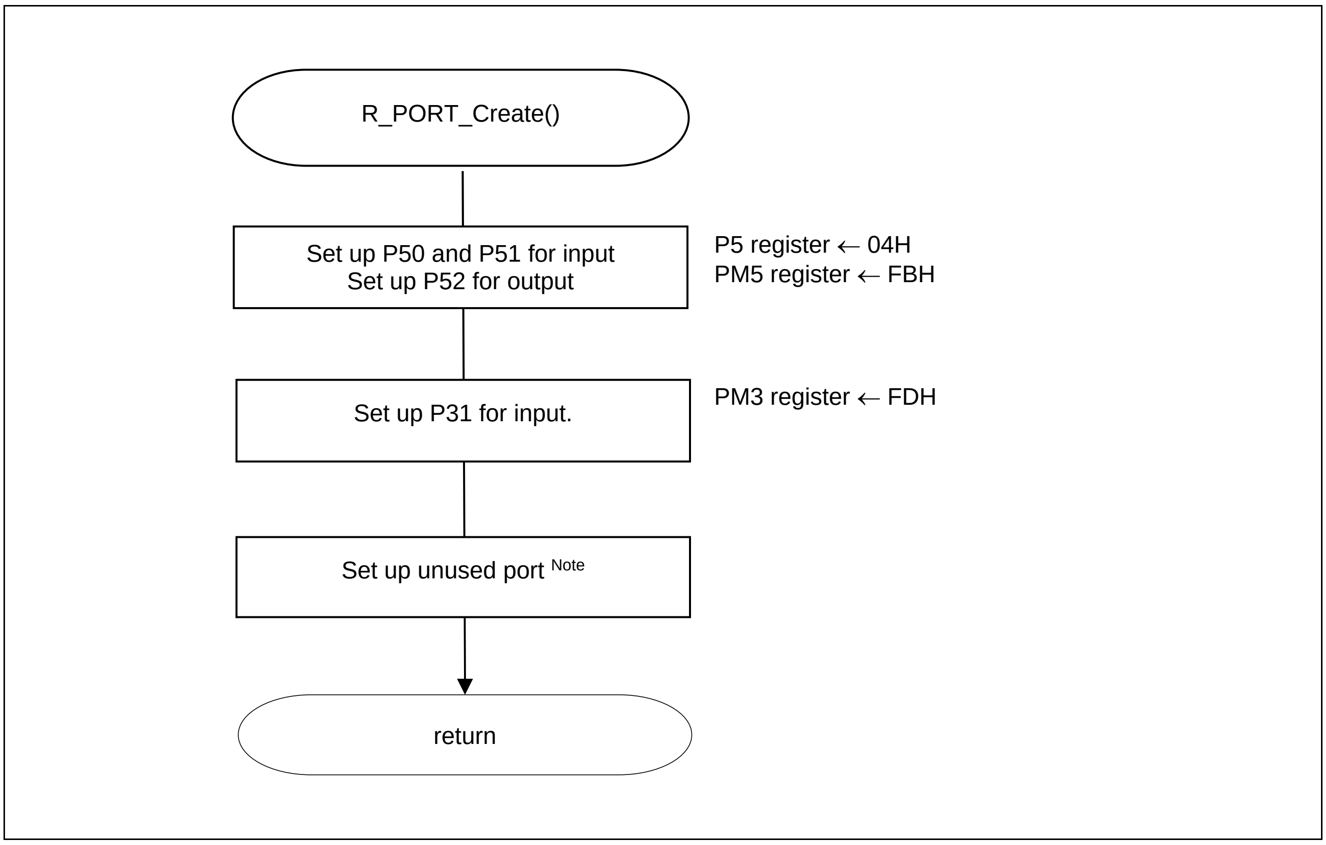


Figure 5.4 I/O Port Setup

Note: Refer to the section entitled "Flowcharts" in RL78/G13 Initialization (R01AN0451E) Application Note for the configuration of the unused ports.

Caution: Provide proper treatment for unused pins so that their electrical specifications are observed. Connect each of any unused input-only ports to V_{DD} or V_{SS} via a separate resistor.

5.9.4 CPU Clock Setup

Figure 5.5 shows the flowchart for CPU clock setup.

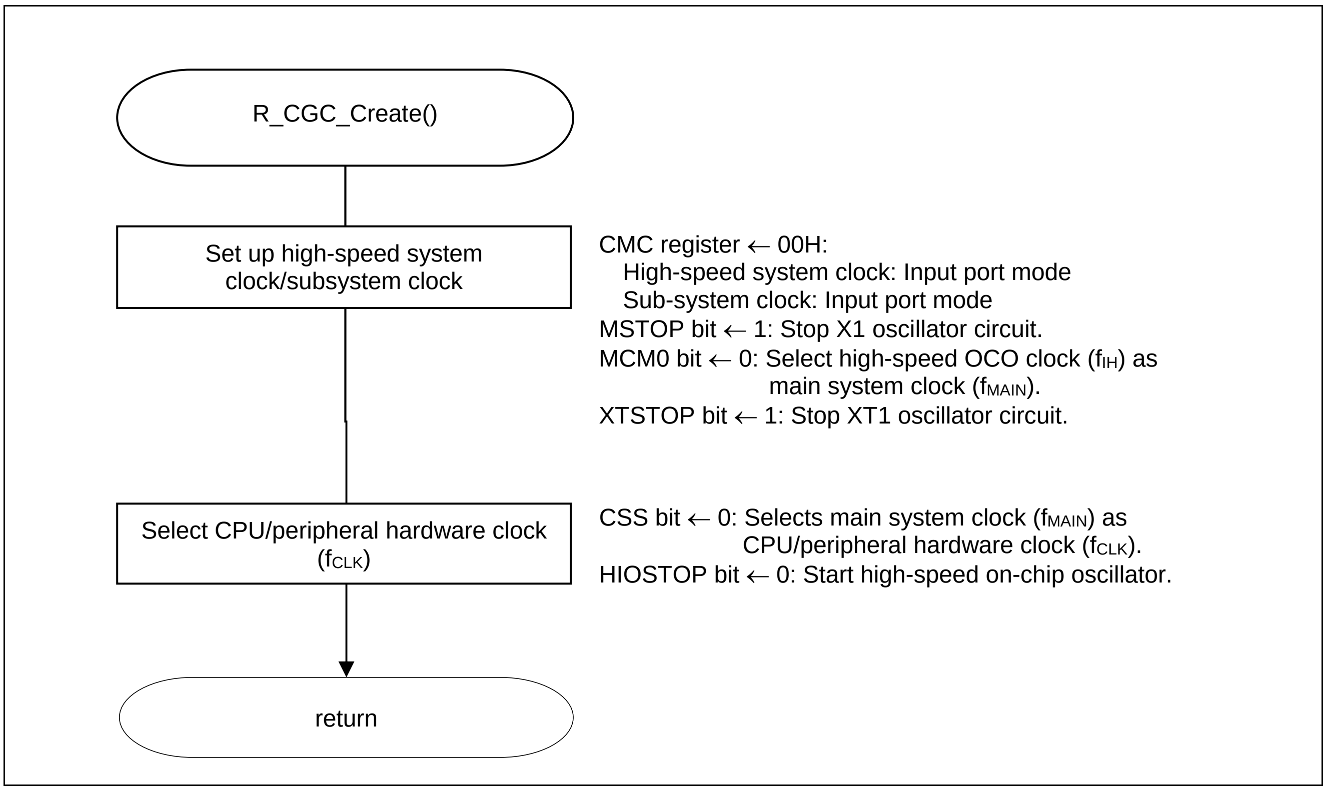

R_CGC_Create()
Set up high-speed system clock/subsystem clock
CMC register ← 00H:
High-speed system clock: Input port mode
Sub-system clock: Input port mode
MSTOP bit ← 1: Stop X1 oscillator circuit.
MCM0 bit ← 0: Select high-speed OCO clock (f_{IH}) as main system clock (f_{MAIN}).
XTSTOP bit ← 1: Stop XT1 oscillator circuit.
Select CPU/peripheral hardware clock (f_{CLK})
CSS bit ← 0: Selects main system clock (f_{MAIN}) as CPU/peripheral hardware clock (f_{CLK}).
HIOSTOP bit ← 0: Start high-speed on-chip oscillator.
return

Figure 5.5 CPU Clock Setup

Caution: For details on the procedure for setting up the CPU clock (R_CGC_Create ()), refer to the section entitled "Flowcharts" in RL78/G13 Initialization (R01AN0451E) Application Note.

5.9.5 TAU0 Setup

Figure 5.6 shows the flowchart for TAU0 setup.

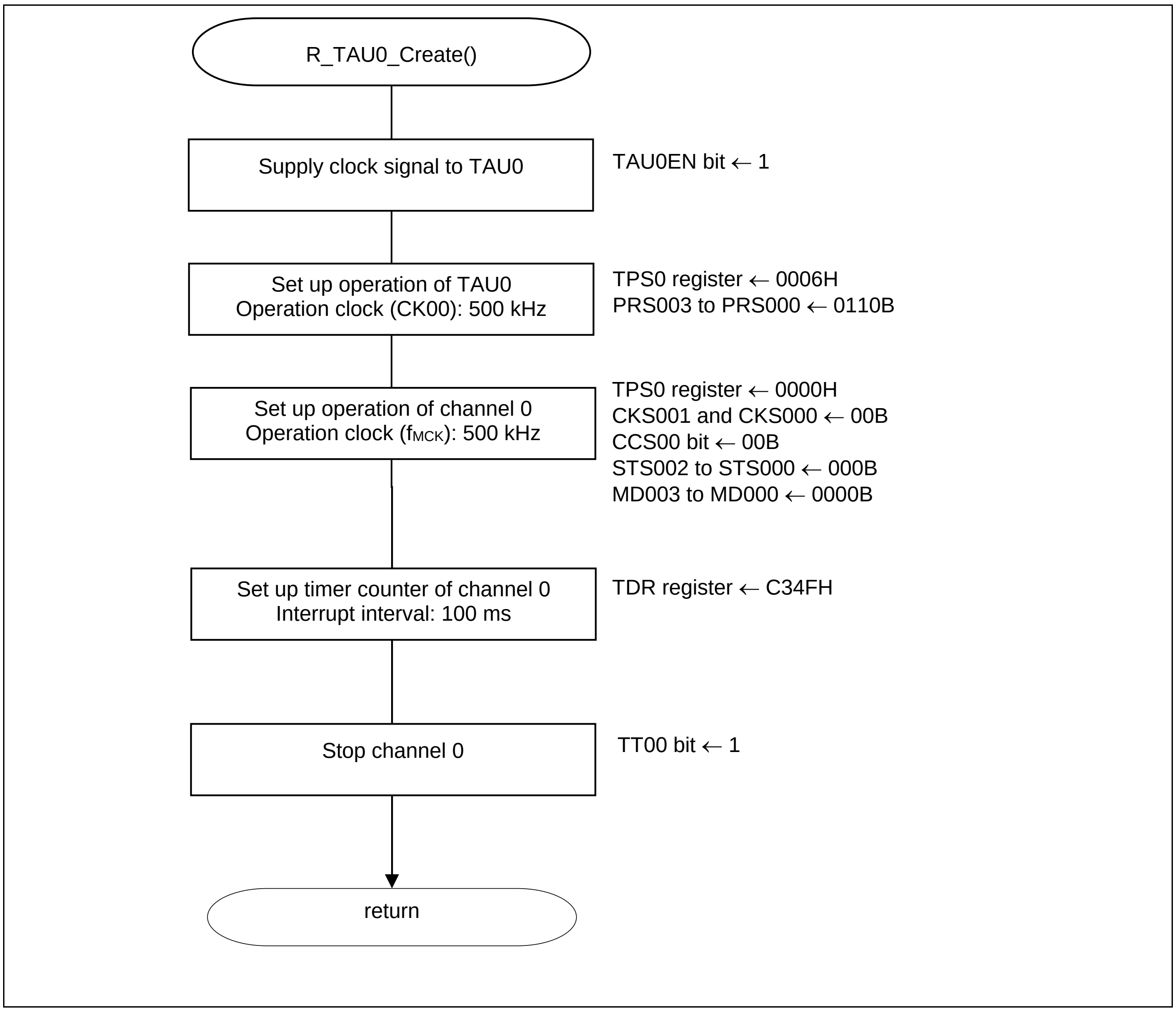


Figure 5.6 TAU0 Setup

5.9.6 Interval Timer Setup

Figure 5.7 shows the flowchart for interval timer setup.

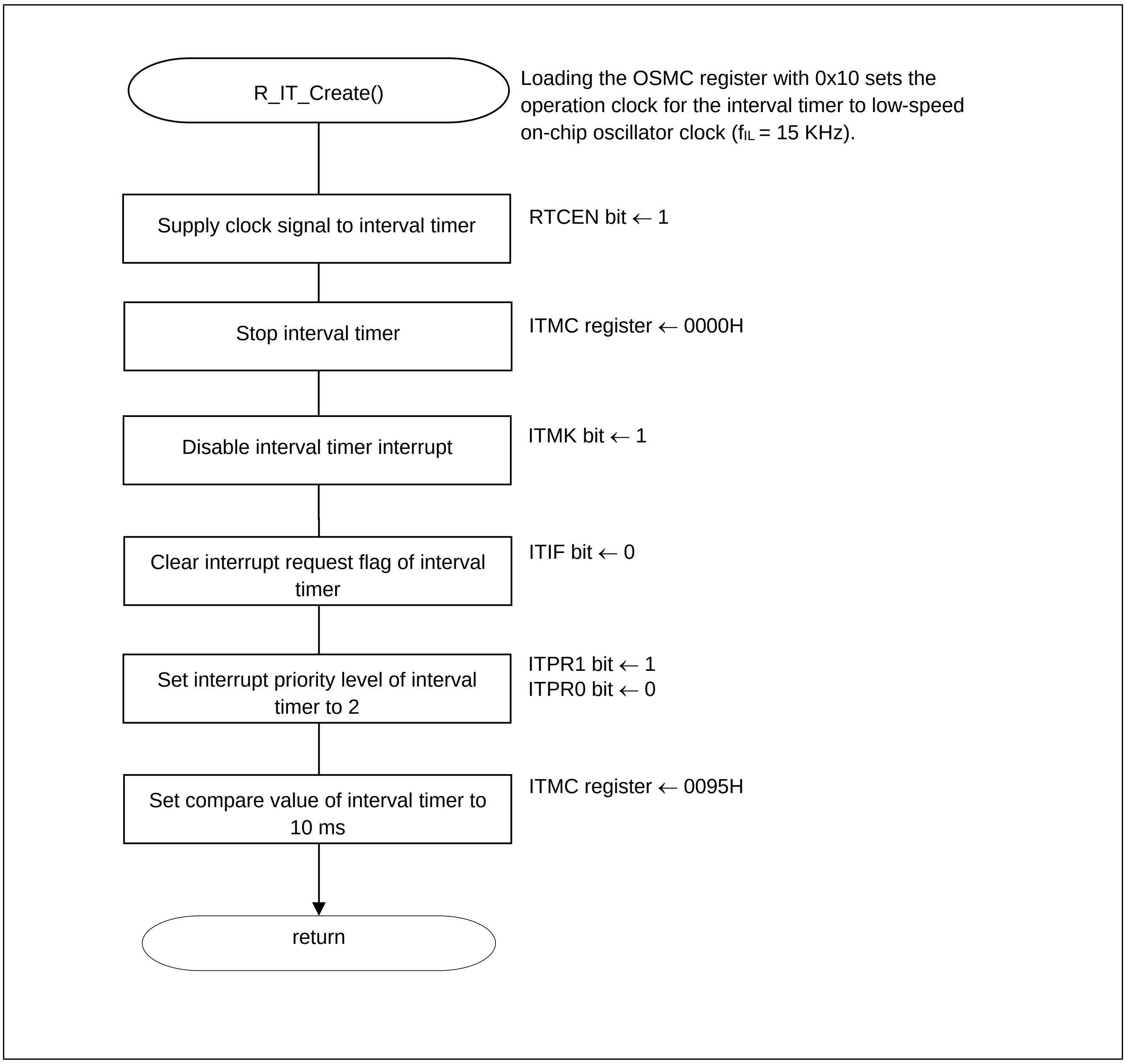

R_IT_Create()
Loading the OSMC register with 0x10 sets the operation clock for the interval timer to low-speed on-chip oscillator clock (fIL = 15 KHz).
Supply clock signal to interval timer
RTCEN bit ← 1
Stop interval timer
ITMC register ← 0000H
Disable interval timer interrupt
ITMK bit ← 1
Clear interrupt request flag of interval timer
ITIF bit ← 0
Set interrupt priority level of interval timer to 2
ITPR1 bit ← 1
ITPR0 bit ← 0
Set compare value of interval timer to 10 ms
ITMC register ← 0095H
return

Figure 5.7 Interval Timer Setup

5.9.7 External Interrupt Input Setup

Figure 5.8 shows the flowchart for external interrupt input setup.

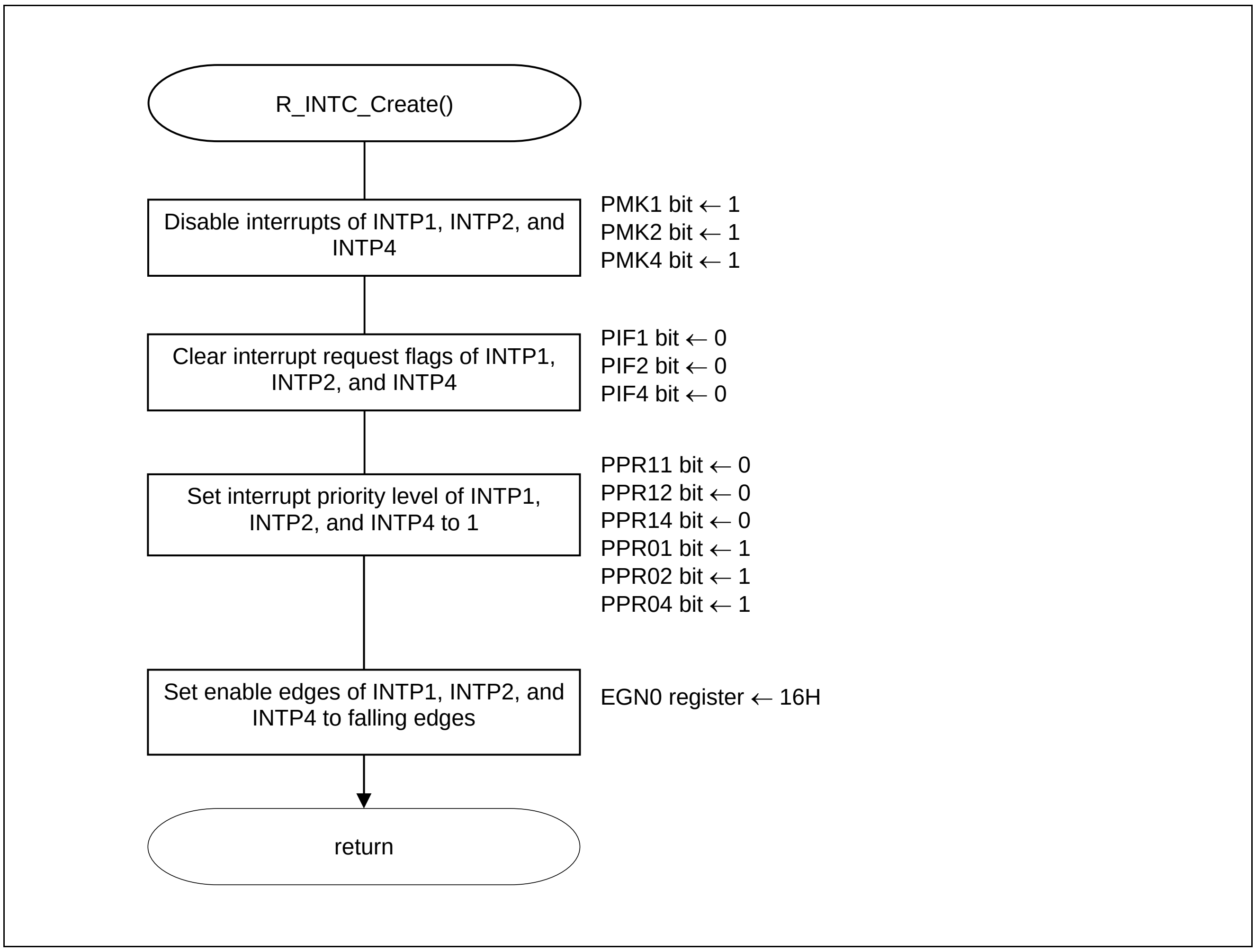

R_INTC_Create()
Disable interrupts of INTP1, INTP2, and INTP4
PMK1 bit ← 1
PMK2 bit ← 1
PMK4 bit ← 1
Clear interrupt request flags of INTP1, INTP2, and INTP4
PIF1 bit ← 0
PIF2 bit ← 0
PIF4 bit ← 0
Set interrupt priority level of INTP1, INTP2, and INTP4 to 1
PPR11 bit ← 0
PPR12 bit ← 0
PPR14 bit ← 0
PPR01 bit ← 1
PPR02 bit ← 1
PPR04 bit ← 1
Set enable edges of INTP1, INTP2, and INTP4 to falling edges
EGN0 register ← 16H
return

Figure 5.8 External Interrupt Input Setup

5.9.8 Main Processing

Figures 5.9 to 5.12 show the flowcharts for main processing.

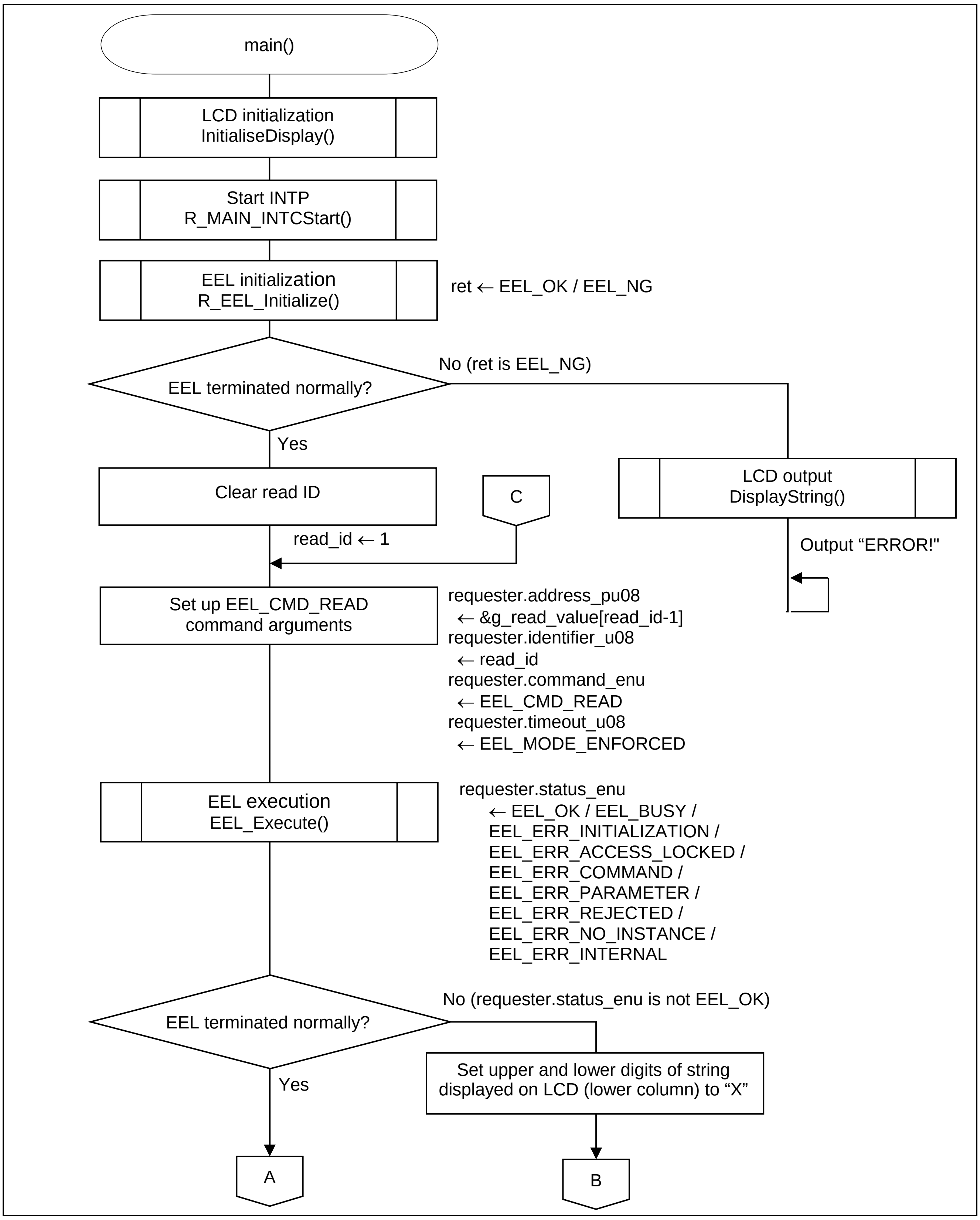


Figure 5.9 Main Processing (1/4)

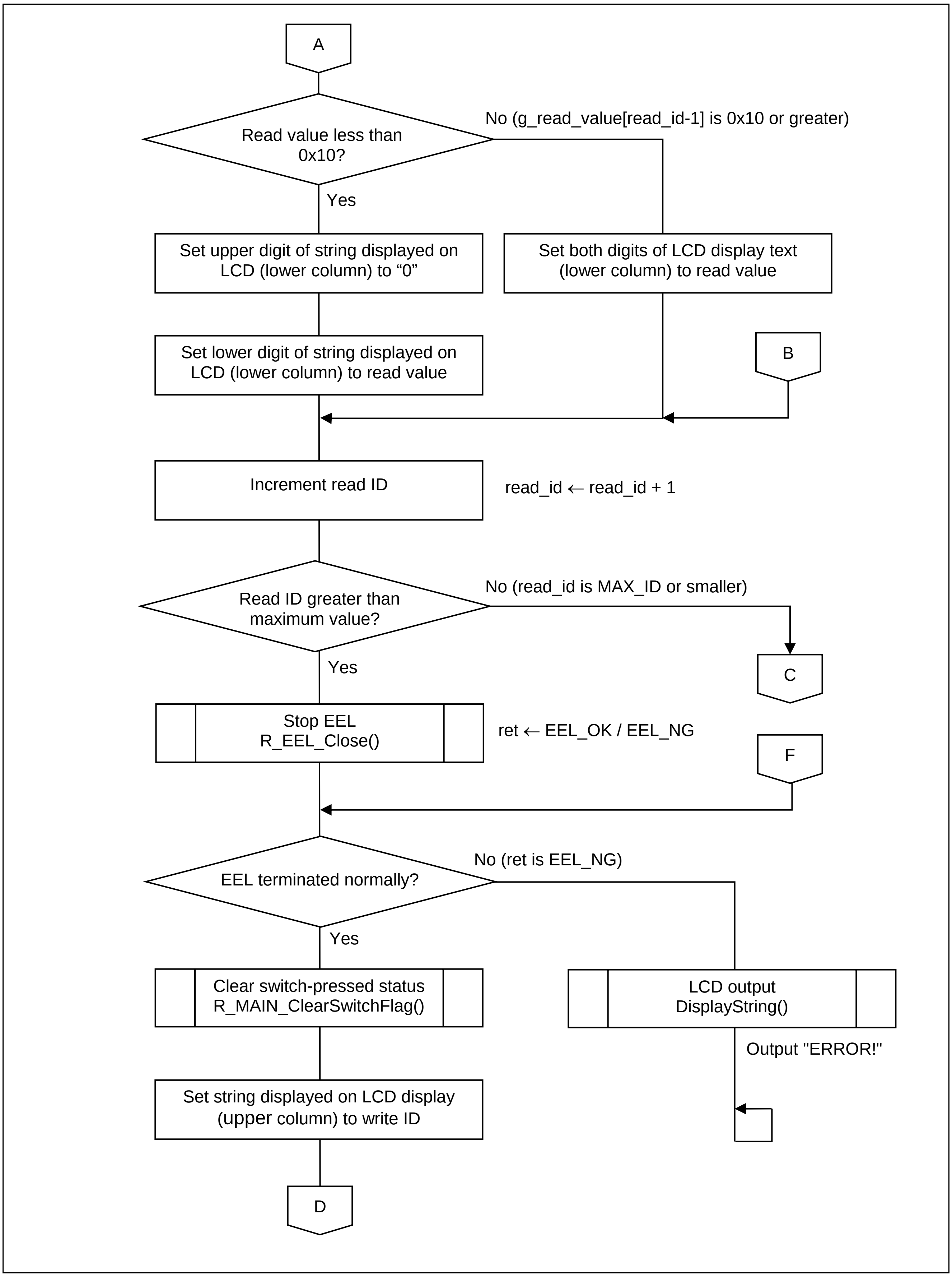

A
Read value less than 0x10?
No (g_read_value[read_id-1] is 0x10 or greater)
Yes
Set upper digit of string displayed on LCD (lower column) to "0"
Set both digits of LCD display text (lower column) to read value
Set lower digit of string displayed on LCD (lower column) to read value
B
Increment read ID
read_id ← read_id + 1
Read ID greater than maximum value?
No (read_id is MAX_ID or smaller)
Yes
C
Stop EEL R_EEL_Close()
ret ← EEL_OK / EEL_NG
F
EEL terminated normally?
No (ret is EEL_NG)
Yes
Clear switch-pressed status R_MAIN_ClearSwitchFlag()
LCD output DisplayString()
Output "ERROR!"
Set string displayed on LCD display (upper column) to write ID
D

Figure 5.10 Main Processing (2/4)

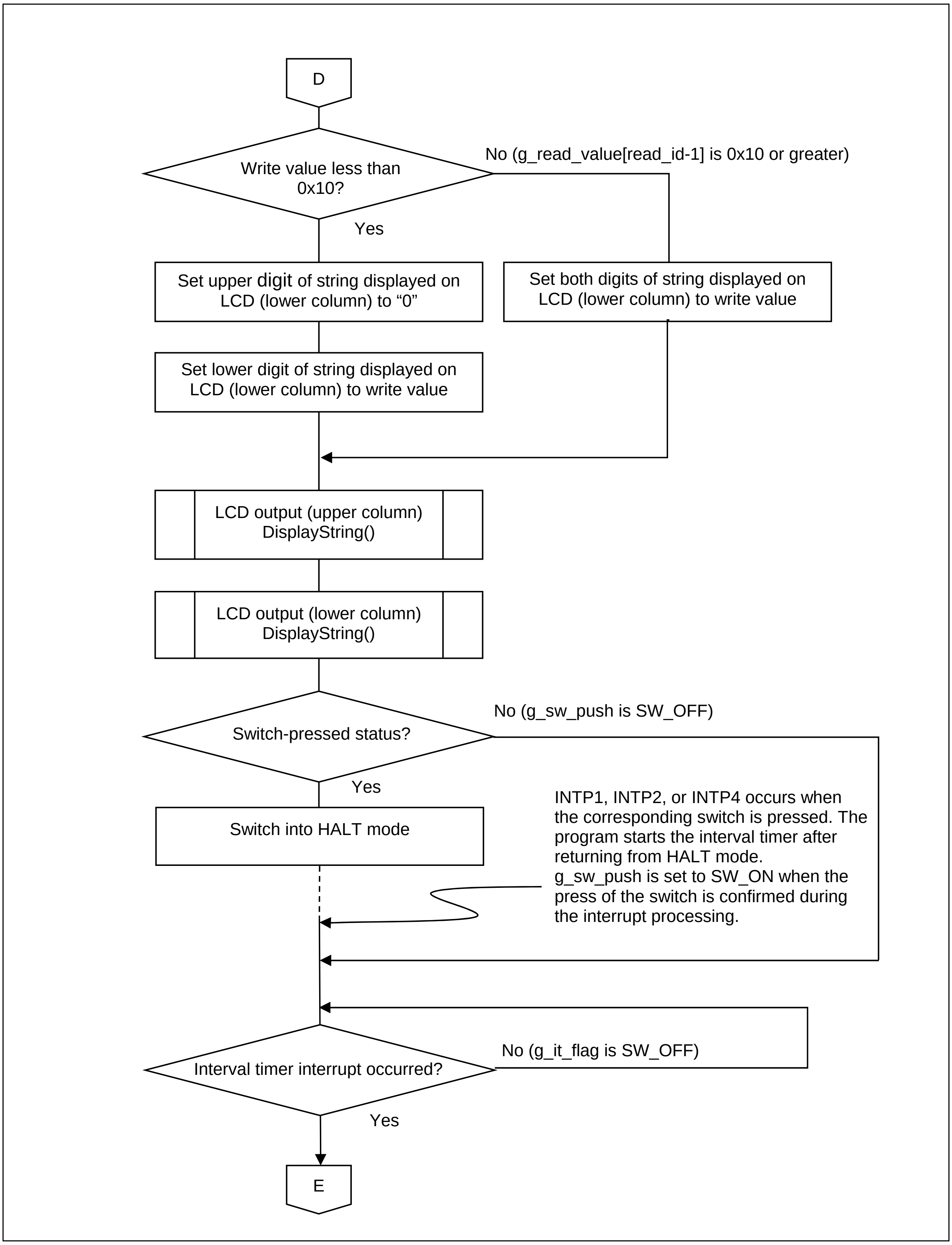

D
Write value less than 0x10?
No (g_read_value[read_id-1] is 0x10 or greater)
Yes
Set upper digit of string displayed on LCD (lower column) to "0"
Set both digits of string displayed on LCD (lower column) to write value
Set lower digit of string displayed on LCD (lower column) to write value
LCD output (upper column) DisplayString()
LCD output (lower column) DisplayString()
Switch-pressed status?
No (g_sw_push is SW_OFF)
Yes
Switch into HALT mode
INTP1, INTP2, or INTP4 occurs when the corresponding switch is pressed. The program starts the interval timer after returning from HALT mode. g_sw_push is set to SW_ON when the press of the switch is confirmed during the interrupt processing.
Interval timer interrupt occurred?
No (g_it_flag is SW_OFF)
Yes
E

Figure 5.11 Main Processing (3/4)

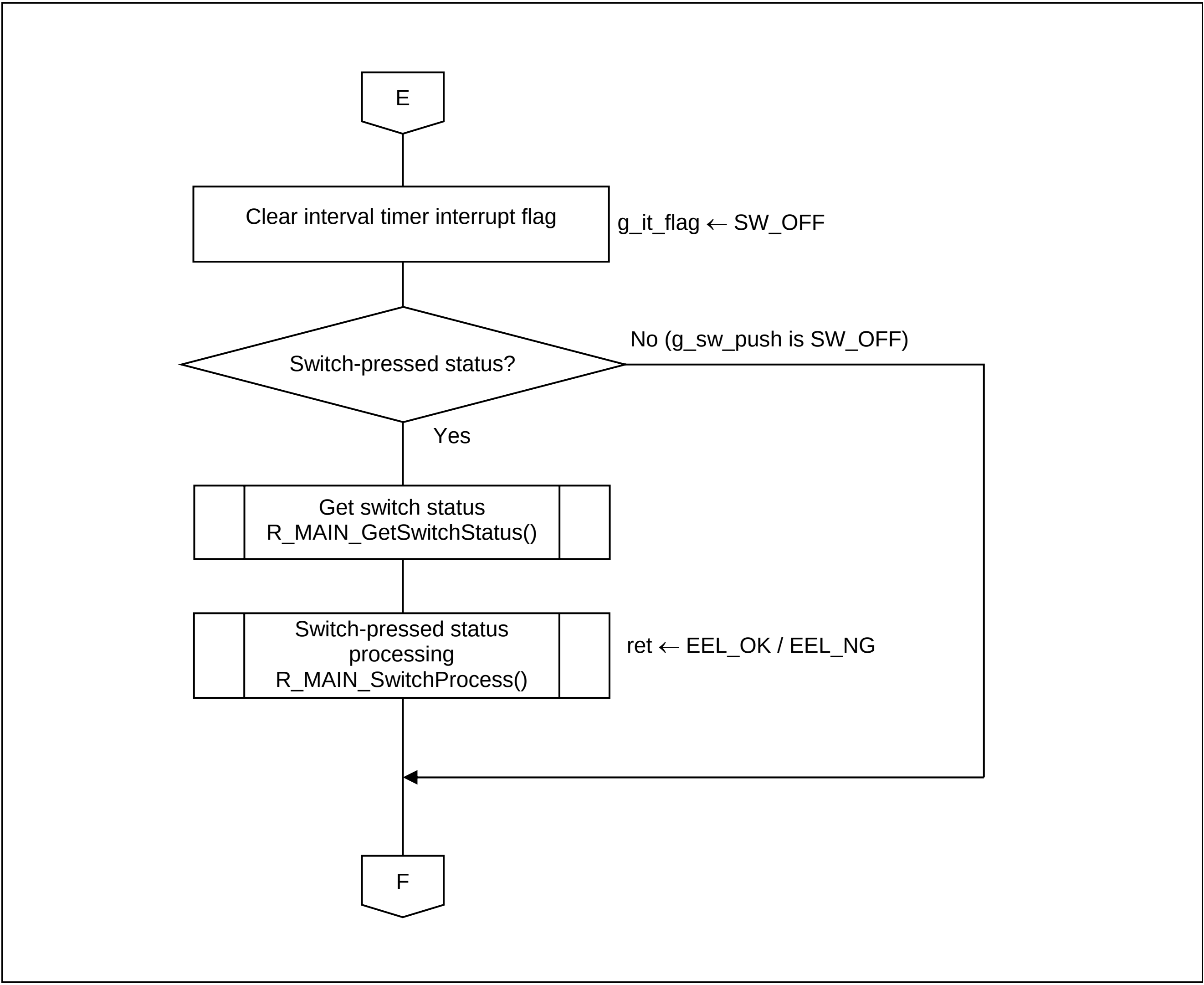


Figure 5.12 Main Processing (4/4)

5.9.9 Starting the INTP

Figure 5.13 shows the flowchart for starting the INTP.

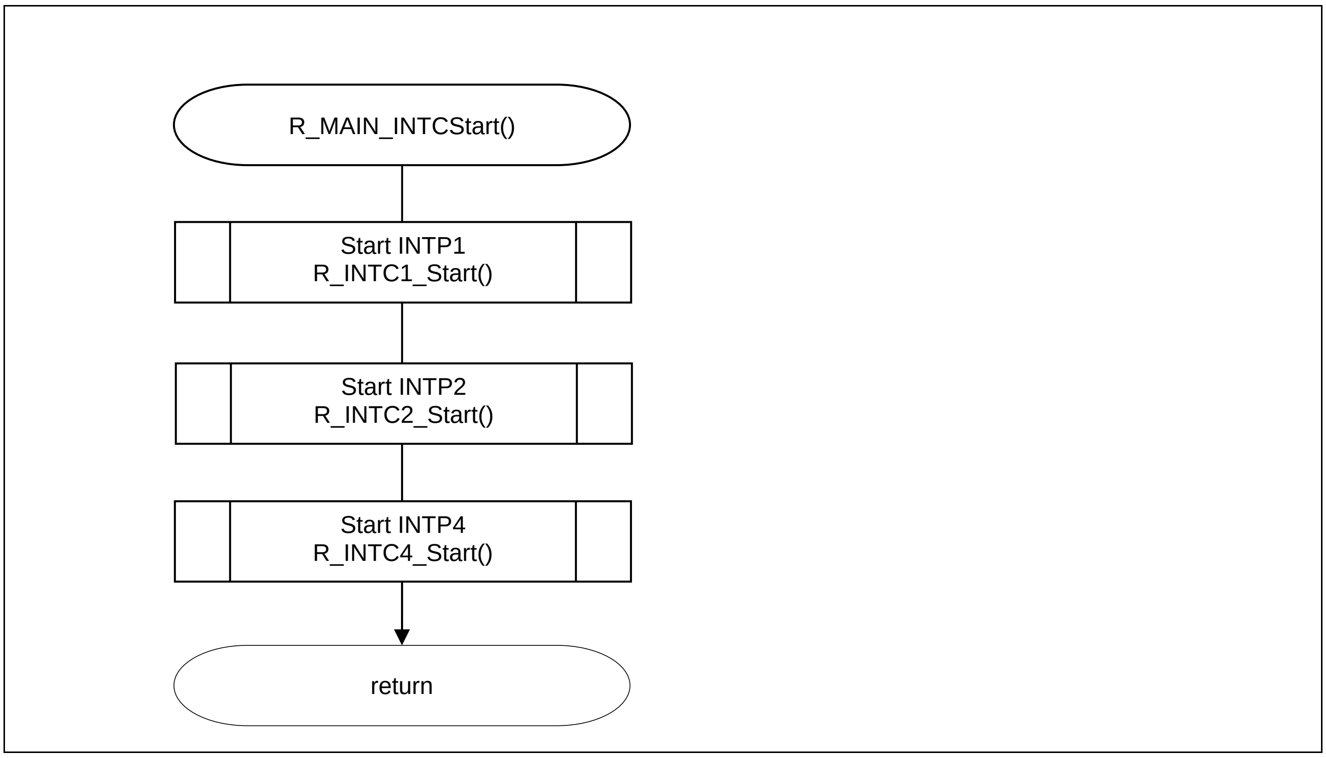


Figure 5.13 Starting the INTP

5.9.10 Starting the INTP1

Figure 5.14 shows the flowchart for starting the INTP1.

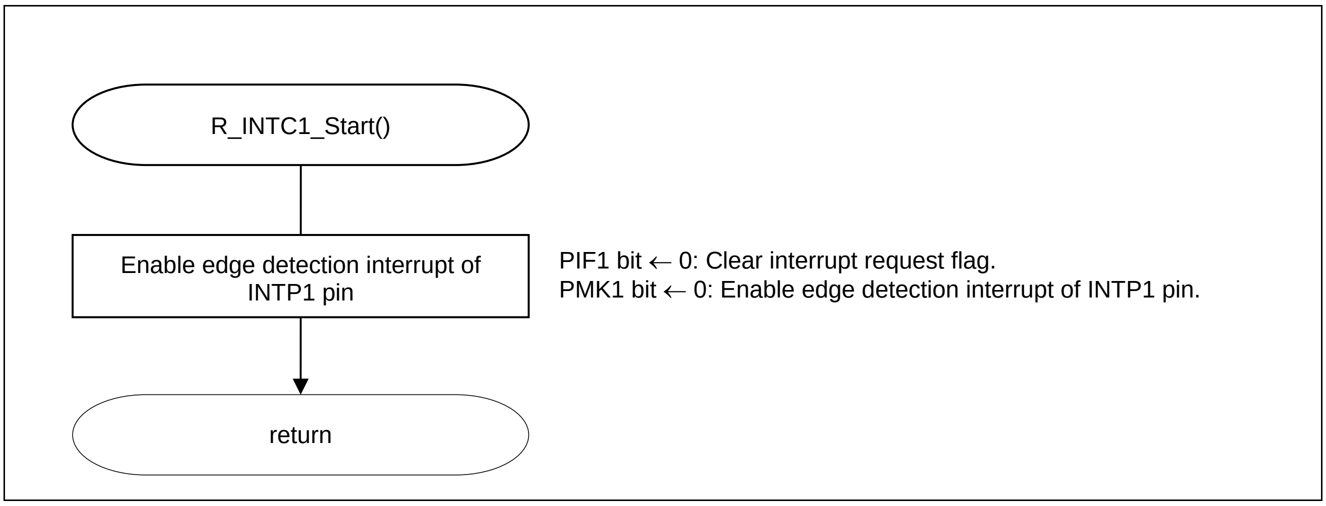

R_INTC1_Start()
Enable edge detection interrupt of INTP1 pin
PIF1 bit ← 0: Clear interrupt request flag.
PMK1 bit ← 0: Enable edge detection interrupt of INTP1 pin.
return

Figure 5.14 Starting the INTP1

5.9.11 INTP1 External Interrupt

Figure 5.15 shows the flowchart for INTP1 external interrupt.

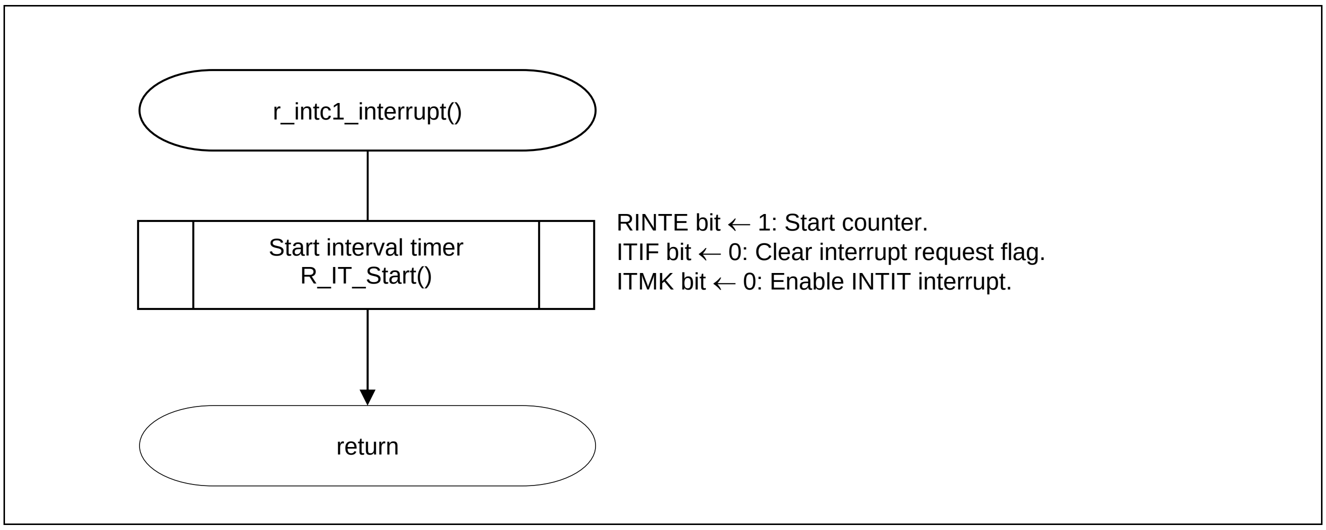

r_intc1_interrupt()
Start interval timer R_IT_Start()
RINTE bit ← 1: Start counter.
ITIF bit ← 0: Clear interrupt request flag.
ITMK bit ← 0: Enable INTIT interrupt.
return

Figure 5.15 INTP1 External Interrupt

5.9.12 Starting the INTP2

Figure 5.16 shows the flowchart for starting the INTP2.

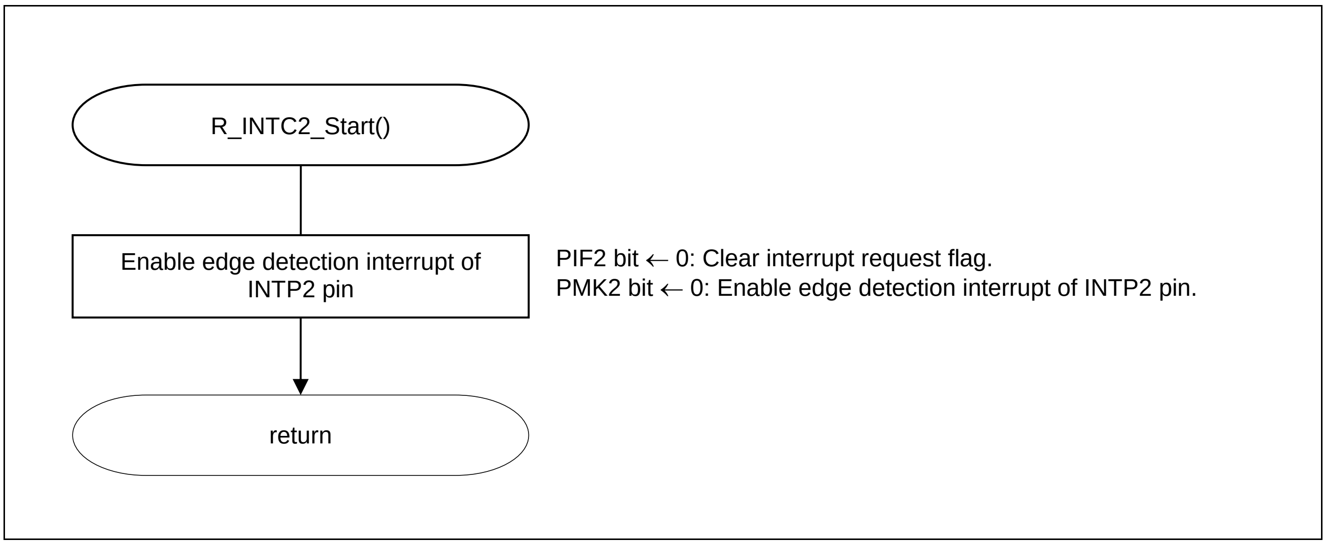

R_INTC2_Start()
Enable edge detection interrupt of INTP2 pin
PIF2 bit ← 0: Clear interrupt request flag.
PMK2 bit ← 0: Enable edge detection interrupt of INTP2 pin.
return

Figure 5.16 Starting the INTP2

5.9.13 INTP2 External Interrupt

Figure 5.17 shows the flowchart for INTP2 external interrupt.

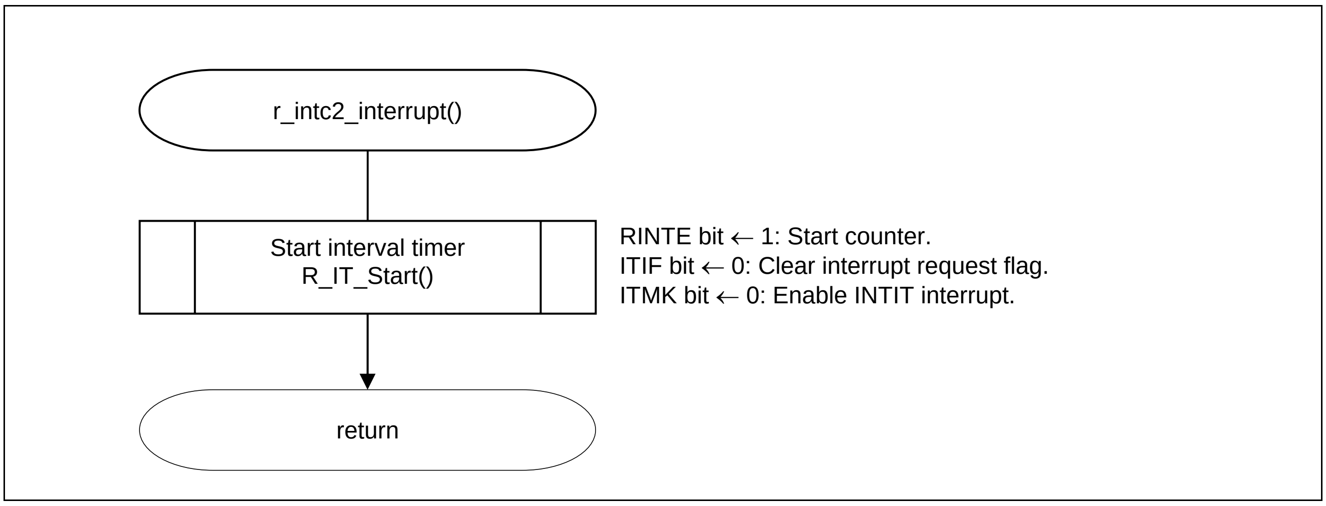

r_intc2_interrupt()
Start interval timer R_IT_Start()
RINTE bit ← 1: Start counter.
ITIF bit ← 0: Clear interrupt request flag.
ITMK bit ← 0: Enable INTIT interrupt.
return

Figure 5.17 INTP2 External Interrupt

5.9.14 Starting the INTP4

Figure 5.18 shows the flowchart for starting the INTP4.

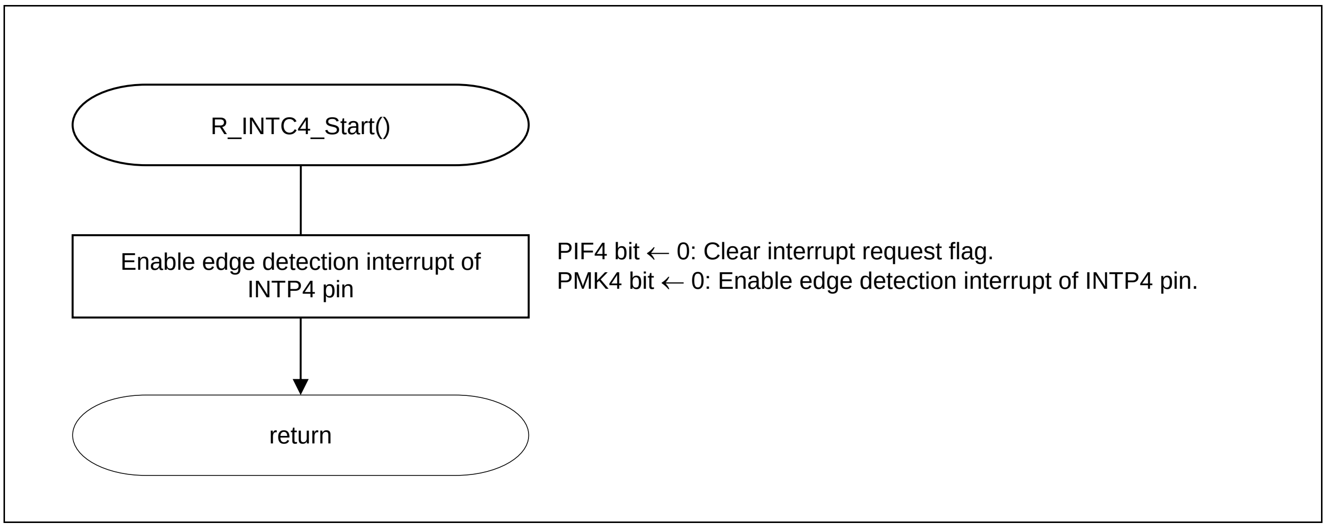

R_INTC4_Start()
Enable edge detection interrupt of INTP4 pin
PIF4 bit ← 0: Clear interrupt request flag.
PMK4 bit ← 0: Enable edge detection interrupt of INTP4 pin.
return

Figure 5.18 Starting the INTP4

5.9.15 INTP4 External Interrupt

Figure 5.19 shows the flowchart for INTP4 external interrupt.

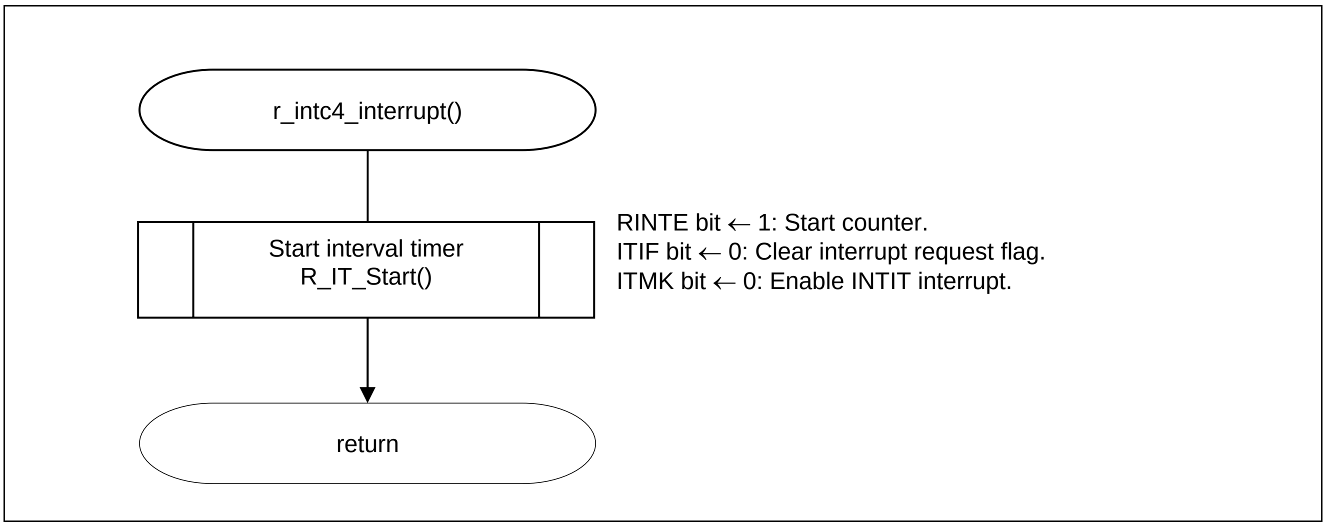

r_intc4_interrupt()
Start interval timer
R_IT_Start()
RINTE bit ← 1: Start counter.
ITIF bit ← 0: Clear interrupt request flag.
ITMK bit ← 0: Enable INTIT interrupt.
return

Figure 5.19 INTP4 External Interrupt

5.9.16 Starting the Interval Timer

Figure 5.20 shows the flowchart for starting the interval timer.

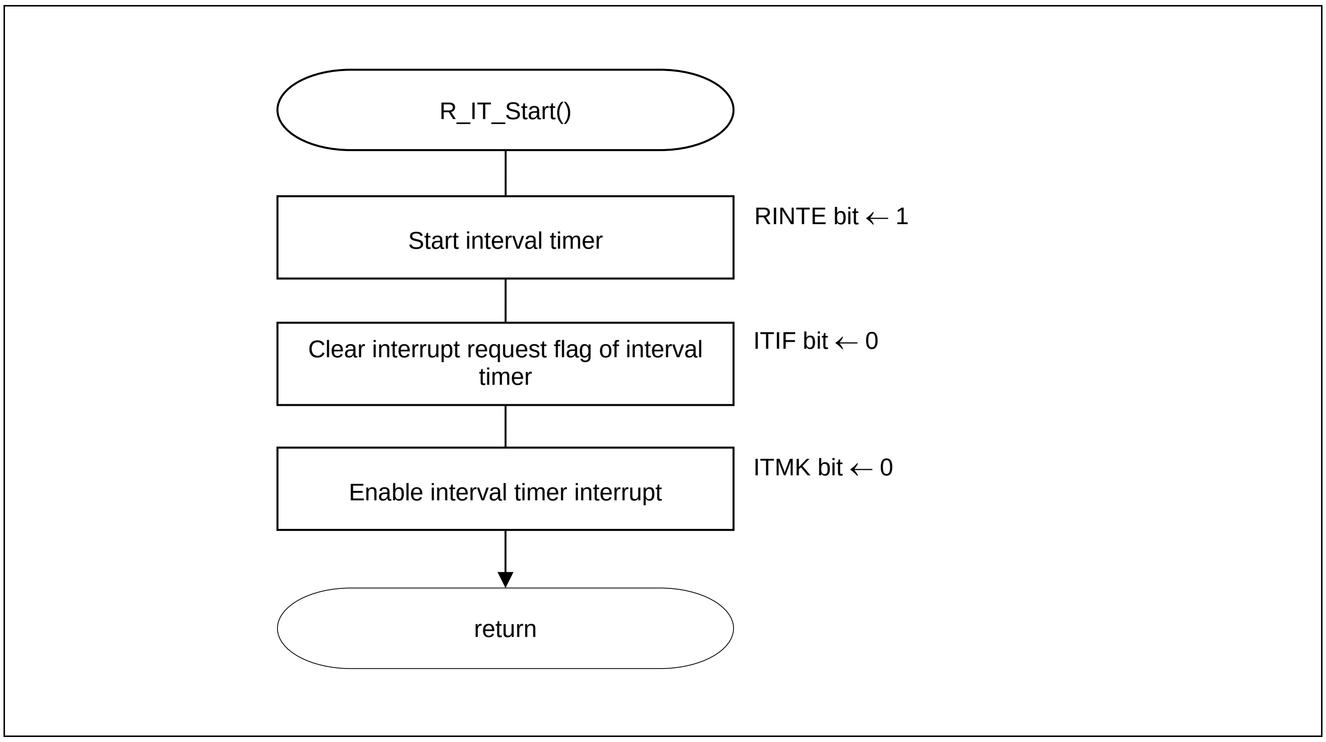

R_IT_Start()
Start interval timer
RINTE bit ← 1
Clear interrupt request flag of interval timer
ITIF bit ← 0
Enable interval timer interrupt
ITMK bit ← 0
return

Figure 5.20 Starting the Interval Timer

5.9.17 Interval Timer Interrupt

Figure 5.21 shows the flowchart for interval timer interrupt.

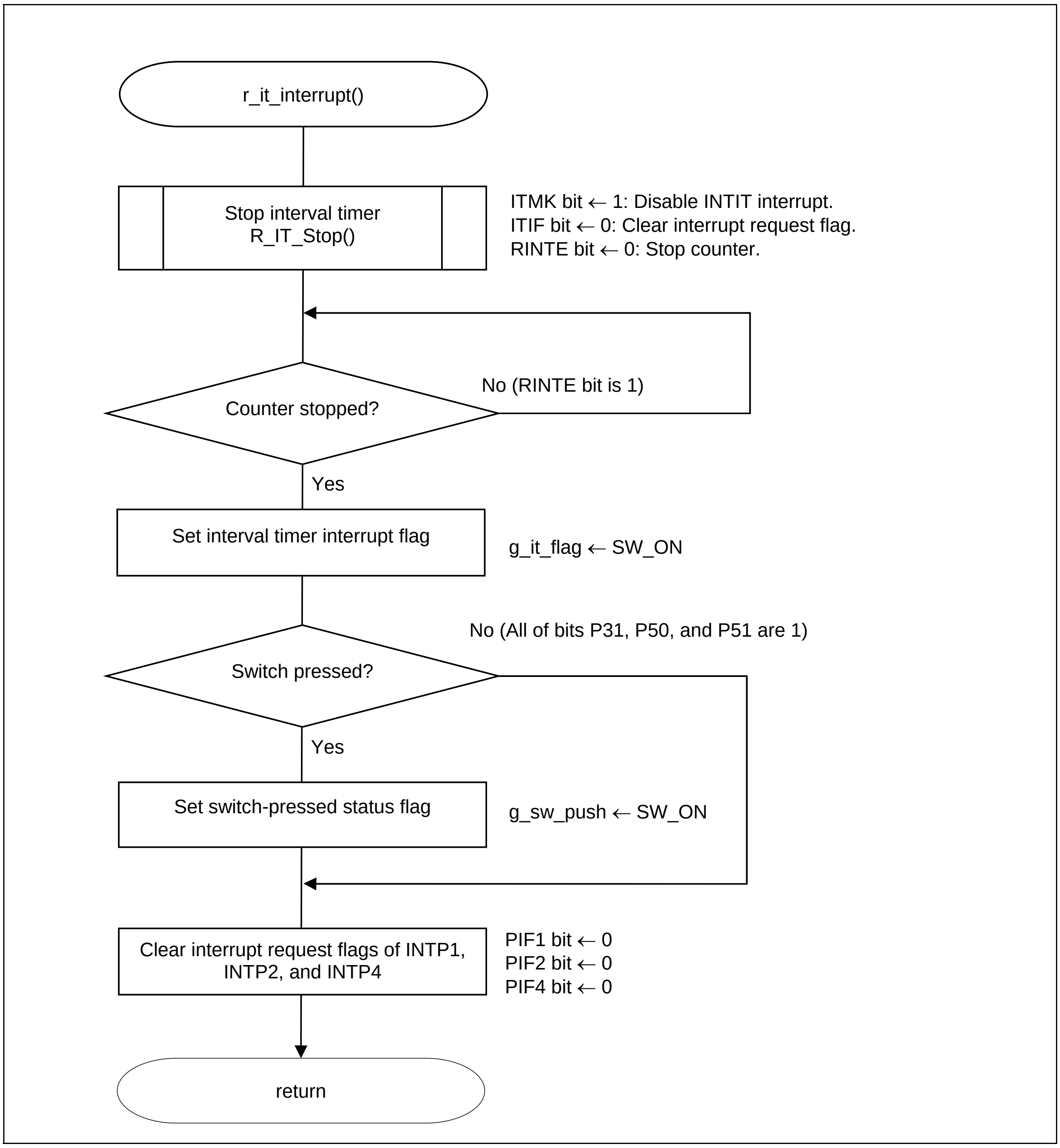

r_it_interrupt()
Stop interval timer
R_IT_Stop()
ITMK bit ← 1: Disable INTIT interrupt.
ITIF bit ← 0: Clear interrupt request flag.
RINTE bit ← 0: Stop counter.
Counter stopped?
No (RINTE bit is 1)
Yes
Set interval timer interrupt flag
g_it_flag ← SW_ON
Switch pressed?
No (All of bits P31, P50, and P51 are 1)
Yes
Set switch-pressed status flag
g_sw_push ← SW_ON
Clear interrupt request flags of INTP1, INTP2, and INTP4
PIF1 bit ← 0
PIF2 bit ← 0
PIF4 bit ← 0
return

Figure 5.21 Interval Timer Interrupt

5.9.18 Stopping the Interval Timer

Figure 5.22 shows the flowchart for stopping the interval timer.

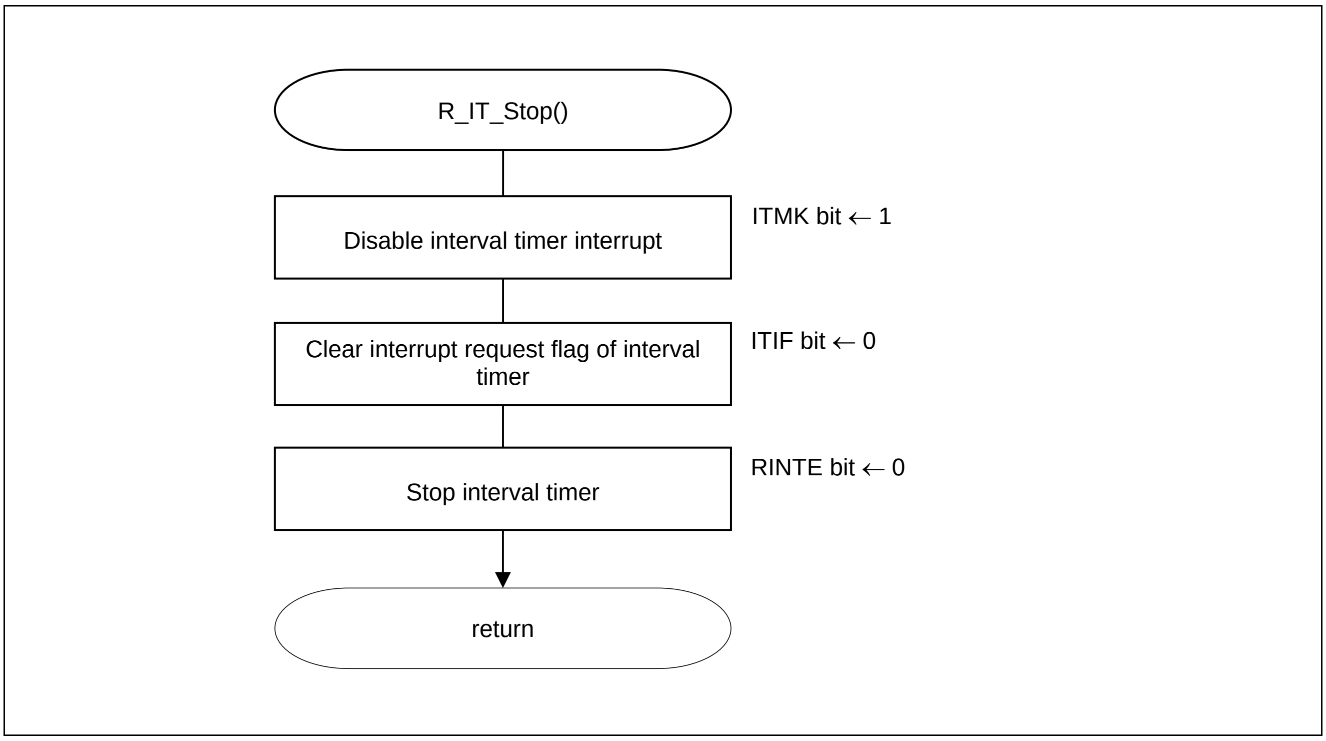

R_IT_Stop()
Disable interval timer interrupt
ITMK bit ← 1
Clear interrupt request flag of interval timer
ITIF bit ← 0
Stop interval timer
RINTE bit ← 0
return

Figure 5.22 Stopping the Interval Timer

5.9.19 Clearing a Switch-Pressed Status

Figure 5.23 shows the flowchart for clearing a switch-pressed status.

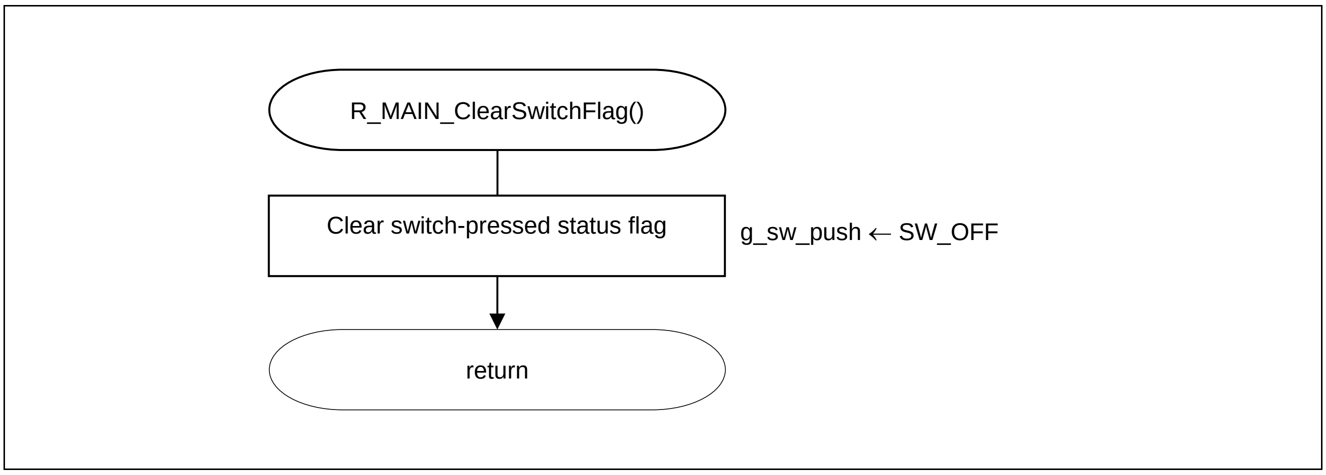


Figure 5.23 Clearing a Switch-Pressed Status

5.9.20 EEL Initialization

Figures 5.24 and 5.25 show the flowcharts of EEL initialization.

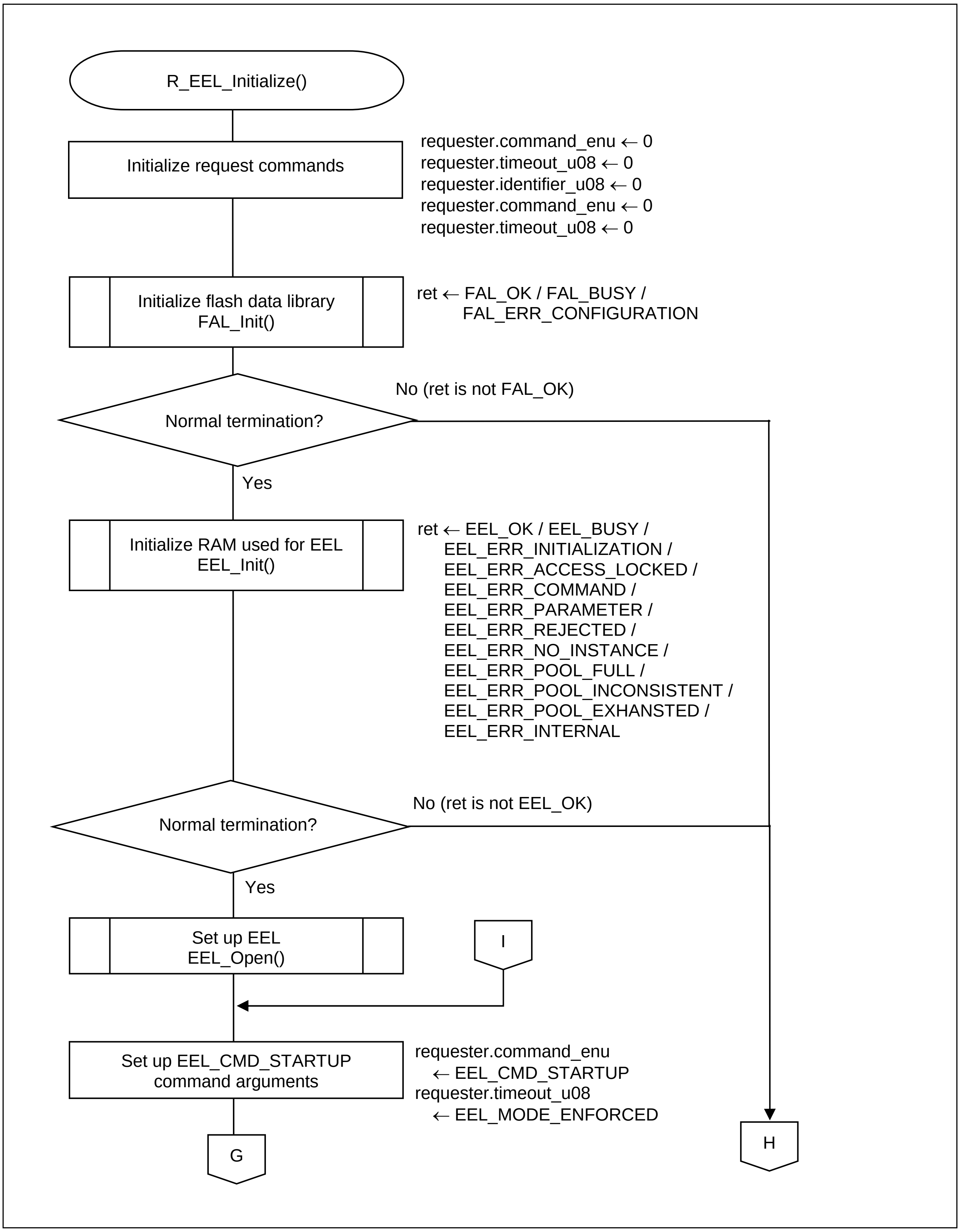


Figure 5.24 EEL Initialization (1/2)

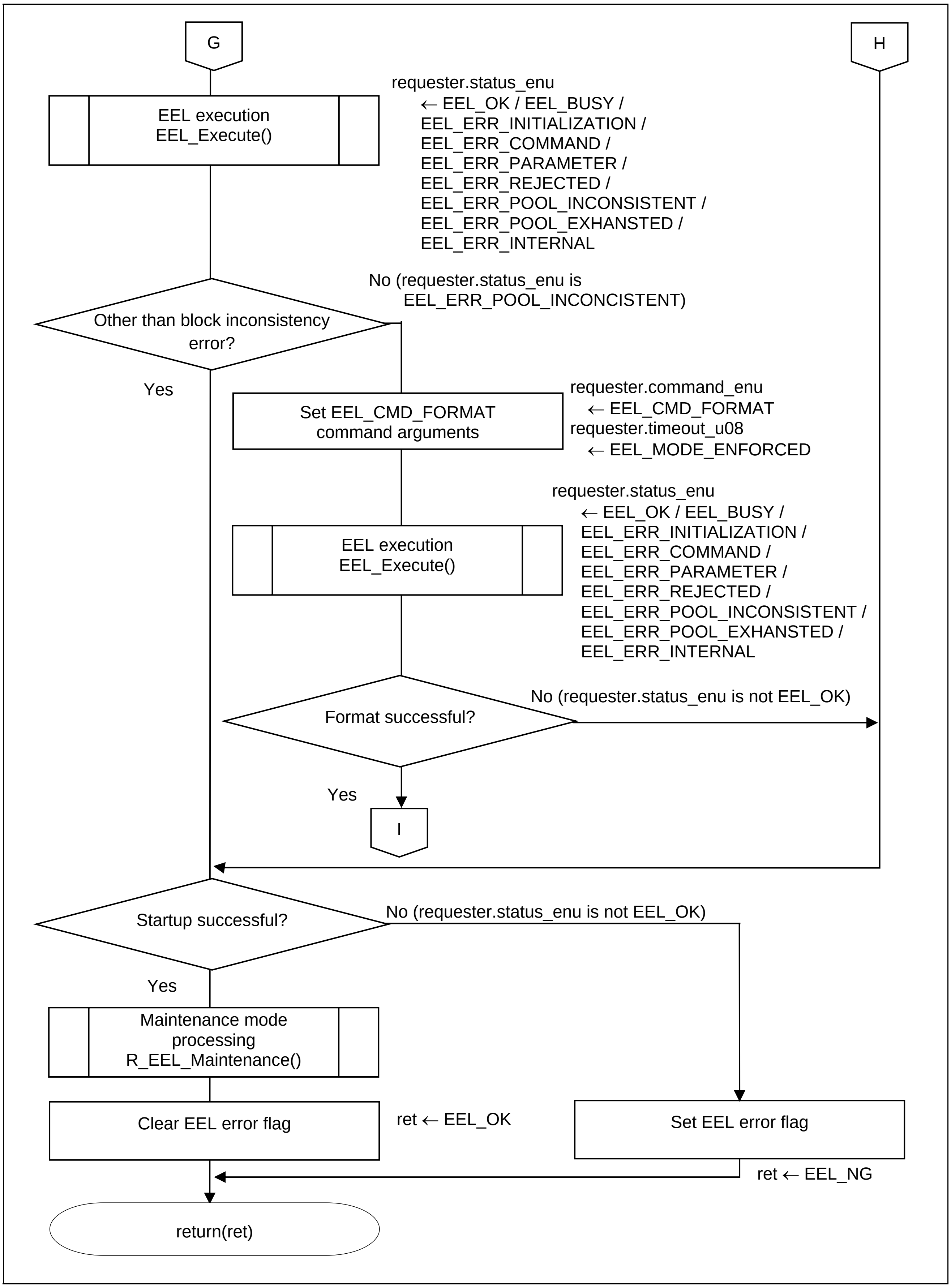


Figure 5.25 EEL Initialization (2/2)

5.9.21 Maintenance Mode Processing

Figure 5.26 shows the flowchart for maintenance mode processing.

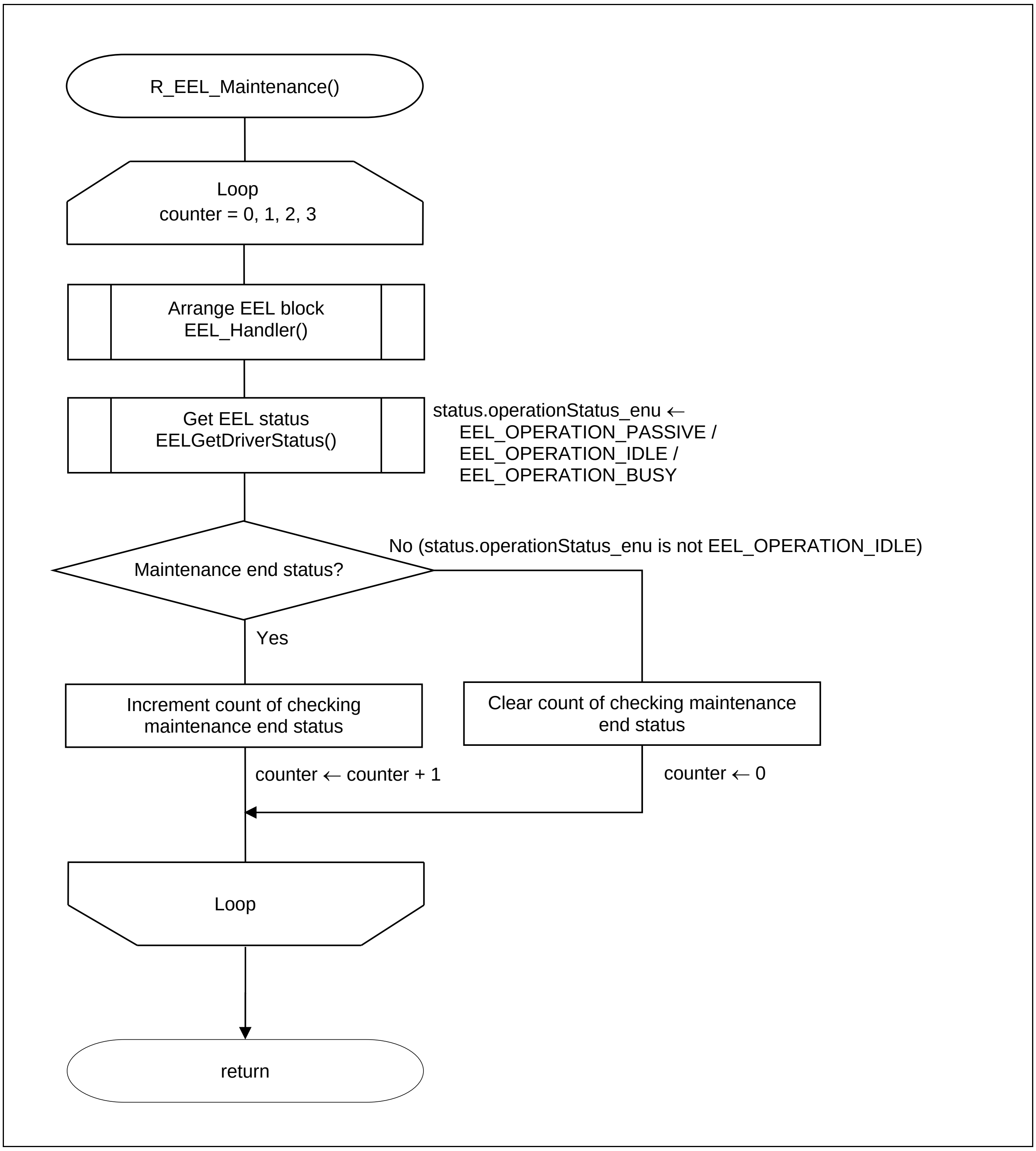


Figure 5.26 Maintenance Mode Processing

5.9.22 Getting a Switch Status

Figure 5.27 shows the flowchart for getting a switch status.

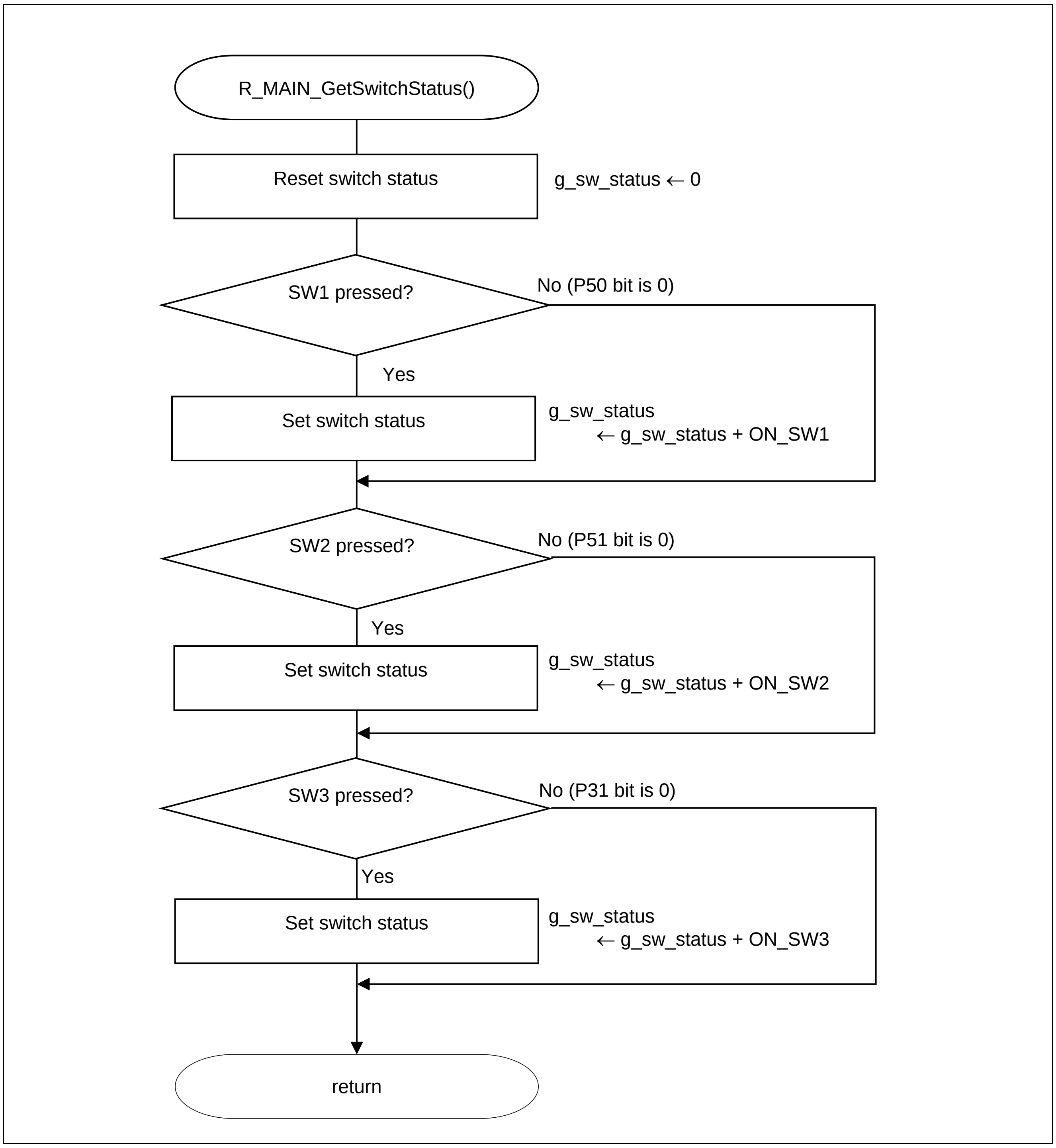


Figure 5.27 Getting a Switch Status

5.9.23 Processing of Switch-Pressed Status

Figure 5.28 shows the flowchart for processing a switch-pressed status.

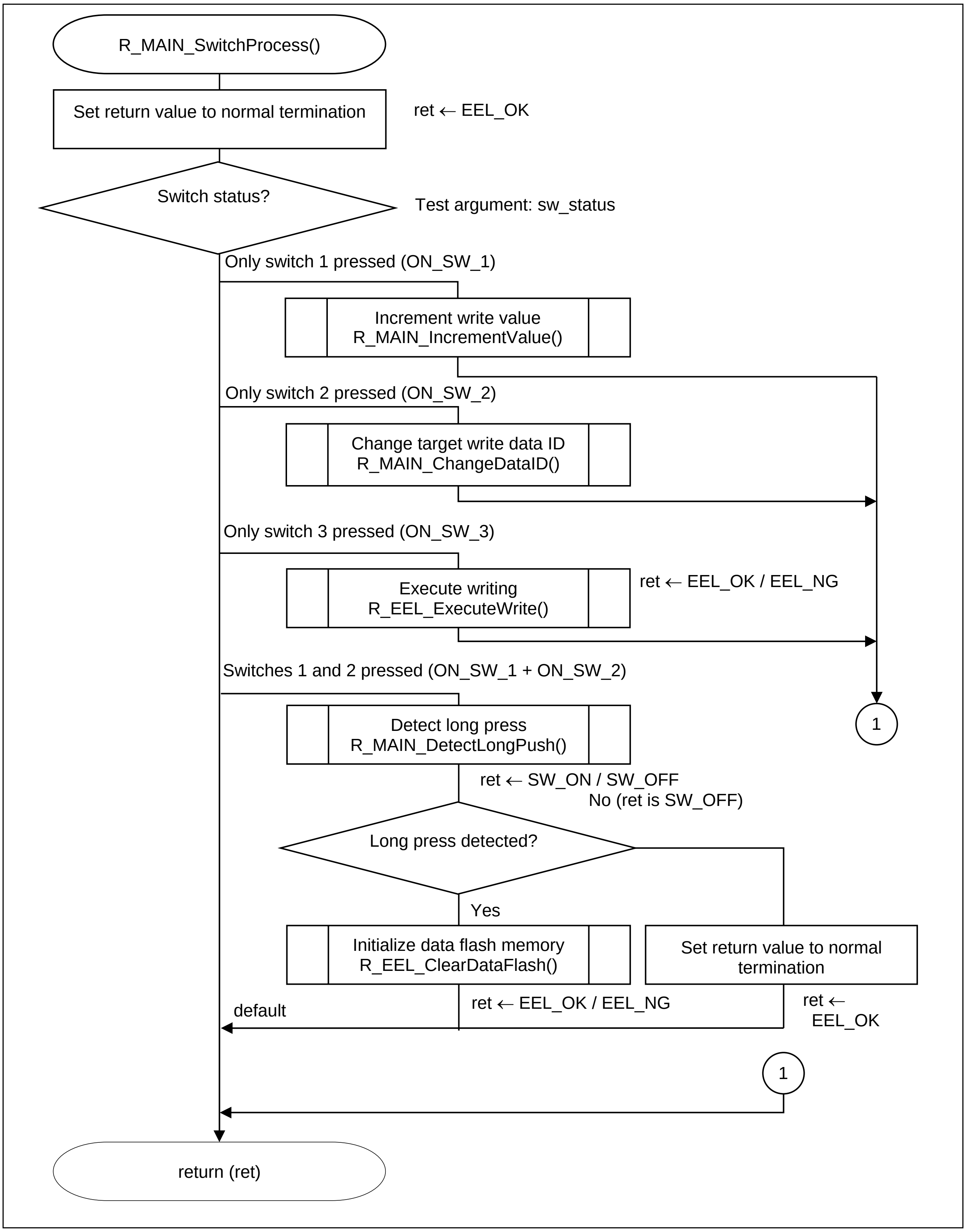


Figure 5.28 Processing of Switch-Pressed Status

5.9.24 Increment of Write Value

Figure 5.29 shows the flowchart for incrementing a write value.

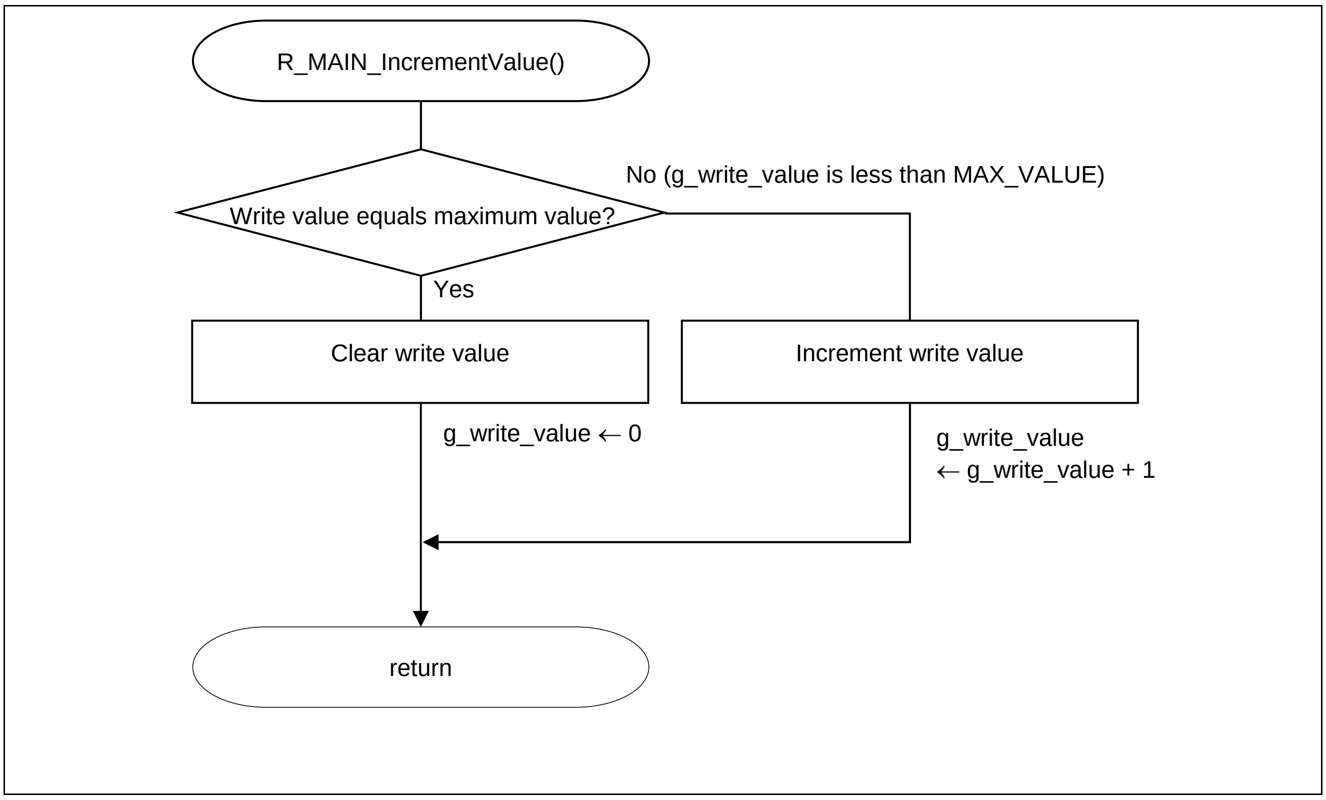


Figure 5.29 Increment of Write Value

5.9.25 Change of Target Write Data ID

Figure 5.30 shows the flowchart for changing a target write data ID.

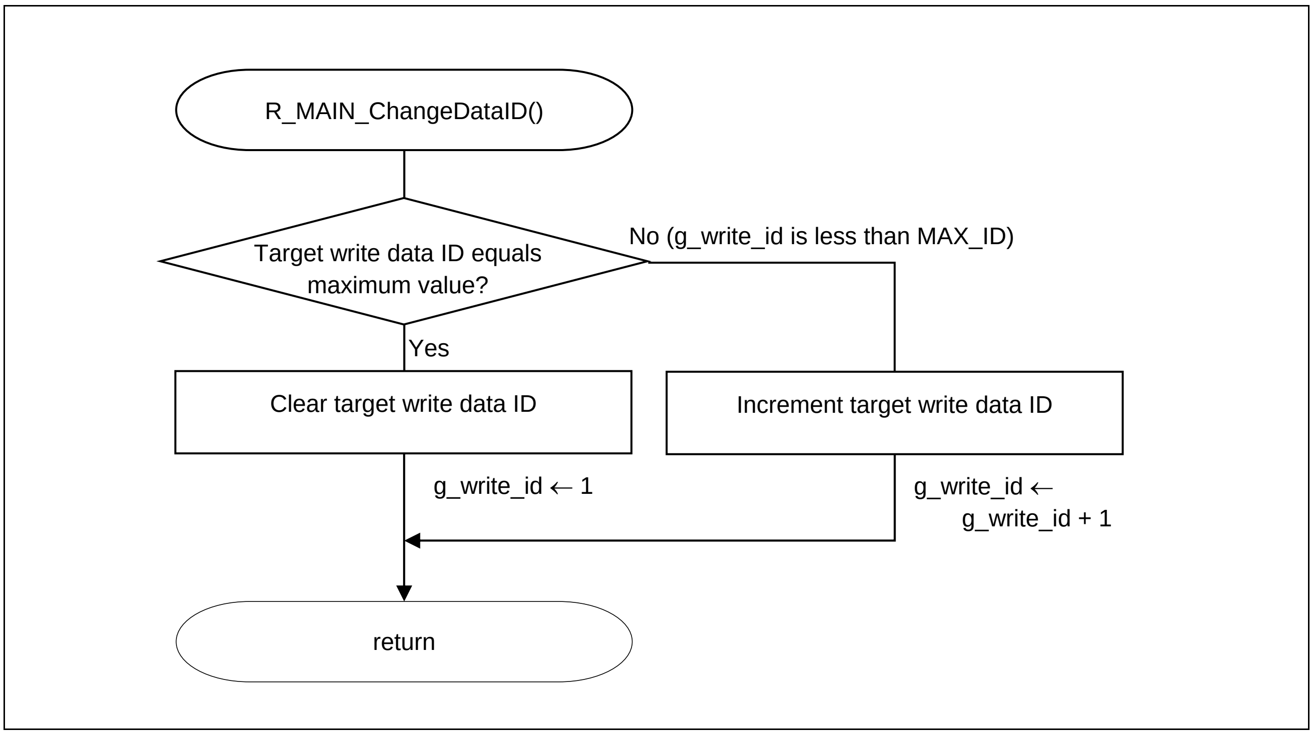

R_MAIN_ChangeDataID()
Target write data ID equals maximum value?
No (g_write_id is less than MAX_ID)
Yes
Clear target write data ID
Increment target write data ID
g_write_id ← 1
g_write_id ← g_write_id + 1
return

Figure 5.30 Change of Target Write Data ID

5.9.26 Writing Execution

Figures 5.31 and 5.32 show the flowcharts for writing execution.

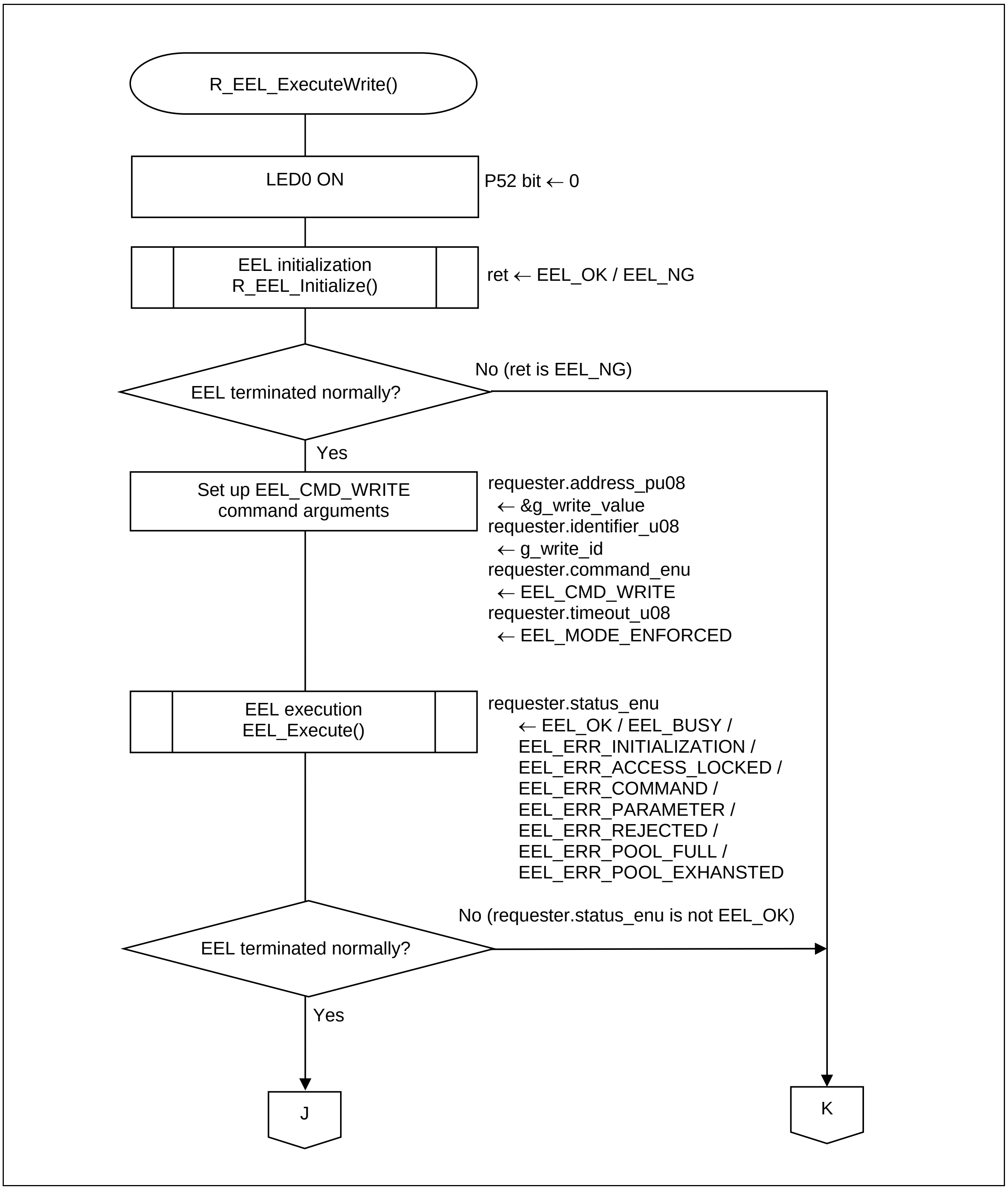

R_EEL_ExecuteWrite()
LED0 ON
P52 bit ← 0
EEL initialization
R_EEL_Initialize()
ret ← EEL_OK / EEL_NG
EEL terminated normally?
No (ret is EEL_NG)
Yes
Set up EEL_CMD_WRITE command arguments
requester.address_pu08 ← &g_write_value
requester.identifier_u08 ← g_write_id
requester.command_enu ← EEL_CMD_WRITE
requester.timeout_u08 ← EEL_MODE_ENFORCED
EEL execution
EEL_Execute()
requester.status_enu ← EEL_OK / EEL_BUSY / EEL_ERR_INITIALIZATION / EEL_ERR_ACCESS_LOCKED / EEL_ERR_COMMAND / EEL_ERR_PARAMETER / EEL_ERR_REJECTED / EEL_ERR_POOL_FULL / EEL_ERR_POOL_EXHANSTED
EEL terminated normally?
No (requester.status_enu is not EEL_OK)
Yes
J
K

Figure 5.31 Writing Execution (1/2)

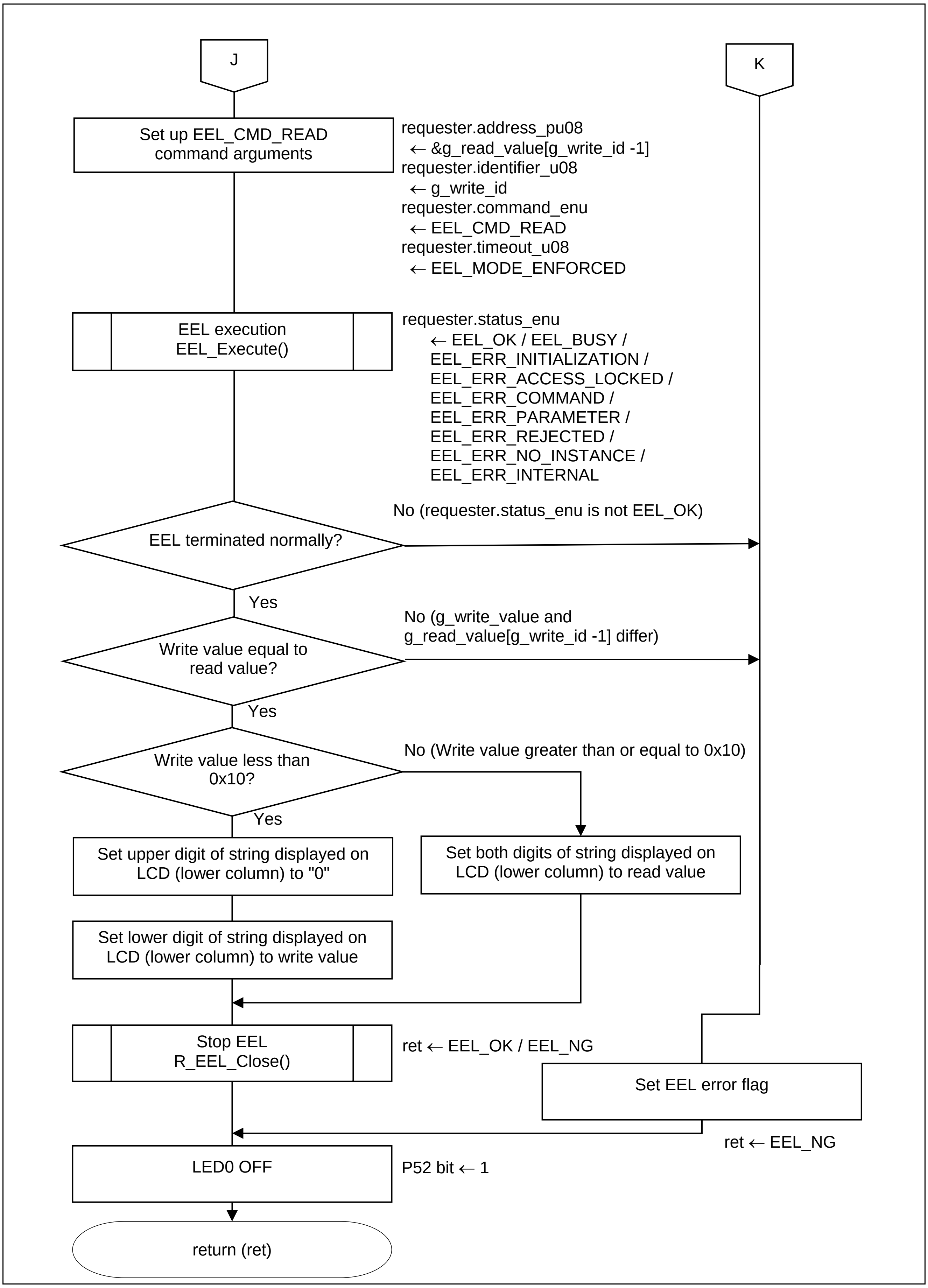

J
K
Set up EEL_CMD_READ command arguments
requester.address_pu08
← &g_read_value[g_write_id -1]
requester.identifier_u08
← g_write_id
requester.command_enu
← EEL_CMD_READ
requester.timeout_u08
← EEL_MODE_ENFORCED
EEL execution EEL_Execute()
requester.status_enu
← EEL_OK / EEL_BUSY /
EEL_ERR_INITIALIZATION /
EEL_ERR_ACCESS_LOCKED /
EEL_ERR_COMMAND /
EEL_ERR_PARAMETER /
EEL_ERR_REJECTED /
EEL_ERR_NO_INSTANCE /
EEL_ERR_INTERNAL
EEL terminated normally?
No (requester.status_enu is not EEL_OK)
Yes
Write value equal to read value?
No (g_write_value and g_read_value[g_write_id -1] differ)
Yes
Write value less than 0x10?
No (Write value greater than or equal to 0x10)
Yes
Set upper digit of string displayed on LCD (lower column) to "0"
Set both digits of string displayed on LCD (lower column) to read value
Set lower digit of string displayed on LCD (lower column) to write value
Stop EEL R_EEL_Close()
ret ← EEL_OK / EEL_NG
Set EEL error flag
ret ← EEL_NG
LED0 OFF
P52 bit ← 1
return (ret)

Figure 5.32 Writing Execution (2/2)

5.9.27 Detection of Long Press

Figures 5.33 and 5.34 show the flowcharts for detection of long press.

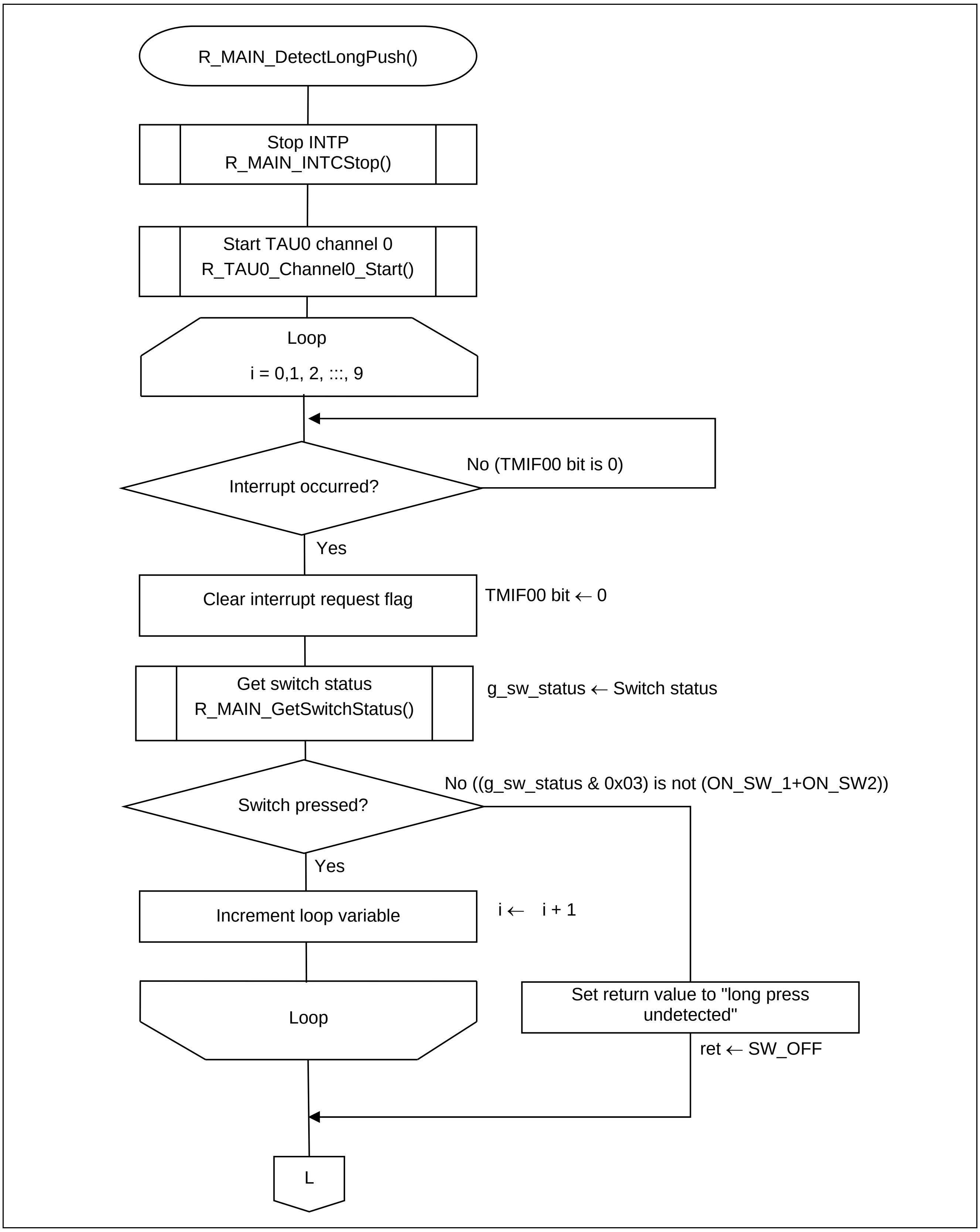


Figure 5.33 Detection of Long Press (1/2)

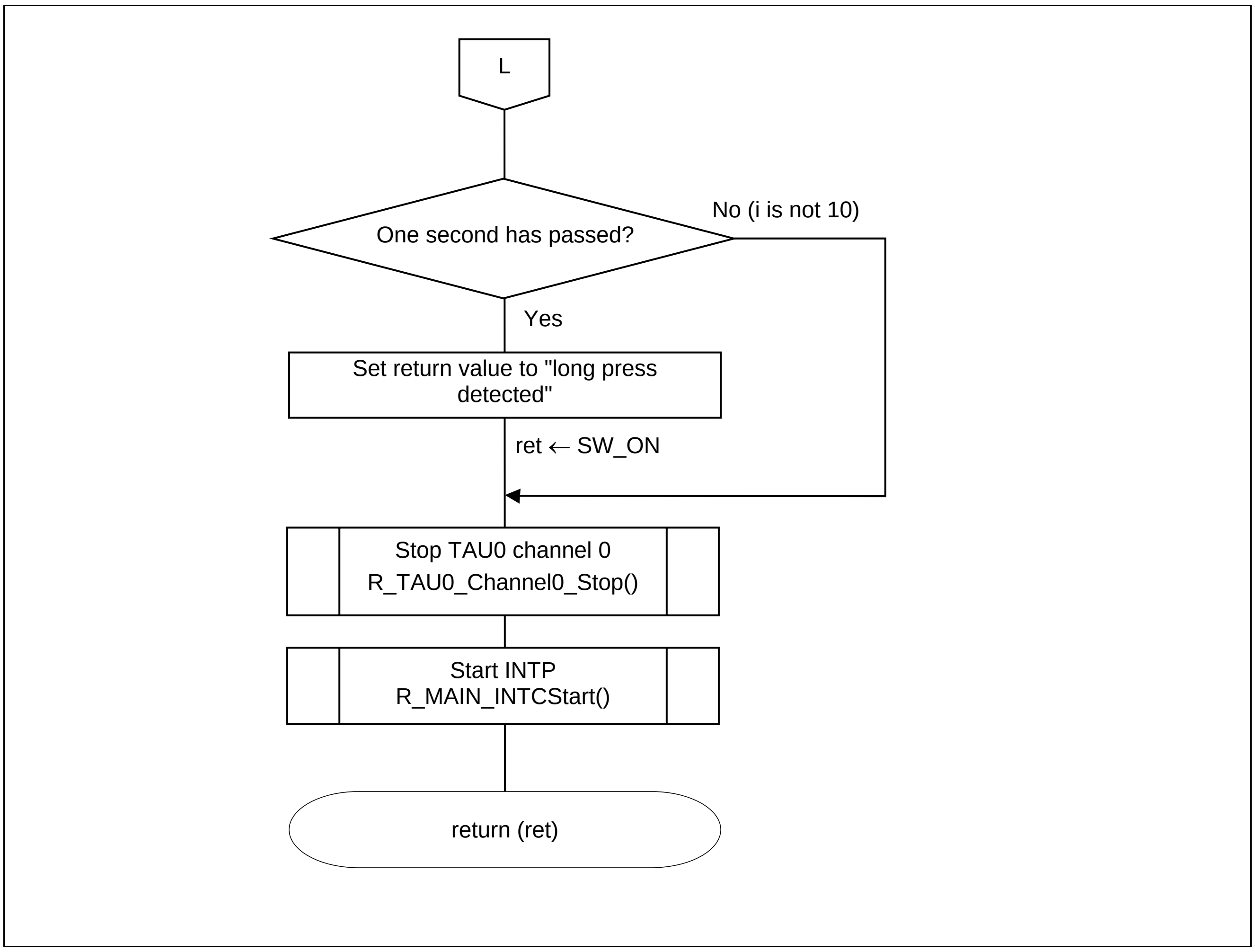


Figure 5.34 Detection of Long Press (2/2)

5.9.28 Stopping the INTP

Figure 5.35 shows the flowchart for stopping the INTP.

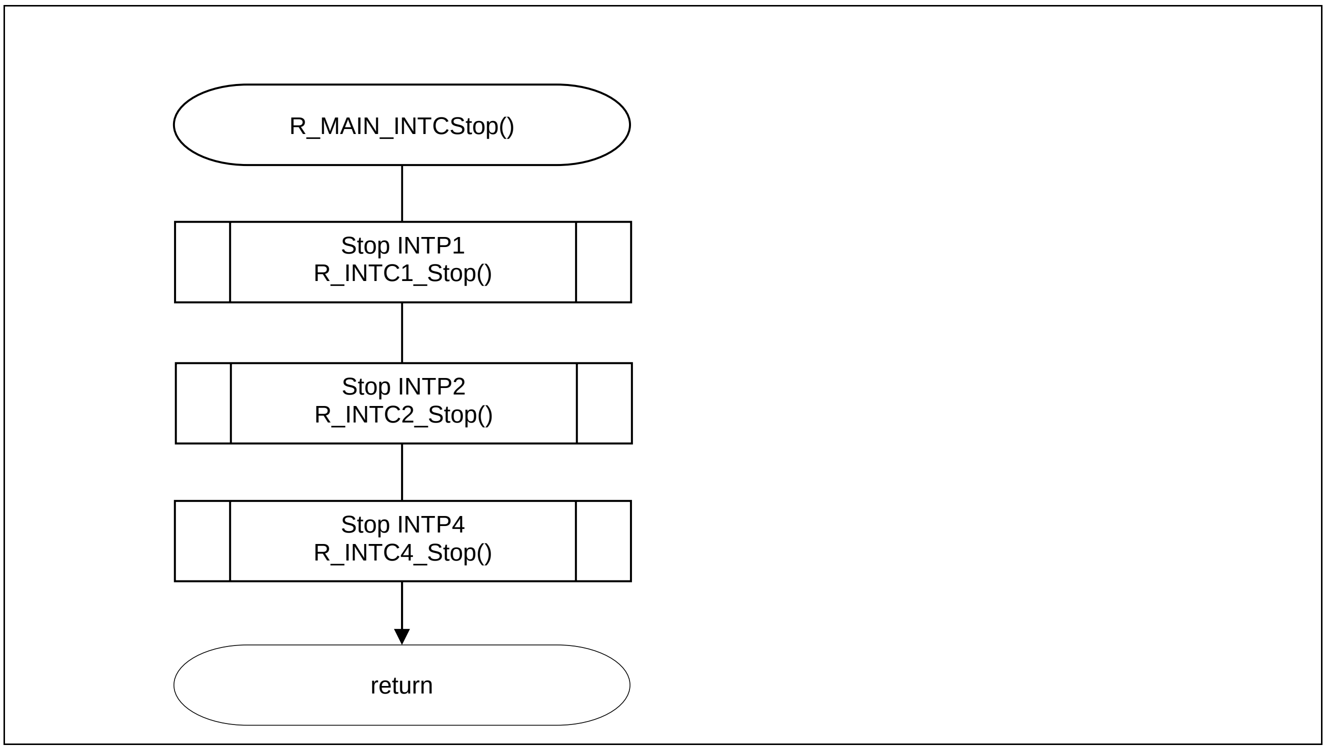


Figure 5.35 Stopping the INTP

5.9.29 Stopping the INTP1

Figure 5.36 shows the flowchart for stopping the INTP1.

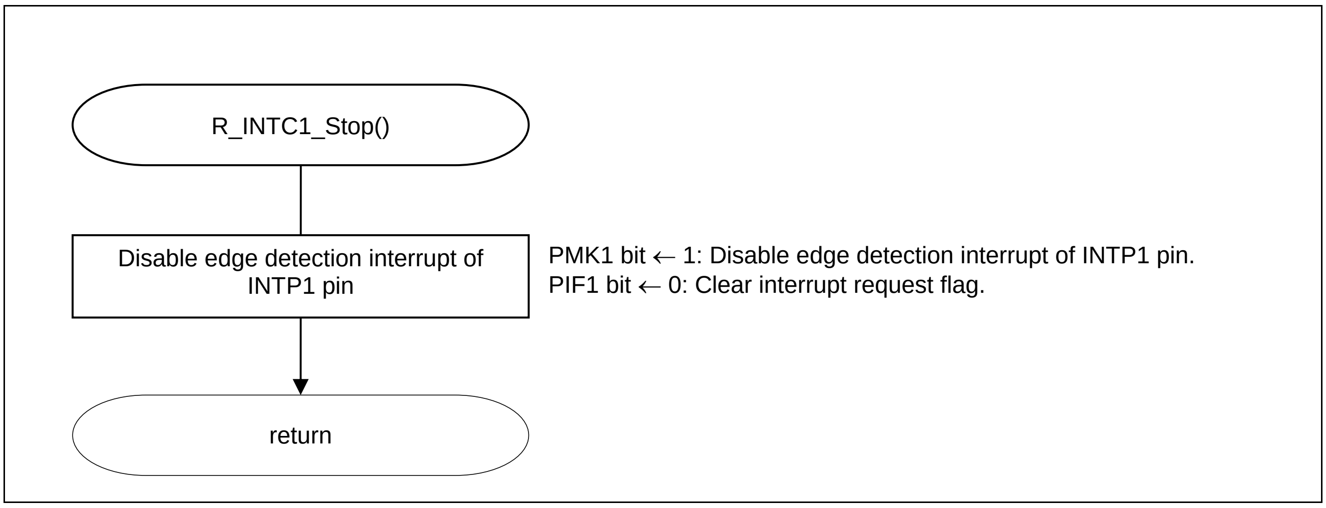

R_INTC1_Stop()
Disable edge detection interrupt of INTP1 pin
PMK1 bit ← 1: Disable edge detection interrupt of INTP1 pin.
PIF1 bit ← 0: Clear interrupt request flag.
return

Figure 5.36 Stopping the INTP1

5.9.30 Stopping the INTP2

Figure 5.37 shows the flowchart for stopping the INTP2.

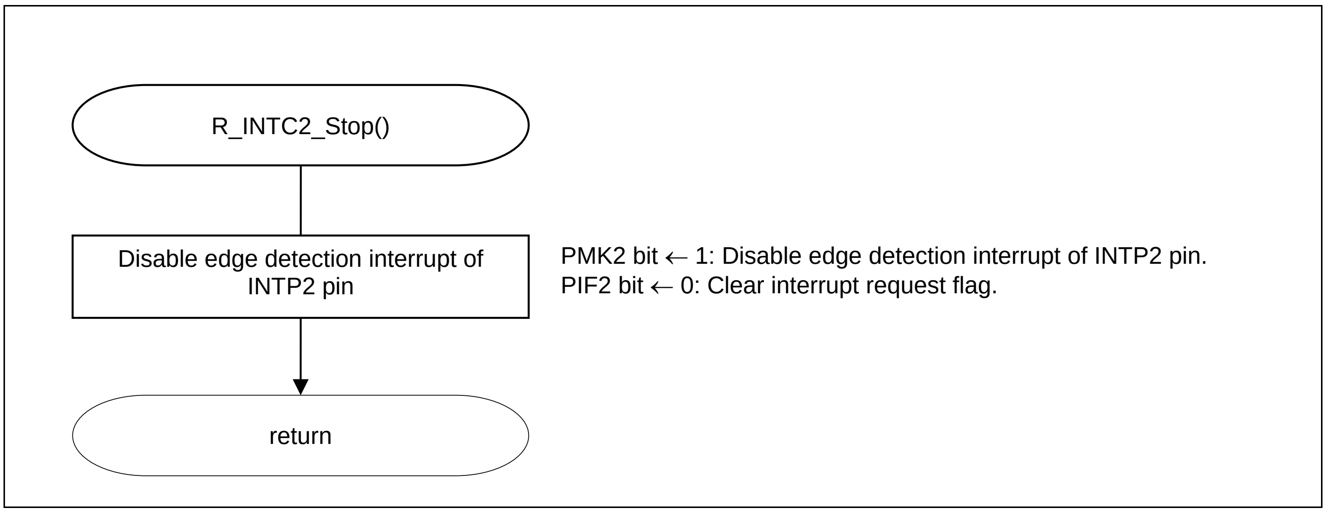

R_INTC2_Stop()
Disable edge detection interrupt of INTP2 pin
PMK2 bit ← 1: Disable edge detection interrupt of INTP2 pin.
PIF2 bit ← 0: Clear interrupt request flag.
return

Figure 5.37 Stopping the INTP2

5.9.31 Stopping the INTP4

Figure 5.38 shows the flowchart for stopping the INTP4.

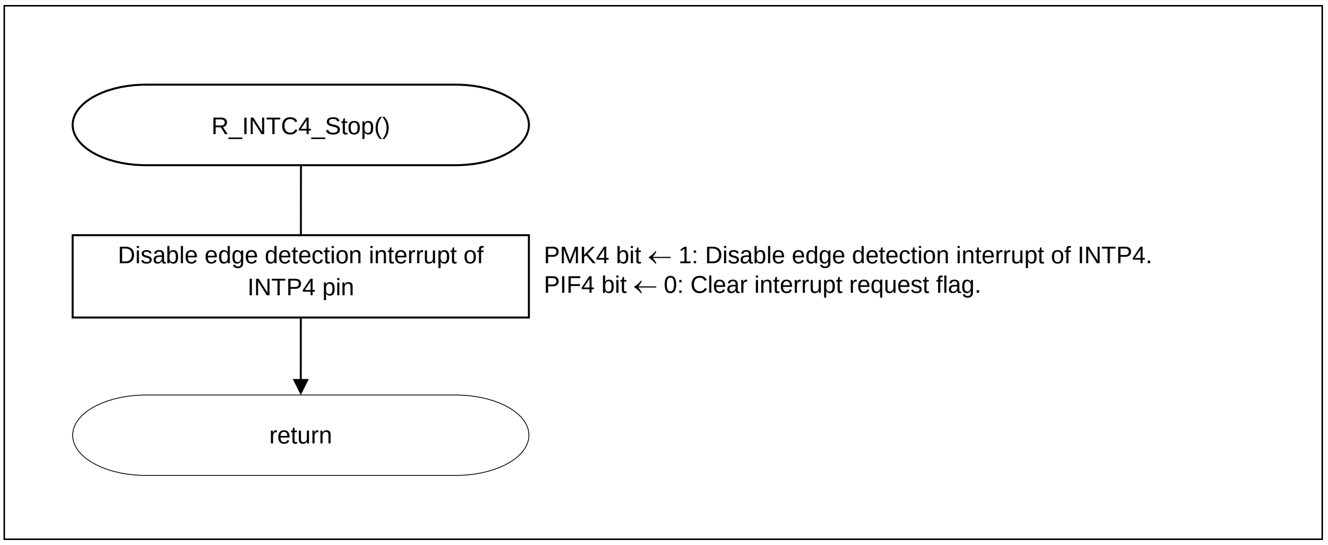

R_INTC4_Stop()
Disable edge detection interrupt of INTP4 pin
PMK4 bit ← 1: Disable edge detection interrupt of INTP4.
PIF4 bit ← 0: Clear interrupt request flag.
return

Figure 5.38 Stopping the INTP4

5.9.32 Starting the TAU0 Channel 0

Figure 5.39 shows the flowchart for starting the TAU0 channel 0.

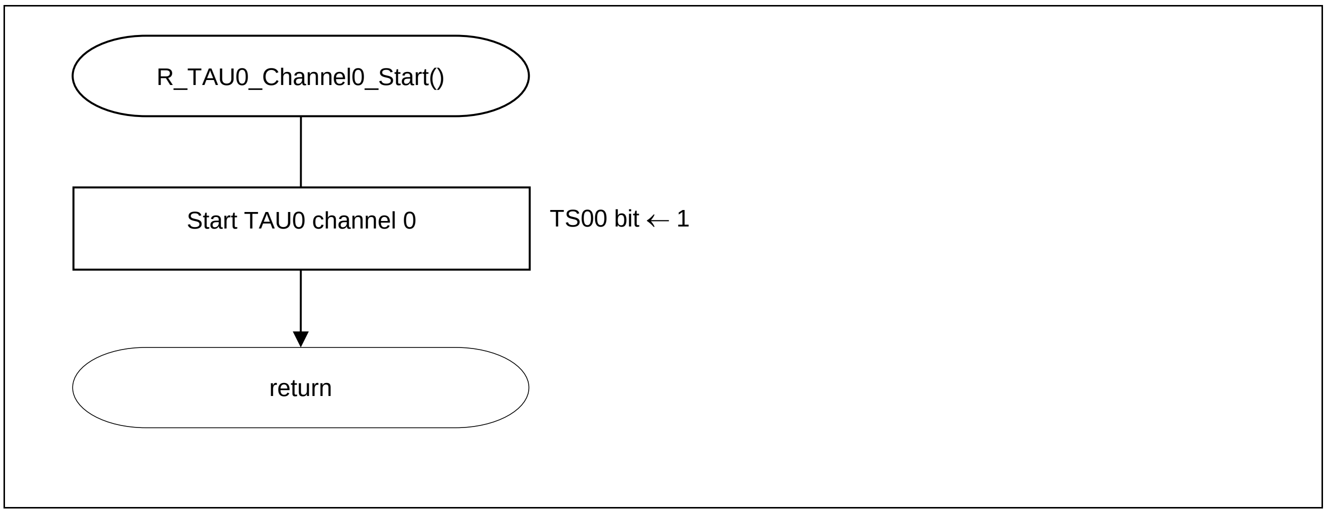


Figure 5.39 Starting the TAU0 Channel 0

5.9.33 Stopping the TAU0 Channel 0

Figure 5.40 shows the flowchart for stopping the TAU0 channel 0.

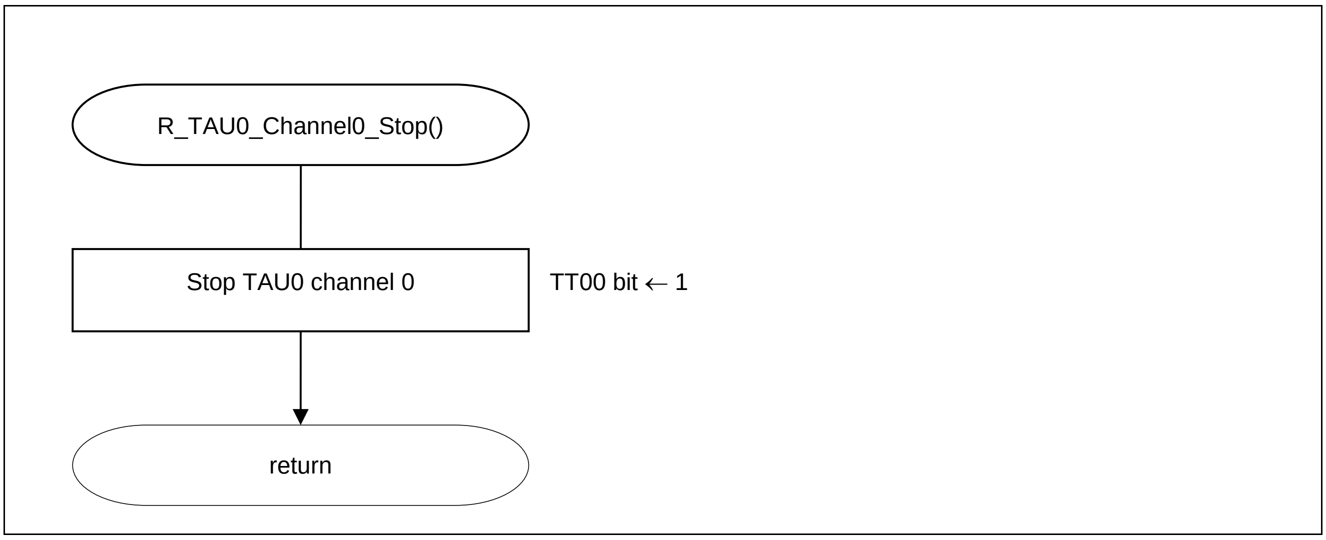
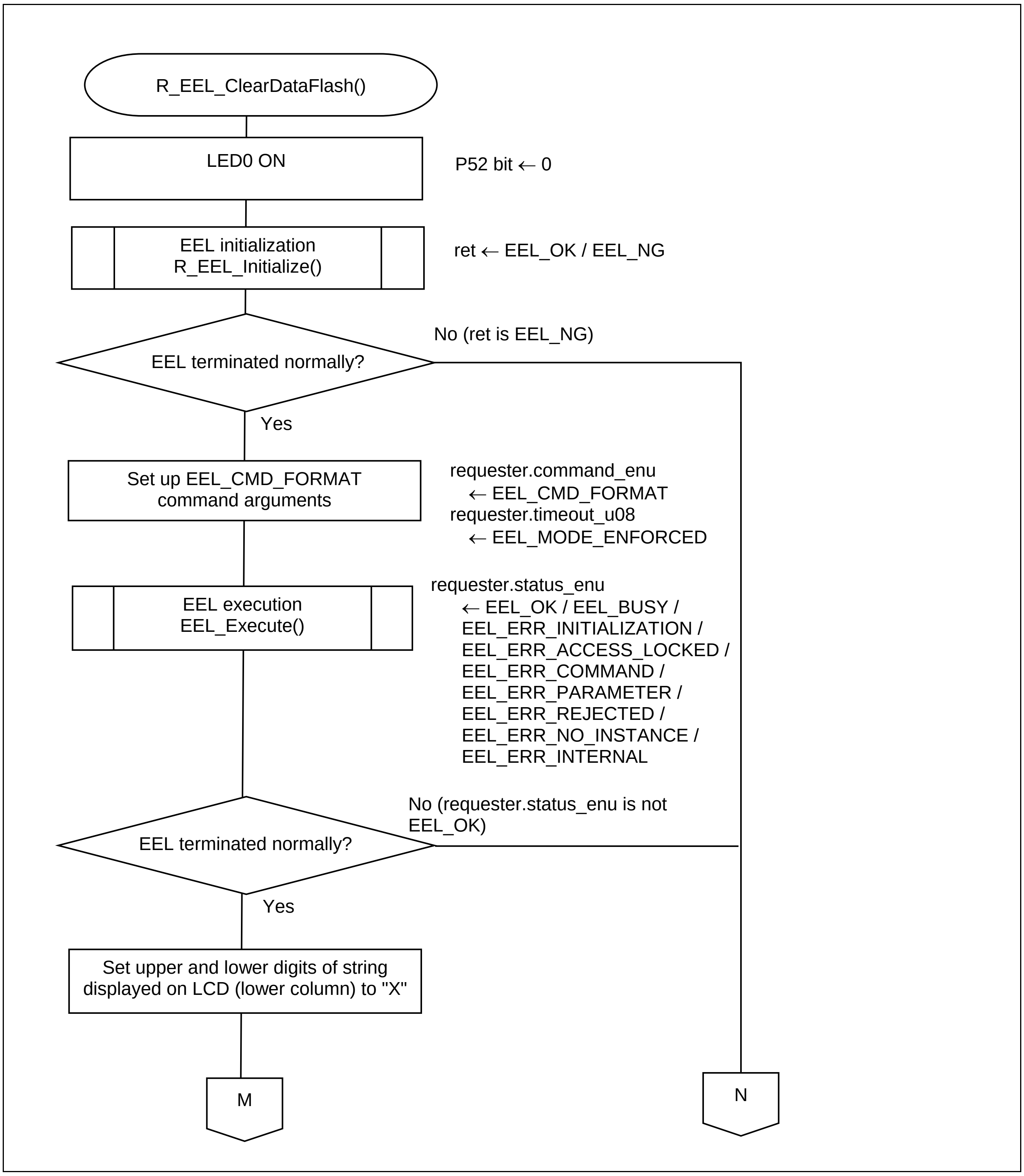


Figure 5.40 Stopping the TAU0 Channel 0

5.9.34 Data Flash Memory Initialization

Figures 5.41 and 5.42 show the flowcharts for data flash memory initialization.



R_EEL_ClearDataFlash()
LED0 ON
P52 bit ← 0
EEL initialization
R_EEL_Initialize()
ret ← EEL_OK / EEL_NG
EEL terminated normally?
No (ret is EEL_NG)
Yes
Set up EEL_CMD_FORMAT command arguments
requester.command_enu ← EEL_CMD_FORMAT
requester.timeout_u08 ← EEL_MODE_ENFORCED
EEL execution
EEL_Execute()
requester.status_enu ← EEL_OK / EEL_BUSY / EEL_ERR_INITIALIZATION / EEL_ERR_ACCESS_LOCKED / EEL_ERR_COMMAND / EEL_ERR_PARAMETER / EEL_ERR_REJECTED / EEL_ERR_NO_INSTANCE / EEL_ERR_INTERNAL
EEL terminated normally?
No (requester.status_enu is not EEL_OK)
Yes
Set upper and lower digits of string displayed on LCD (lower column) to "X"
M
N

Figure 5.41 Data Flash Memory Initialization (1/2)

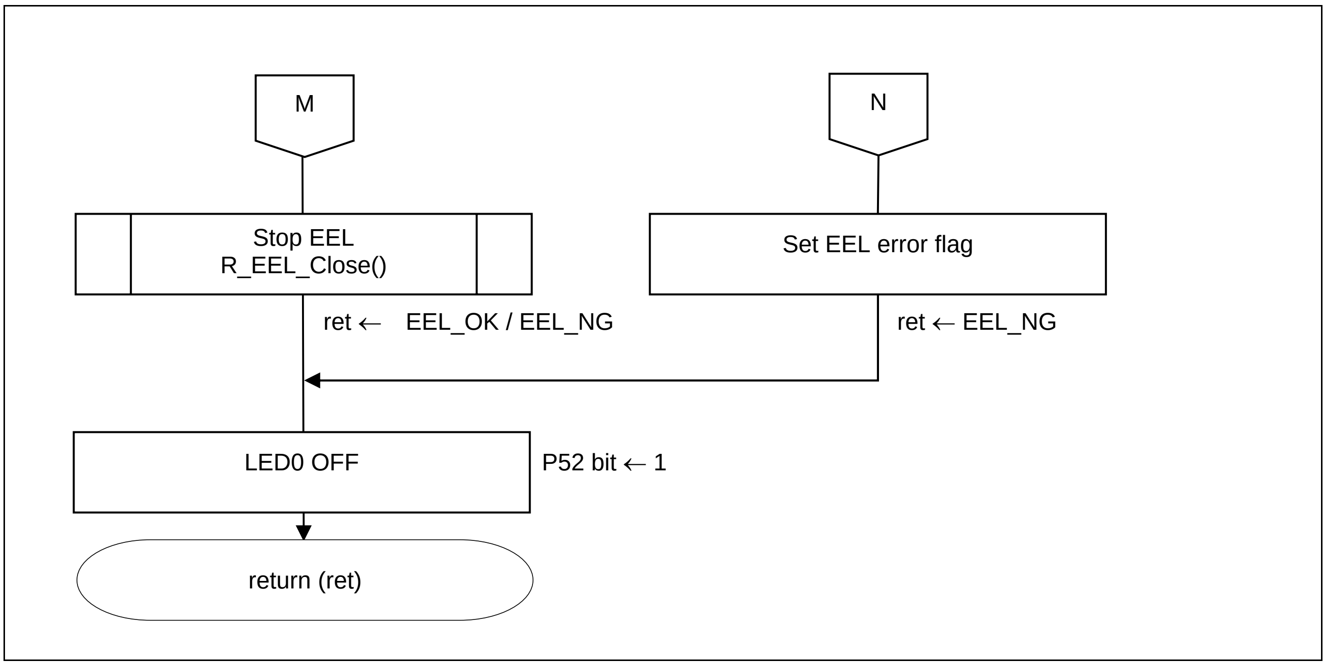


Figure 5.42 Data Flash Memory Initialization (2/2)

5.9.35 Stopping the EEL

Figure 5.43 shows the flowchart for stopping the EEL.

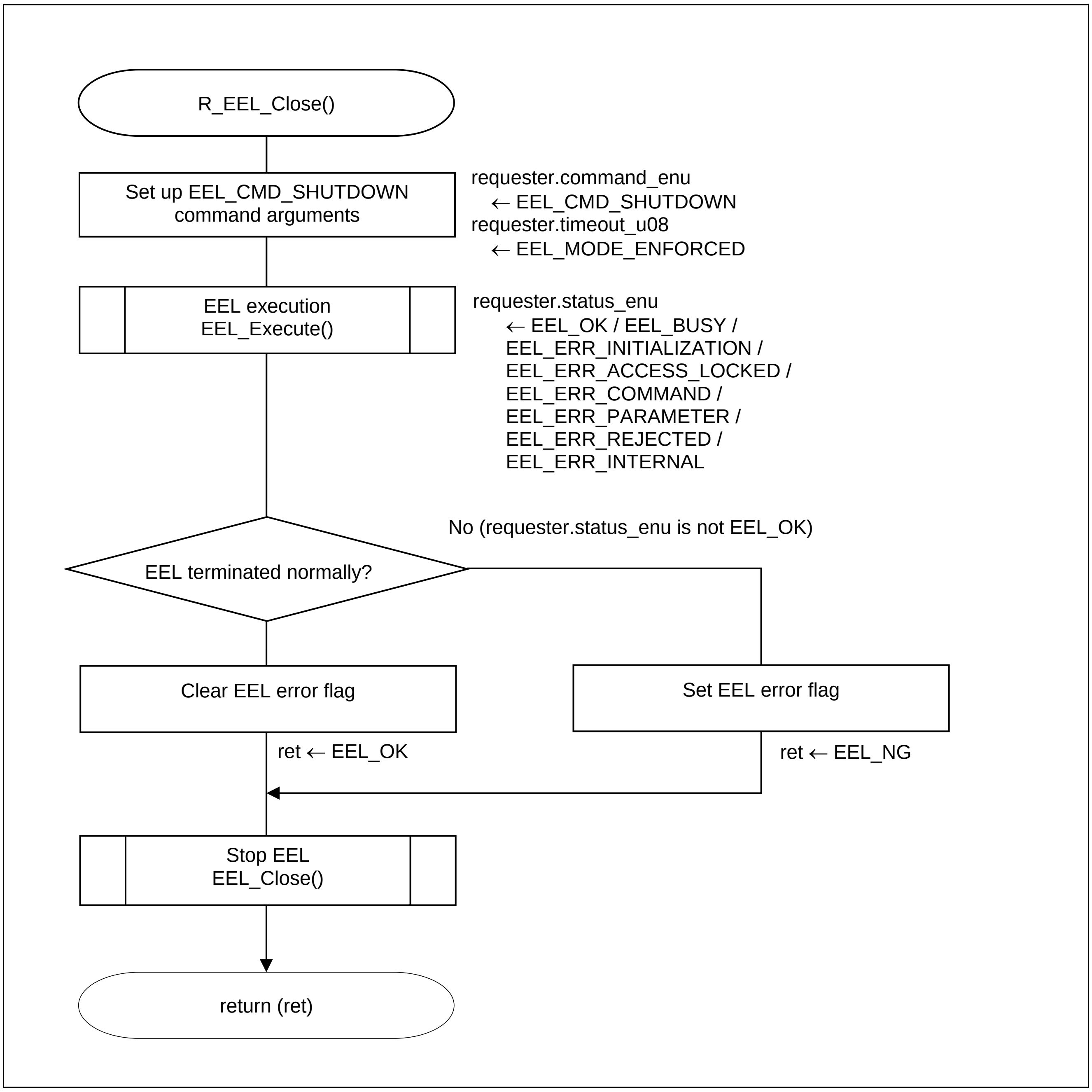

R_EEL_Close()
Set up EEL_CMD_SHUTDOWN command arguments
requester.command_enu ← EEL_CMD_SHUTDOWN
requester.timeout_u08 ← EEL_MODE_ENFORCED
EEL execution EEL_Execute()
requester.status_enu ← EEL_OK / EEL_BUSY / EEL_ERR_INITIALIZATION / EEL_ERR_ACCESS_LOCKED / EEL_ERR_COMMAND / EEL_ERR_PARAMETER / EEL_ERR_REJECTED / EEL_ERR_INTERNAL
EEL terminated normally?
No (requester.status_enu is not EEL_OK)
Clear EEL error flag
ret ← EEL_OK
Set EEL error flag
ret ← EEL_NG
Stop EEL EEL_Close()
return (ret)

Figure 5.43 Stopping the EEL

6. Sample Code

The sample code is available on the Renesas Electronics Website.

7. Documents for Reference

RL78/G13 User's Manual: Hardware (R01UH0146E)

RL78 Family User's Manual: Software (R01US0015E)

(The latest versions of the documents are available on the Renesas Electronics Website.)

Technical Updates/Technical Brochures

(The latest versions of the documents are available on the Renesas Electronics Website.)

Website and Support

Renesas Electronics Website

- http://www.renesas.com/index.jsp

Inquiries

- http://www.renesas.com/contact/

Revision Record	RL78/G13 EEPROM Emulation Library Pack01

Rev.	Date	Description	
		Page	Summary
1.00	Mar.01, 2013	—	First edition issued
1.10	June 01, 2016	11	Modification of 1.4 How to Get the EEPROM Emulation Library.

RENESAS

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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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: +44-1628-585-100, Fax: +44-1628-585-900

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, P. R. China 200333
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 II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0