SH7734 Group

SH7734 Example of Initialization

R01AN0665EJ0101
Rev.1.01
Jul 18, 2012

Introduction

This application note describes an example illustrating items that must be set to configure the device for CS0 boot after a power-on reset.

SH7734 Boot Modes

- A mode setting at startup can be used to select the boot behavior of the SH7734 from among the following: boot from memory connected to the CS0 space, boot from NAND flash memory, boot from serial flash memory, boot from flash memory connected to the MMC, boot from eSD device, and HIF boot.
- This application note describes an example of initialization when boot from memory connected to the CS0 space (NOR flash) is selected as the boot mode.

Positioning of the Application Note

This application note is based on *SH7730 Example of Initialization* (REJ06B0848), an application note for another MCU that is also built around the SH-4A CPU core. It is recommended that first-time users of a SuperH RISC engine (SH) family product with the SH-4A CPU core and users desiring a brief introduction of relevant fundamental background material refer to *SH7730 Example of Initialization* (REJ06B0848) before consulting the present application note.

Target Device

SH7734 Group (R8A77343)

For a detailed listing of product numbers, see *SH7734 User's Manual: Hardware* (R01UH0233EJ).

In order to apply the contents of this application note to other MCU products, appropriate changes and thorough evaluation are necessary.

Contents

1. Specifications

After release from the reset state, initial settings are made to the clock pulse generator (CPG), local bus state controller (LBSC), memory controller (DBSC3), and cache.

In addition, debugging the sample program code (LEDs, debug serial interface, timer, key matrix interface) using the SH7734 evaluation platform (R0P7734C00000RZ) is described in section 7, Sample Application.

The various application notes for the SH7734 assume use of the sample program code presented in this application note for making initial settings.

Table 1.1 lists the peripheral functions used in the sample application and their applications, and figure 1.1 shows the processing sequence following a power-on reset.

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
CPG	Clock mode setting Note: The setting of the CPG is dependent on the external pin settings. For details, see the Clock Pulse Generator (CPG) section in *SH7734 User's Manual: Hardware* (R01UH0233EJ).
FPU	Floating-point mode setting
LBSC	Area 0: Connected to NOR flash memory (JS28F512M29EWLA (Numonyx) × 1) Data bus width: 16 bits Standard (SRAM) interface selected For details, see the LBSC within Bus Bridge (LBSC) section in *SH7734 User's Manual: Hardware* (R01UH0233EJ).
DBSC3	Areas 2 and 3: Connected to DDR2-SDRAM (MT47H64M16HR-3 (Micron) × 1) Data bus width: 16 bits For details, see the Memory Controller (DBSC3) section in *SH7734 User's Manual: Hardware* (R01UH0233EJ).
Cache	Instruction cache enabled Operand cache enabled
GPIO (debugging applications)	User open LED control
SCIF (debugging applications)	Debug serial interface control Standard I/O by functions such as puts and printf when connected using a terminal software application.
TMU (debugging applications)	Timer control
MTU2, ADC (debugging applications)	Key matrix interface control

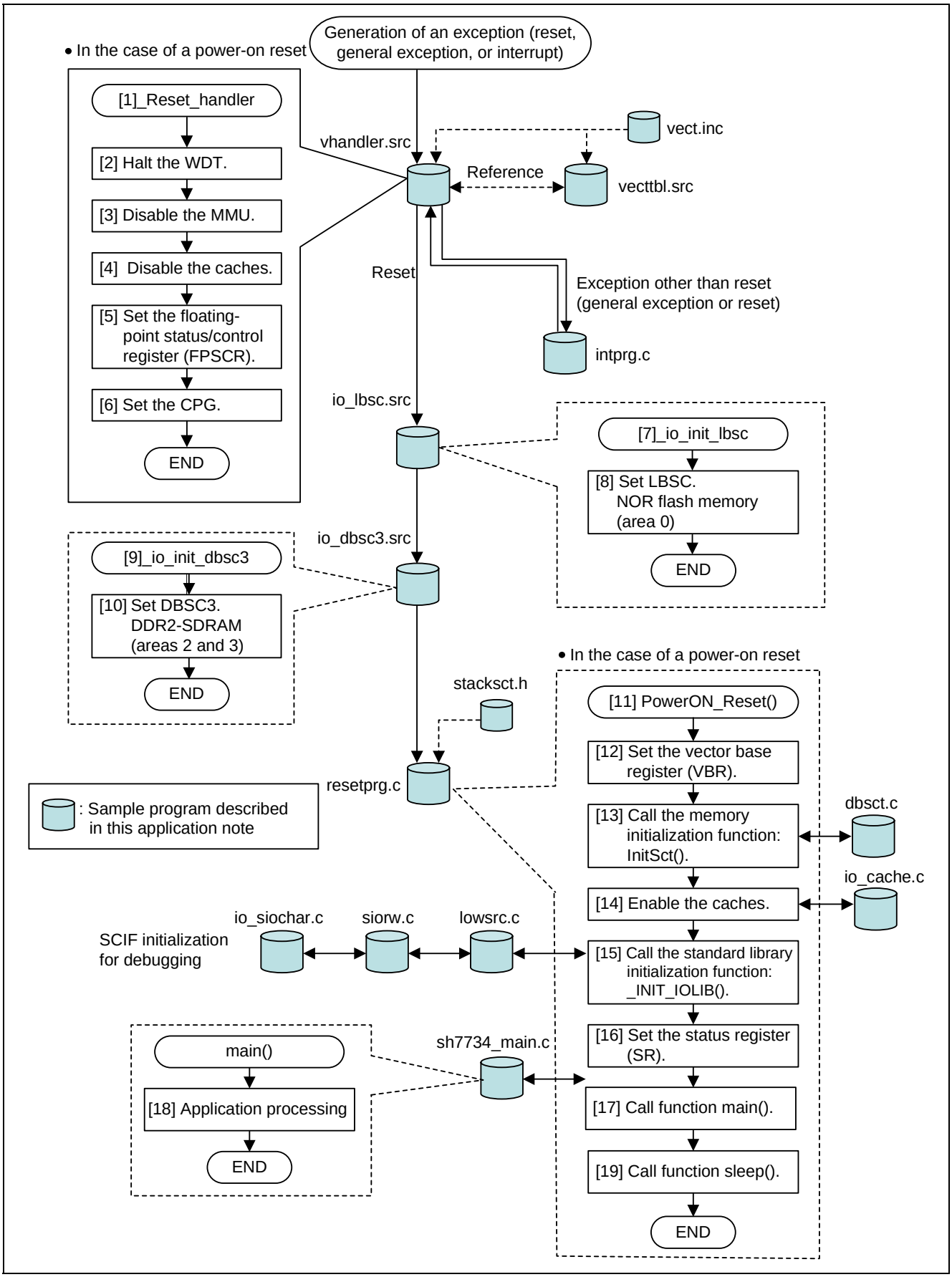


Figure 1.1 Processing Sequence Following a Power-On Reset

2. Conditions Under Which Operation Has Been Confirmed

The sample code presented in this application note has been confirmed to run under the following conditions.

Table 2.1 Conditions Under Which Operation Has Been Confirmed

Item	Description
MCU	SH7734 (R8A77343)
Operating frequencies	EXTAL input frequency: 33.3333 MHz CPU clock (clki): 400 MHz SHwy clock (clks): 200 MHz SHwy clock (clks1): 100 MHz DDR clock (MCK0/MCK0#/MCK1/MCK1#): 200 MHz Bus clock (clkb): 50 MHz Peripheral clock (clkp): 50 MHz
Operating voltage	IO supply power (3.3 V) Core supply power (1.25 V)
Integrated development environment	Renesas Electronics High-performance Embedded Workshop (Ver.4.08.00.011)
C compiler	Renesas Electronics C/C++ Compiler Package for SuperH Family (Ver.9.04 release00)
	Compile options -cpu=sh4a -endian=little -include="$(PROJDIR)\inc" -change_message=warning -object="$(CONFIGDIR)\$(FILELEAF).obj" -debug -optimize=0 -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 -nologo
Sample code version	Ver.1.01
Endian mode	Little endian
Processing mode	Operates in privileged mode only.
Boot mode	CS0 boot mode
Address expansion mode	29-bit
Memory management unit (MMU)	Disabled
Watchdog timer (WDT)	Disabled
Board used	Renesas Electronics SH7734 Evaluation Platform (R0P7734C00000RZ)

3. Related Application Notes

Application notes related to this application note are listed below. Refer to them in conjunction with this application note.

- SH7730 Group Example of Initialization (REJ06B0848)

Note: The operation frequencies differ in some of the above.

4. Description of Peripheral Functions

This application note provides supplementary information about operating modes. For details, see *SH7734 User's Manual: Hardware* (R01UH0233EJ) and *SH7734 Evaluation Platform User's Manual* (R0P7734C00000RZ).

Table 4.1 MD Pin and Pin Function

Pin Name	I/O	Setting	Description
MD0	Input	Free-running mode MD0 = 0	Signal for switching between free-running mode and step-up mode Note: The setting value is fixed on the SH7734 evaluation platform.
MD1 MD2 MD3 MD4	Input	400 MHz, mode (2) MD0 = 0, MD1 = 0, MD2 = 0, MD3 = 0	Operating frequency setting Note: Only fixed settings of 0 for MD4 and MD3 are supported.
MD5 MD6	Input	16-bit bus width MD5 = 0, MD6 = 1	Area 0 bus width setting Note: The setting value is fixed on the SH7734 evaluation platform.
MD7 MD9	Input	MD7 = 0, MD8 = 0*1	Area division setting Note: The setting value is fixed on the SH7734 evaluation platform.
MD8	Input	Little endian MD8 = 1	Big/little-endian setting
MD10	Input	External clock input on EXTAL pin MD10 = 0	Quartz resonator/oscillator setting Note: The setting value is fixed on the SH7734 evaluation platform.
MD11 MD12	Input	×12 MD11 = 0, MD12 = 0	PLL1 multiplication ratio setting
MD13	Input	29-bit addressing mode MD13 = 0	29/32-bit addressing mode setting
MD14 MD16 MD17 MD18 MD19	Input	CS0 boot mode MD14 = 0, MD16 = 0, MD17 = 0, MD18 = 0, MD19 = 0	Boot mode setting*2 Note: The SH7734 evaluation platform supports the CS0 boot, serial boot, and NAND boot modes. Note that for NAND only a connector is provided.
MD15	Input	MD15 = 0	Note: Only MD15 = 0 is supported.

Notes: 1. For information on the areas supported by the LBSC, as viewed from the CPU, and an address map, see the LBSC within Bus Bridge (LBSC) section in *SH7734 User's Manual: Hardware* (R01UH0233EJ).

2. For information on boot mode settings, see the Operating Modes section in *SH7734 User's Manual: Hardware* (R01UH0233EJ).

5. Description of Hardware

5.1 List of Pins used

Table 5.1 lists the pins used and their functions.

Table 5.1 Pins and Functions Used

Module	Pin Name	I/O	Description
LBSC	A[25:0]*[1]	Output	Address output
	D[15:0]	I/O	Bidirectional data bus
	CS0#	Output	Chip select
	CKO	Output	System clock output
	RD#	Output	Read strobe
	RD/WR#	Output	Read/write
	WE1# to WE0#	Output	Write enable
	WAIT#	Input	External wait cycle request input
	BS#	Output	Bus cycle start indication signal
	CS1#/A26	Output	Not used in the sample application
	EX_CS5 to EX_CS0#	Output	Not used in the sample application
	EX_WAIT0 to EX_WAIT2	Input	Not used in the sample application
	DACK0, DACK1	Output	Not used in the sample application
	DREQ0, DREQ1	Input	Not used in the sample application
	DRACK0	Output	Not used in the sample application
DBSC3	MCK0	Output	Clock output
	MCK0#	Output	Inverted clock output of clock output MCK0
	MCKE	Output	CKE output signal
	MCS#	Output	Chip select output signal
	MWE#	Output	Write enable output signal
	MRAS#	Output	Row address strobe output signal
	MCAS#	Output	Column address strobe output signal
	MA[13:0]	Output	Address output signal
	MBA[2:0]	Output	Bank address output signal
	MDQ[15:0]	I/O	Data I/O signal
	MDQS1, MDQS0	I/O	Data strobe I/O signal
	MDQS1#, MDQS0#	I/O	Data strobe I/O signal Inversion of MDQS1 and MDQ0
	MDM1, MDM0	Output	Data mask output signal
	MODT	Output	SDRAM ODT enable output signal
	MZQ	I/O	Calibration pin Note: Connect to VSS via a 120 Ω (precision 1% or greater) resistor.
	MRESET#	Output	DDR3-SDRAM reset output Note: For connection to DDR2-SDRAM, so left open.
	SDBUP	Input	Power supply backup monitor This pin should be fixed low-level when using DDR2-SDRAM.
	SDSELF	Output	DDR self-refresh notification Provides notification of a transition to DDR self-refresh mode. Not used in the sample application.
	MVREFCA	Input	Input reference voltage Connect to GND.

Module	Pin Name	I/O	Description
SCIF3 (debugging applications)	TXD	Output	Data output to terminal software
	RXD	Input	Data input from terminal software
GPIO (debugging applications)	GP1[15]	Output	LED4
	GP1[22]		LED5
AD (debugging applications)	AN0	Input	PS3 to PS5
	AN1		PS6 to PS8
	AN2		PS9 to PS11

Notes: Table 5.1 only lists the pins related to the sample application. Pins related to the clocks, system control, H-UDI, interrupts, and power supplies are not listed. At startup of the SH7734, do not fail to make pin multiplexing settings in the pin multiplexing setting register. Also give careful consideration to the pin pull-up settings in the pin pull-up setting register.

1. After a power-on reset, GP0[25] and GP0[26] are assigned to the default GPIO setting. It is therefore necessary to set them to A24 and A25 before performing downloading by using a flash programming tool (such as fmtool). In the sample application this is accomplished by using an appended .hdc file.

5.2 Reference Circuit

See *SH7734 Evaluation Platform User's Manual* (R0P7734C00000RZ).

6. Description of Software

6.1 Operation Overview

In order to run the main function, which is written in the C programming language, it is first necessary to execute an initial settings program, which performs the minimum essential hardware initialization tasks, such as initialization of the memory, following a power-on reset. This application note presents an example of initialization by using such an initial settings program.

Figures 6.1 and 6.2 illustrate the processing sequence from the power-on reset to the main function.

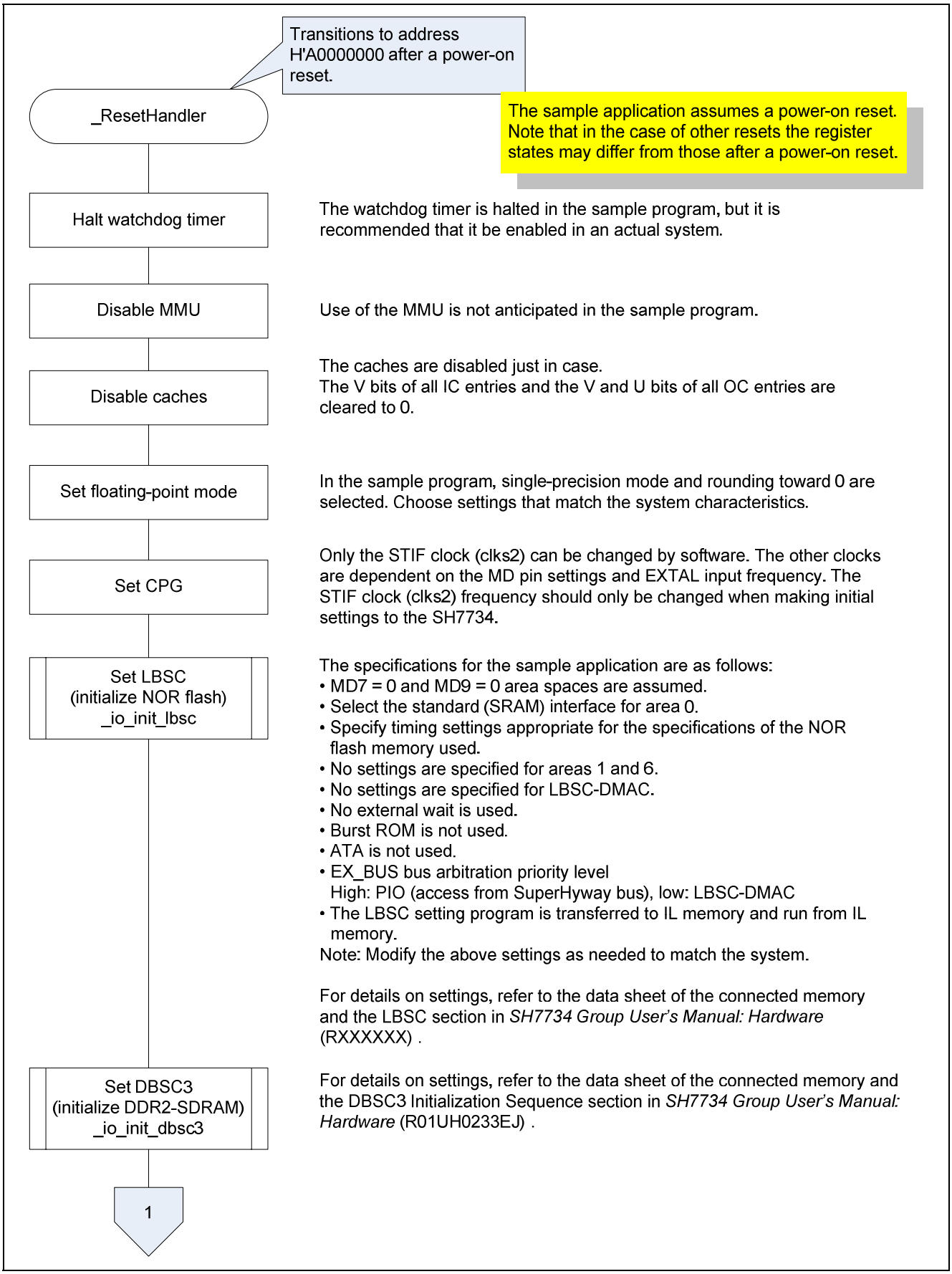
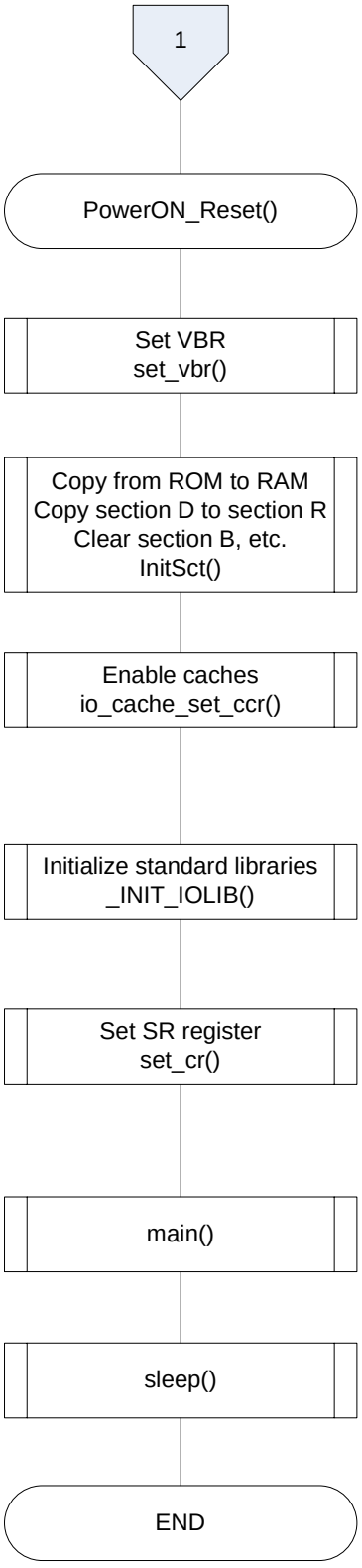


Figure 6.1 Processing Sequence From The Power-On Reset To The Main Function (1)



On the SH-4A CPU core the general exception routine start address is VBR + H'100, so a value of general exception routine start address – H'100 is set in VBR.

In the sample application the contents of section P (program related) are also run after being transferred to RAM.
For details, see the description of section allocation later in this application note.

In the sample application the caches are enabled after data is copied from ROM to RAM (InitSct()). Copying the data from ROM to RAM (**InitSct()**) after enabling the caches would result in faster completion of the copy operation, but it would also be necessary to perform a write-back to memory of the data remaining in the cache.

In order to use functions such as puts or printf to perform standard I/O, it is necessary to first enable standard I/O by using function _INIT_IOLIB. In the sample application SCIF3 is set to standard output. Function _INIT_IOLIB and the low-level interface routine are defined in lowsrc.c.

At this point in time the BL and IMASK bits are cleared to 0. Interrupts are not accepted before this point in time.
Note that the RB bit is set to 1 by an exception or an interrupt. Therefore, the RB should be cleared to 0 during normal operation (when no exception or interrupt has occurred).

This is where processing by the main application occurs.
Application notes to be released in future will cover this processing.

Figure 6.2 Processing Sequence From The Power-On Reset To The Main Function (2)

6.2 File Structure

Table 6.1 lists the files comprising the sample code.

Table 6.1 File Structure

File Name	Description	Remarks
vhandler.src	Defines the exception (reset, general exception, interrupt) handlers.	Refer to the SH7730 Group application note *SH7730 Example of Initialization* (REJ06B0848) for detailed descriptions of these files.
vecttbl.src	Defines the exception (reset, general exception, interrupt) function tables and the interrupt level setting tables used during processing of exception functions.	
resetprg.c	Defines the function written in the C programing language (PowerON_Reset()) that is executed first after a power-on reset.	
dbsct.c	Performs processing such as clearing of section B to 0 and copying of data from section D to section R. Also copies program code from ROM to RAM.	
sh7734_main.c	Defines function main().	
intprg.c	Defines general exception and interrupt functions.	
vect.inc	Contains **.**global definitions for the exception (reset, general exception, interrupt) functions.	
sbrk.c	These files are generated automatically by the integrated development environment.	For details, see the application note *SuperH RISC Engine C/C++ Compiler Package* (REJ05B0463).
sbrk.h		
stacksct.h		
env.inc		
typedefine.h		
io_cache.c	These files contain definitions related to cache control.	For details, see the SH7730 Group application note *Examples of Cache Memory Settings* (REJ06B0851-0100).
io_cache.h		
iodefine.h	Version of iodefine.h for the SH7734.	
io_lbsc.src	Contains initial setting definitions for the NOR flash memory (area 0).	
io_dbsc3.src	Contains initial setting definitions for the DDR2-SDRAM (areas 2 and 3).	
lowsrc.c	These files provide support for standard I/O. They implement function _INIT_IOLIB and the low-level interface routine.	Used for debugging applications. For details, see the application note *SuperH RISC Engine C/C++ Compiler Package* (REJ05B0463).
lowsrc.h		
io_siorw.c		
io_siochar.c		
io_led.c	These files define the sample code related to user open LED control.	Used for debugging applications.
io_led.h		
io_key.c	These files define the sample code related to key matrix control.	Used for debugging applications.
io_key.h		
io_tmu.c	These files define the sample code related to the timer.	Used for debugging applications.
io_tmu.h		
rop7734c00000rz.h	Defines settings that are dependent on the SH7734 evaluation platform.	

6.3 List of Constants

Table 6.2 lists the constants used in the sample code.

Table 6.2 Constants Used in Sample Code

Constant	Setting Value	Description
IMASKclr [vhandler.src]	H'FFFFFF0F	Used when clearing the IMASK bits in the SR register.
RBBLclr [vhandler.src]	H'CFFFFFFF	Used when clearing the RB and BL bits in the SR register.
MDRBBLset [vhandler.src]	H'70000000	Used when setting the MD, RB, and BL bits in the SR register.
SR_Init [vhandler.src]	H'40000000	Sets the SR register state before execution of function main().
INT_OFFSET [vhandler.src]	H'00000100	Used when setting VBR.
D_CACHE_OFF [io_cache.h]	H'00000000	Initializes the CCR register,
D_CACHE_I_INVALID [io_cache.h]	H'00000800	Sets the ICI bit in the CCR register. Clears to 0 the V bits of all IC entries.
D_CACHE_I_ON [io_cache.h]	H'00000100	Sets the ICE bit in the CCR register. Enables the instruction cache.
D_CACHE_O_INVALID [io_cache.h]	H'00000008	Sets the OCI bit in the CCR register. Clears to 0 the V and U bits of all OC entries.
D_CACHE_O_ON [io_cache.h]	H'00000001	Sets the OCE bit in the CCR register. Enables the operand cache.
D_CACHE_IO_ON [io_cache.h]	(CACHE_I_ON | CACHE_O_ON)	Enables the instruction cache and operand cache.
D_CACHE_O_WT [io_cache.h]	H'00000002	Specifies write-through mode.

Items that are related to register addresses, items that are generated automatically by the integrated development environment, and items for debugging applications are omitted from the above list.

6.4 Section Allocation

Table 6.3 summarizes section allocation and table 6.4 the mapping of sections from ROM to RAM.

Table 6.3 Section Allocation

Section Name	Section Application	Area	Allocation Address (Virtual Address)	
P*[2]	Program area (if not otherwise specified)	ROM	H'00003000	Area P0 (Cacheable, MMU address conversion supported)
C	Constant area	ROM		
P$PSEC*[3]	Section initialization program area	ROM		
C$BSEC	Uninitialized data area address structure	ROM		
C$DSEC	Initialized data area address structure	ROM		
D	Initialized data (initial value)	ROM		
B	Uninitialized data area	RAM	H'0C000000	
R	Initialized data area	RAM		
PRAM*[2]	Copy area for program code in ROM (P)	RAM		
S	Stack area	RAM	0x0FFFF9F0	
PINTHandler*[4]	Exception/interrupt handler	ROM	H'80000800	Area P1 (cacheable, MMU address conversion not supported)
VECTTBL	Reset vector table	ROM		
INTTBL*[4]	Interrupt vector table Interrupt mask table	ROM		
PIntPRG*[4]	Interrupt functions	ROM		
SP_S*[1]	TLB miss handler dedicated stack area	RAM	H'8FFFFDF0	
RSTHandler	Reset handler	ROM	H'A0000000	Area P2 (not cacheable, MMU address conversion not supported)
PResetPRG	Reset program	ROM		
P_LBSC_ROM*[5]	Area for program code in ROM (for LBSC)	ROM		
P_DBSC3_ROM*[5]	Area for program code in ROM (for DBSC3)	ROM		
PnonCache*[1]	Program area (accessed when cache disabled)	ROM		
INTTBL_OL*[4]	Interrupt mask table copy area	RAM	H'E500E000	OL memory
PINTHandler_IL*[4]	Exception/interrupt handler copy area	RAM	H'E5200000	IL memory
PIntPRG_IL*[4]	Interrupt functions copy area	RAM		
P_LBSC_IL*[5]	Copy area for program code in ROM (for LBSC)	RAM		

Notes: 1. For information on why special sections are provided, see the SH7730 Group application note *SH7730 Example of Initialization* (REJ06B0848).

2. In the sample application, the program code allocated to section P (in NOR flash) is copied to the PRAM section (in SDRAM) so it can be run from there.
3. When using a standard library function (_INITSCT()) to copy the section, as described in note 2, it is necessary to consider the following:
   Function _INITSCT() is allocated to section P, and calling _INITSCT() causes the MCU to attempt to run it not from section P (ROM), but from the PRAM section (RAM), since this section is mapped from ROM to RAM. However, _INITSCT() has itself not yet been copied to the PRAM section (RAM) when it is called for the purpose of copying the contents of section P, so it cannot be executed.

As a workaround, the sample application creates a separate function (InitSct()) that does the same job as _INITSCT(). This function is allocated to the P$PSEC section (ROM) and is executed from there.

4. In the sample application, the program code allocated to the exception/interrupt functions and interrupt functions is run from the IL memory to ensure faster execution. In addition, the table data (INTTBL) referenced during interrupt processing is referenced from the OL memory.
5. It is not recommended that the settings (in LBSC, DBSC3, etc.) for an area be changed while the area is in use for program execution. (In particular, care should be exercised when making burst ROM settings.) Therefore, in the sample application, area 0 is accessed again only after program code in portions of the IL memory other than area 0 have changed the setting of area 0 (after the LBSC setting change). As for DBSC3 settings, they are performed by program code in area 0 and therefore left unchanged.

Table 6.4 Mapping of Sections from ROM to RAM

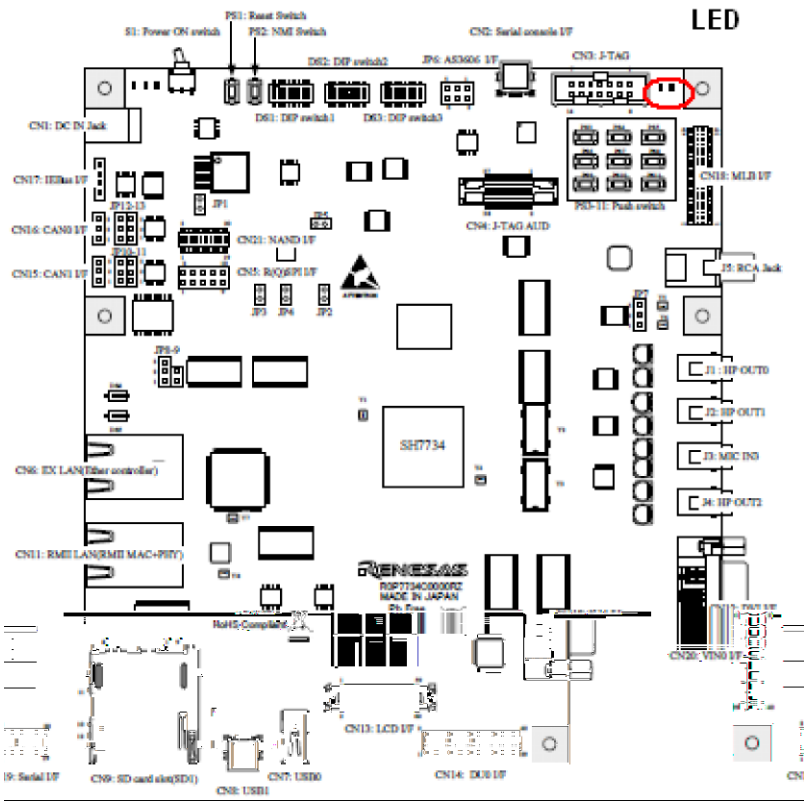
ROM	Memory	RAM	Memory
P	NOR flash	PRAM	SDRAM
D	NOR flash	R	SDRAM
PINTHandler	NOR flash	PINTHandler_IL	IL memory
PIntPRG	NOR flash	PIntPRG_IL	IL memory
P_LBSC_ROM	NOR flash	P_LBSC_IL	IL memory
INTTBL	NOR flash	INTTBL_OL	OL memory

7. Sample Application

A sample application for debugging using the SH7734 evaluation platform (R0P7734C00000RZ) is described below.

7.1 User Open LED Control

The program code for controlling the user open LEDs (LED4 and LED5) is described below.



SH7734 Evaluation Platform (R0P7734C00000RZ)

Figure 7.1 User Open LEDs

7.1.1 List of Functions

Table 7.1 lists the user open LED control functions. It is assumed that they will be called from main().

Table 7.1 User Open LED Control Functions

Function	Description
io_led_init	Initializes GP1[15] (LED4) and GP1[22] (LED5) port output.
io_led_on	Turns on the target LED.
io_led_off	Turns off the target LED.

7.1.2 Function Specifications

The specifications of the functions included in the sample code are listed below.

io_led_init	
Overview	Initializes GP1[15] (LED4) and GP1[22] (LED5) port output.
Header	io_led.h
Declaration	void io_led_init(void)
Description	Initializes GP1[15] (LED4) and GP1[22] (LED5) port output.
Arguments	None
Return value	None
Notes	

io_led_on		
Overview	Turns on the target LED.	
Header	io_led.h	
Declaration	void io_led_on(E_ID_USER_LED ledno)	
Description	Turns on the LED specified by the argument. It is also possible to turn on LED4 and LED5 at the same time.	
Arguments	First argument: ledno	D_ID_USER_LED4: Turn on LED4. D_ID_USER_LED5: Turn on LED5. D_ID_USER_LED_ALL: Turn on LED4 and LED5.
Return value	None	
Notes		

io_led_off		
Overview	Turns off the target LED.	
Header	io_led.h	
Declaration	void io_led_off(E_ID_USER_LED ledno)	
Description	Turns off the LED specified by the argument. It is also possible to turn off LED4 and LED5 at the same time.	
Arguments	First argument: ledno	D_ID_USER_LED4: Turn off LED4. D_ID_USER_LED5: Turn off LED5. D_ID_USER_LED_ALL: Turn off LED4 and LED5.
Return value	None	
Notes		

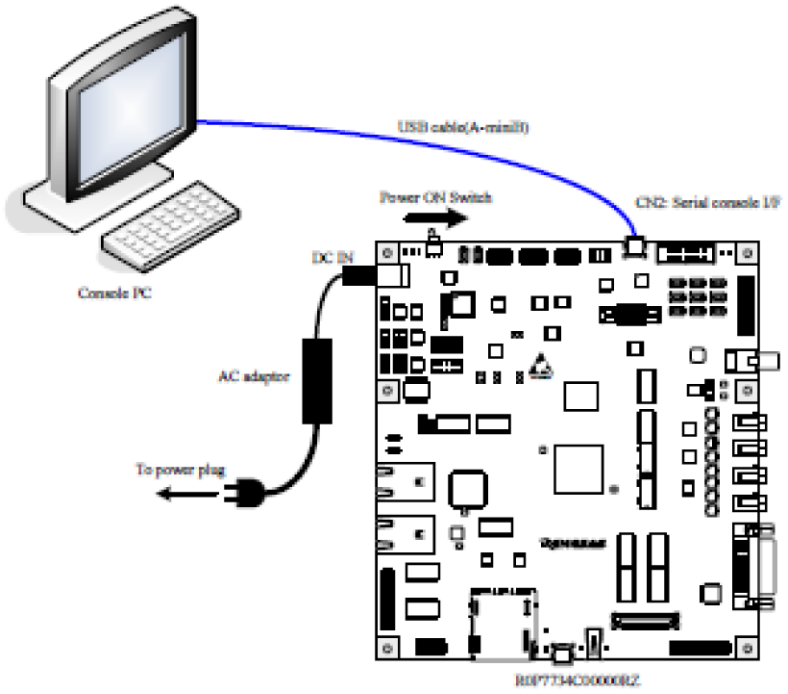
7.1.3 Usage Example

```
void main(void)
{

  /* Initialization */
  io_led_init();

  io_led_on(D_ID_USER_LED4);      /* Turn on LED4. */
  io_led_on(D_ID_USER_LED5);      /* Turn on LED5. */

  io_led_off(D_ID_USER_LED4);     /* Turn off LED4. */
  io_led_off(D_ID_USER_LED5);     /* Turn off LED5. */

  io_led_on(D_ID_USER_LED_ALL);   /* Turn on LED4 and LED5. */
  io_led_off(D_ID_USER_LED_ALL);  /* Turn off LED4 and LED5. */

}
```

7.2 Debug Serial Interface (SCIF3)

The program code for controlling the debug serial interface (SCIF3) is described below.

Input and output of debugging information using a terminal program running on a PC is supported.



SH7734 Evaluation Platform (R0P7734C00000RZ)

Figure 7.2 Debug Serial Interface (SCIF3)

7.2.1 Standard I/O

Standard I/O functions such as puts, get, and printf can be used. SCIF3 is set to standard I/O. Function _INIT_IOLIB and the low-level interface routine are defined in lowsrc.c.

Note: The sample program code does not implement exclusive control. It is not assumed that its functions will be called from an interrupt functions.

7.2.2 Terminal Program Settings

Table 7.2 lists the terminal program settings.

Table 7.2 Terminal Program Settings

Item	Setting Value
Bit rate	115,200 bps
Data length	8 bits
Stop bit length	1 bit
Parity	None

7.2.3 Usage Example

```
void main(void)
{

  puts( "" );
  puts("\nSH7734 Sample Program. Ver.1.00.00");
  puts("Copyright (C) 2011 Renesas Electronics Corp. All Rights Reserved");
  puts("and Renesas Solutions Corp. All Rights Reserved");
  puts( "" );
  printf("test sample\n");
  fflush(stdout);

}
```

7.3 Timer Control

The program code for controlling the timer (TMU0) is described below. When a timeout occurs, an interrupt is generated and the registered callback function is called.

7.3.1 List of Functions

Table 7.3 lists the timer control functions.

Table 7.3 Timer Control Functions

Function Name	Description
io_tmu0_create	Initializes TMU0.
io_tmu0_start	Starts the timer.
io_tmu0_stop	Stops the timer.
io_tmu0_compare_match	Called from the interrupt functions when a timeout occurs.

7.3.2 Function Specifications

The specifications of the functions included in the sample code are listed below.

io_tmu0_create	
Overview	Initializes TMU0.
Header	io_tmu.h
Declaration	void io_tmu0_create (T_TMU_TimerTYPE *i_pTimerType)
Description	Enables interrupts. Sets the priority of the interrupt specified by the argument.
Arguments	First argument: i_pTimerType Timer type structure address
Return value	None
Notes	

io_tmu0_start	
Overview	Starts the timer.
Header	io_tmu.h
Declaration	void tmu0_start(T_TMU_SETTCOR_INFO *i_pTCORInfo, TMU_TIMEOUT_CALLBACK i_func)
Description	Sets the timer duration specified by the argument in the timer counter. Registers the callback function specified by the argument. Starts the timer.
Arguments	First argument: i_pTCORInfo TCOR setting information structure address First argument: i_func Callback function at timeout
Return value	None
Notes	Timer settings are in ms units, and setting values from 1 ms to 1000 ms are supported. A timer value outside the above range is treated as a setting of 1 ms.

io_tmu0_stop	
Overview	Stops the timer.
Header	io_tmu.h
Declaration	void tmu0_stop(void)
Description	Stops the timer.
Arguments	None
Return value	None
Notes	

io_tmu0_compare_match	
Overview	Called from the interrupt functions when a timeout occurs.
Header	io_tmu.h
Declaration	void tmu0_compare_match (void)
Description	Clears the timer interrupt source. The callback function registered by io_tmu0_start() is called.
Arguments	None
Return value	None
Notes	

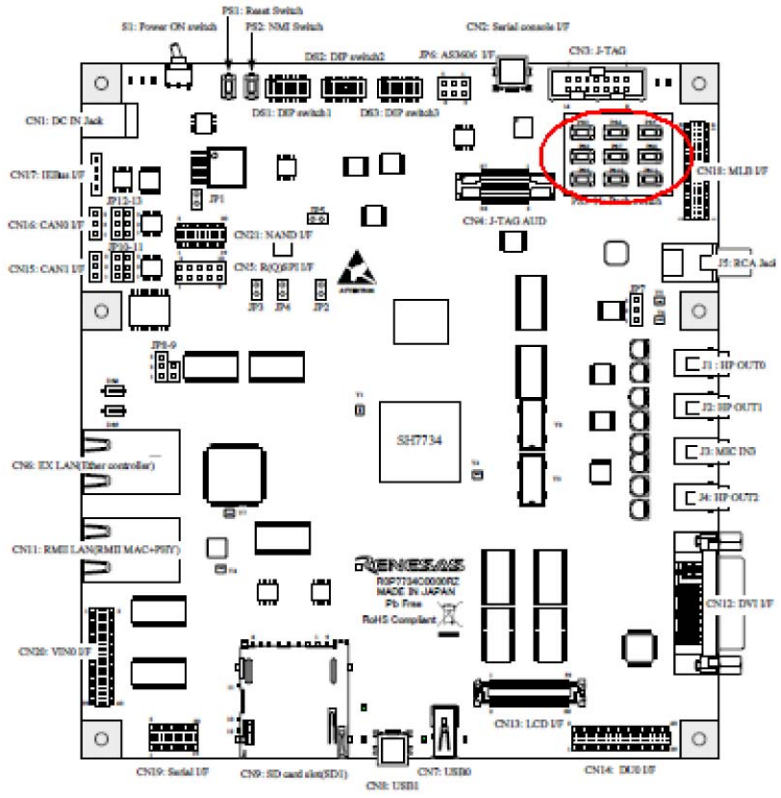
7.3.3 Usage Example

This sample program code causes an LED to flash at 500 ms intervals (by calling **led_onoff()**).

```
/*Main function */
void main(void)
{

  T_TMU_TimerTYPE          TimerType;
  T_TMU_SETTCOR_INFO       SetTCORInfo;

  memset(&TimerType, 0x00, sizeof(TimerType));
  memset(&SetTCORInfo, 0x00, sizeof(SetTCORInfo));

  io_led_init(); /* LED initialization */

  /* Timer create structure settings */
  TimerType.mClockSelect = D_TMU_DIV_64;      /* clkp /64 */
  TimerType.mIntcPri = D_TMU_PRI_1;           /* Interrupt level 1 */

  /* TMU0 timer create processing */
  io_tmu0_create(&TimerType);

  /* Timer duration setting */
  SetTCORInfo.mTimeValue = 500;               /* Timer setting value (500 ms)
*/

  /* TMU0 timer start processing */
  io_tmu0_start(&SetTCORInfo, led_onoff);

}

/* Interrupt function */
void INT_TMU00(void)
{
  io_tmu0_compare_match();
}

/* Callback function */
void led_onoff(void)
{
  if(g_onoff == 0)
  {
   io_led_on(D_ID_USER_LED4);
   g_onoff = 1;
  }
  else
  {
   io_led_off(D_ID_USER_LED4);
   g_onoff = 0;
  }

}
```

7.4 Key Matrix Interface Control

The program code for controlling the key matrix interface is described below.



SH7734 Evaluation Platform (R0P7734C00000RZ)

Figure 7.3 Key Matrix Interface

7.4.1 List of Functions

Table 7.4 lists the key matrix interface control functions. It is assumed that they will be called from main().

The states of AN0 to AN2 change according to the keys that are pressed, as described in the Key Matrix Interface (ADC) section in *SH7734 Evaluation Platform User's Manual* (R0P7734C00000RZ).

The ADC (AN0 to AN2) state is continually A/D converted at the conversion triggers (10 ms intervals) from MTU2.

In the interrupt processing when A/D conversion ends, the key that was depressed is determined from the converted A/D data, and the corresponding callback function is called.

If multiple keys are pressed at the same time, the sample application uses a priority ranking of PS9 to PS11 > PS6 to PS8 > PS3 to PS5 to recognize the key press.

Table 7.4 Key Matrix Interface Control Functions

Function	Description
io_key_init	Initializes MTU2 and the ADC. Registers callback functions corresponding to the keys.
io_key_start	Starts counting by MTU2. Starts A/D conversion using conversion triggers (at 10 ms intervals) from MTU2.
io_key_stop	Stops counting by MTU2.
io_key_int	Called from the interrupt functions when A/D conversion ends.

7.4.2 Function Specifications

The specifications of the functions included in the sample code are listed below.

io_key_init	
Overview	Initializes MTU2 and the ADC. Registers callback functions corresponding to the keys.
Header	io_key.h
Declaration	void io_key_init(T_KEY_MTX *key_mtx)
Description	Initializes MTU2. Initializes the ADC. Registers callback functions corresponding to the keys. Calls the registered function when a key is pressed.
Arguments	First argument: *key_mtx Key matrix information structure
Return value	None
Notes	

io_key_start	
Overview	Starts counting by MTU2. Starts A/D conversion using conversion triggers (at 10 ms intervals) from MTU2.
Header	io_key.h
Declaration	void io_key_start (void)
Description	Starts counting by MTU2 (10 ms). When a MTU2 counter overflow occurs (after 10 ms have elapsed), starts A/D conversion.
Arguments	None
Return value	None
Notes	

io_key_stop	
Overview	Stops counting by MTU2 and A/D conversion.
Header	io_key.h
Declaration	void io_key_stop (void)
Description	Stops MTU2 counting. Stops A/D conversion.
Arguments	None
Return value	None
Notes	

io_key_int	
Overview	Called from the interrupt functions when A/D conversion ends.
Header	io_key.h
Declaration	void io_key_int(void)
Description	The key that was depressed is determined from the A/D converted data from AN0 to AN2. (For ADC input specifications, see *SH7734 Evaluation Platform User's Manual* (R0P7734C00000RZ).) Clears the interrupt source flag. The registered callback function is called.
Arguments	None
Return value	None
Notes	

7.4.3 Usage Example

In this example, LED4 lights when key 0 is pressed and LED4 extinguishes when key 8 is pressed. No processing is performed when other keys are pressed.

```
/* Processing when key 0 is pressed */
void Key_0(void)
{
  io_led_on(D_ID_USER_LED4);
}

/* Processing when key 8 is pressed */
void Key_8(void)
{
  io_led_off(D_ID_USER_LED4);
}

/* Definition of callback functions called when keys are pressed */
static T_KEY_MTX t_key_tbl[E_KEY_FLG_MAX] =
{
  Key_0,
  NULL,  /* None registered */
  NULL,  /* None registered */
  NULL,  /* None registered */
  NULL,  /* None registered */
  NULL,  /* None registered */
  NULL,  /* None registered */
  NULL,  /* None registered */
  Key_8,
}

/* Main function */
void main(void)
{
  /* LED initialization processing */
  io_led_init();

  /* Key matrix initialization processing */
  io_key_init(t_key_tbl);

  /* Key matrix start processing */
  io_key_start();

}

/* Interrupt function */
void INT_ADC (void)
{
  /* Key matrix interrupt functions */
  io_key_int();
}
```

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jun.22.12	—	First edition issued
1.01	Jul.18.12	—	Sample code corrected (iodefine.h, io_tmu.c) and and sample code version revised

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 manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord 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 one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

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.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

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: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

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

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, 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 Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 2.0