To our customers,

# Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

# Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. 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.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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.
5. When exporting the 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. You should not use Renesas Electronics products or the 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. 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.
6. 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.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. 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 categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. 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 an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “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.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. 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.
9. 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 system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. 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.
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.

# SH7730 Group

## SH7730 Example of Initialization

### Introduction

This application note describes an example of items that must be set when starting up the SH7730 MCU.

### Target Device

SH7730

### Contents

## Positioning of the Application Note

This application note describes the procedure for SH7730 initialization. It is intended to be the first application note for reference regarding the SH7730. The descriptions include brief introductions of relevant fundamental background material to take first-time users of a Super H RISC engine Family product with the SH-4A CPU core into account.

The structure of this application note is as follows.

- Section 1 gives specifications and applicable conditions for the sample program of the application note.
- Section 2 summarizes the development environment, introduces fundamental background material on initialization, and supplementary information regarding Super H Family MCUs. This section is the minimum required reading of this document for users. For those who already have fundamental background knowledge of Super H Family products and initialization, please skip this section.
- Section 3 describes the actual processing involved in making initial settings, including the conduct of initialization and points to keep in mind.
- Section 4 includes sample programs for the processing described in section 3.
- Section 5 gives a list of documents for reference.

## SH7730-Related Application Notes

Refer to the following application notes in combination with this one. These documents include descriptions of the individual settings for particular modules and functions.

- SH7730 Group Application Note: *Example of BSC SDRAM Interface Connection (32-Bit Data Bus)* (REJ06B0850): Describes initial settings of the BSC for use with external SDRAM.
- SH7730 Group Application Note: *Example of BSC Interface Connection to NOR-Type Flash Memory* (REJ06B0849): Describes initial settings of the BSC for use with external memory.
- SH7730 Group Application Note: *Examples of Cache Memory Settings* (REJ06B0851): Describes initial settings to enable the instruction/operand cache.
- SH7730 Group Application Note: *Example of Writing Back from the Operand Cache* (REJ06B0853): Describes writing back data from the operand cache to memory.

# 1. Preface

## 1.1 Specifications

The clock pulse generator (CPG), bus state controller (BSC), and cache are initialized after release from the reset state.

## 1.2 Modules Used

- Clock pulse generator (CPG)
- Bus state controller (BSC)
- Cache

## 1.3 Applicable Conditions

- Evaluation board: The AP-SH4A-1A board incorporates the SH7730 with SH-4A CPU core and is available from AlphaProject Co., Ltd.
  External memory (area 0) 4-MB NOR-type flash memory: S29AL032D70TF104 from Spansion
  (area 3) 32-MB SDR-SDRAM (16 MB × 2): K4S281632F-UC75 from Samsung
- MCU: SH7730 (R8A77301)
- Operating frequency: Internal clock: 266.66 MHz
  SuperHyway bus clock: 133.33 MHz
  Bus clock: 66.66 MHz
  Peripheral clock: 33.33 MHz
- Bus width for area 0: 16-bit fixed (with the MD3 pin at the low level)
- Clock operating mode: Mode 2 (with the MD0 pin at the low level, and MD1 pin at the high level)
- Endian: Big endian (with the MD5 pin at the low level)
- Toolchain: SuperH RISC engine Standard Toolchain Ver.9.1.1.0 from Renesas Technology
- Compiler options: Default settings of High-performance Embedded Workshop
  -cpu=sh4a -include="$(PROJDIR)\inc" -object="$(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

## 2. Essential Items for Setting

This section gives fundamental required background knowledge, things to do in general, and points to keep in mind regarding SH7730 initialization.

### 2.1 Fundamental Background

Before using the sample program, make sure you obtain the manuals listed in section 5, and understand the development environment and the SH7730 CPU. Points for reference in the manuals are indicated below.

#### 2.1.1 Development Environment

- How to set up the High-performance Embedded Workshop
  In this document, the High-performance Embedded Workshop is assumed to serve as the development environment. See the document: *SuperH RISC engine C/C++ Compiler Package Application Note: [Introduction Guide] Sample File Guide for SH-3, SH-4, and SH-4A* (REJ06J0012) for information on how to set it up. Also consult the Help function in the High-performance Embedded Workshop menu bar for information on usage.
- Downloading to flash memory
  In this sample program, the flash memory downloading function of the E10A-USB emulator is used to download the user's programs to an external flash memory area. See the Application Note *Flash Memory Download Program for the E10A-USB Emulator* (REJ10J1221) for information on using the emulator for this purpose.

#### 2.1.2 SH7730 CPU

- Sections
  See the section on programming in the User's Manual: *SuperH™RISC engine C/C++ Compiler, Assembler, Optimizing Linkage Editor Compiler Package V.9.01* (REJ10J1571).
- Register descriptions
  See the section on register descriptions in the *SH7730 Group Hardware Manual* (REJ09B0359).
- Instruction set
  See the section on instruction set in the *SH7730 Group Hardware Manual* (REJ09B0359) and the User's Manual: *SuperH™RISC engine C/C++ Compiler, Assembler, Optimizing Linkage Editor Compiler Package V.9.01* (REJ10J1571).
- Exception handling
  See the section on exception handling in the *SH7730 Group Hardware Manual* (REJ09B0359).
- Virtual addresses, areas P0 to P4, physical addresses
  See the section on the memory management unit (MMU) in the *SH7730 Group Hardware Manual* (REJ09B0359).
- Division into areas (0 to 7), shadow areas, address map
  See the section on the bus state controller (BSC) in the *SH7730 Group Hardware Manual* (REJ09B0359).
- Cache
  See the following SH7730 Group Application Notes: *Examples of Cache Memory Settings* (REJ06B0851) and *Example of Writing Back from the Operand Cache* (REJ06B0853).
- BSC setting
  See the following SH7730 Group Application Notes: *Example of BSC SDRAM Interface Connection (32-Bit Data Bus)* (REJ06B0850) and *Example of BSC Interface Connection to NOR-Type Flash Memory* (REJ06B0849).

## 2.2 Preparing the Development Environment

### 2.2.1 Preparing the Evaluation Board for the SH7730

In this document, the evaluation board is assumed to be the AP-SH4-1A board produced by AlphaProject Co., Ltd. As the program is written to external flash memory on the evaluation board, consult the Application Note *Flash Memory Download Program for the E10A-USB Emulator* (REJ10J1221) and make the E10A-USB emulator connectable.

### 2.2.2 Preparing the for Environment: High-performance Embedded Workshop

A new project for the High-performance Embedded Workshop is launched and the following settings are made.

| | |
|---|---|
| • Project name: | sh7730 (any project name is acceptable)) |
| • CPU: | SuperH RISC engine |
| • CPU series: | SH-4A |
| • CPU type: | SH7730 |
| • Stack pointer address: | Values of the stack area in table 1 *Allocation of Sections* are set. |
| • Target: | SessionSH-4A_E10A-USB_SYSTEM is ticked. |

If a new project is launched with the above settings having been made, the following files are automatically generated. For the contents of these automatically generated files and further details, see *SuperH RISC Engine C/C++ Compiler Package Application Note: [Introduction Guide] Sample File Guide for SH-3, Sh-4, and SH-4A* (REJ06J0012).

- sh7730.c
- dbsct.c
- resetprg.c
- sbrk.c
- iodefine.h
- sbrk.h
- stacksct.h
- typedefine.h
- env.inc
- vect.inc
- intprg.src
- vecttbl.src
- vhandler.src

## 2.3 Points for Setting and Caution

### 2.3.1 Allocation of Sections

Make settings so that sections are allocated at the addresses given in table 1. When debugging proceeds during development, data (programs) which would usually be written to the ROM area are written to the RAM area instead. Carefully consider the sizes of sections and whether caching is enabled or disabled for the areas where sections are allocated.

Basically, the settings listed in table 1 are automatically made by the High-performance Embedded Workshop. In cases where other sections need to be added, please take it into consideration that individual settings are separately made (sections indicated by *1 and *2 are newly added).

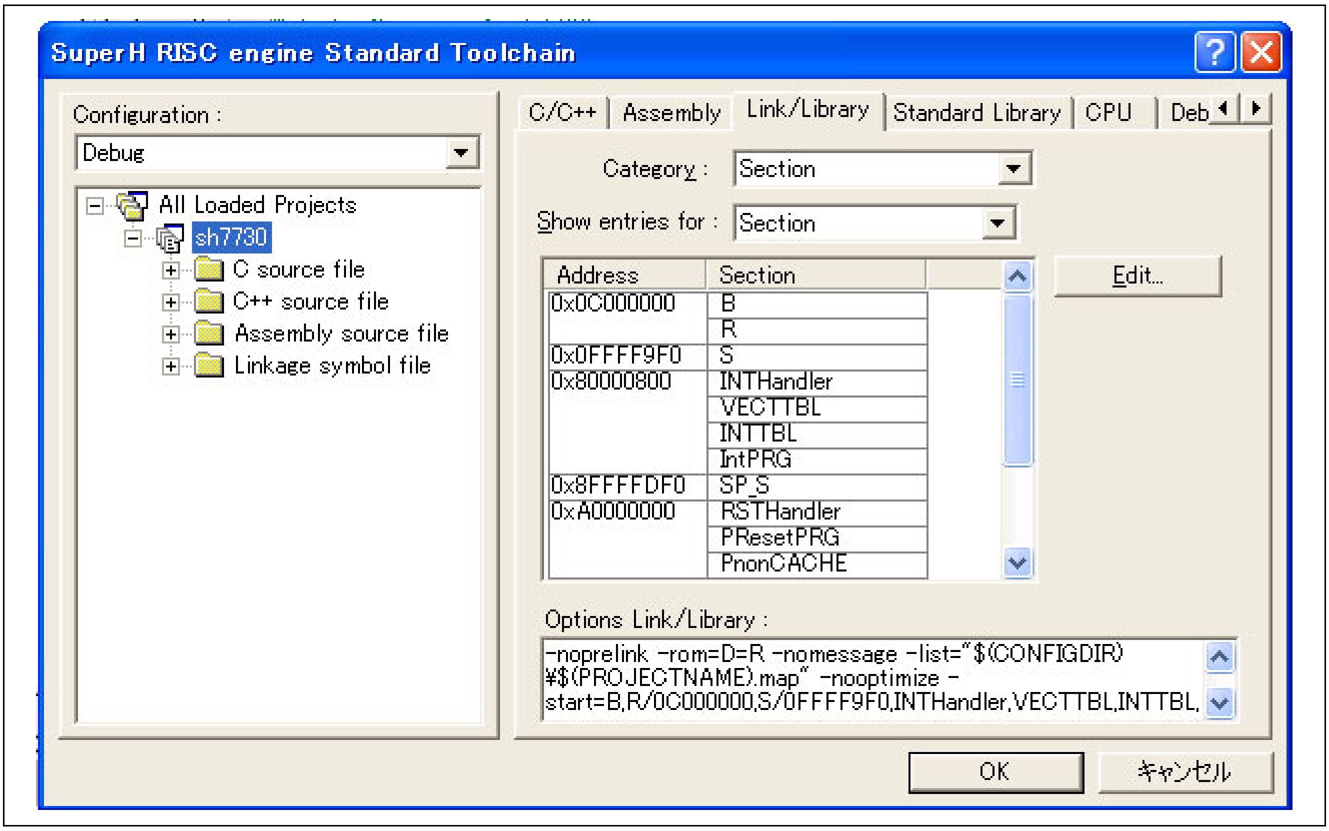
**Table 1 Allocation of Sections**

| Section Name | Application of Section | Area | Allocation Address (Virtual Address) | |
|---|---|---|---|---|
| P | Program area (in the case of none specified) | ROM | 0x00003000 | Area P0<br>(caching is enabled, MMU addresses can be translated) |
| C | Constant area | ROM | | |
| C$BSEC | Address structure for non-initialized data area | ROM | | |
| C$DSEC | Address structure for initialized data area | ROM | | |
| D | Initialized data (initial value) | ROM | | |
| B | Non-initialized data area | RAM | 0x0C000000 | |
| R | Initialized data area | RAM | | |
| S | Stack area | RAM | 0x0DFFF9F0 | |
| INTHandler | Exception/interrupt handler | ROM | 0x80000800 | Area P1<br>(caching is enabled, MMU addresses cannot be translated) |
| VECTTBL | Reset vector table<br>Interrupt vector table | ROM | | |
| INTTBL | Interrupt mask table | ROM | | |
| PIntPRG | Interrupt function | ROM | | |
| SP_S*[1] | Stack area for handler of TLB misses | RAM | 0x8DFFFDF0 | |
| RSTHandler | Reset handler | ROM | 0xA0000000 | Area P2<br>(caching is disabled, MMU addresses cannot be translated) |
| PResetPRG | Reset program | ROM | | |
| PnonCache*[2] | Program area (non-cacheable access) | ROM | | |

[Reference] How to set sections

The following procedure makes the window in the figure below appear.

1. Select **Build (B)** in the High-performance Embedded Workshop menu bar.
2. Select **SuperH RISC engine Standard Toolchain**.
3. Select **Link/Library**.
4. Select “Section” for **Category (Y).**



Category : Section

Show entries for : Section

| Address | Section |
|---|---|
| 0x0C000000 | B |
| | R |
| 0x0FFFF9F0 | S |
| 0x80000800 | INTHandler |
| | VECTTBL |
| | INTTBL |
| | IntPRG |
| 0x8FFFFDF0 | SP_S |
| 0xA0000000 | RSTHandler |
| | PResetPRG |
| | PnonCACHE |

Options Link/Library :

```
-noprelink -rom=D=R -nomessage -list="$(CONFIGDIR)
¥$(PROJECTNAME).map" -nooptimize -
start=B,R/0C000000,S/0FFFF9F0,INTHandler,VECTTBL,INTTBL,
```

**Figure 1 Section Setting Window**

### 2.3.2 Stack Settings

A stack area is required to run a program; specify the stack size and stack-pointer address. The High-performance Embedded Workshop automatically sets these to the values which have been set when the project was launched. To change the size and address of the stack area, select: **Project (P)** in the High-performance Embedded Workshop menu bar → **Edit Project Configuration (E)** → the **Stack** tab.

Furthermore when the memory management unit (MMU) is used, the stack area for the handler of TLB misses (SP_S*[1]) as given in table 1 needs to be taken into consideration. See 3.2.1. vhandler.src in section 3.2 Description of the Sample Program. For details on the MMU, see the section on the MMU in the *SH7730 Group Hardware Manual* (REJ09B0359).

### 2.3.3 Setting of the Watchdog Timer (WDT)

In the initial state, counting by the watchdog timer starts. When the counter overflows, an internal reset occurs. To activate a system, halting of the WDT or regular clearing of its counter is required.

### 2.3.4 Setting of the Floating-Point Status/Control Register (FPSCR)

This register is used to specify whether floating-point instructions are executed as single-precision operations or double-precision operations. Settings should be made in accord with the system design. The initial setting is for single-precision mode.

### 2.3.5 Setting of the Bus State Controller (BSC)

Settings for the BSC should be in accord with the timing specifications for reading and writing of external memory. The stack area can be used when a function written in the C language is called. Accordingly, if the stack area is allocated in external memory such as SDRAM, the BSC must be initialized in advance of program execution. In the sample program, the code that handles processing for BSC initialization is in the exception handler (vhandler.src). For details, see 3.2.1. vhandler.src in section 3.2, Description of the Sample Program.

### 2.3.6 Setting of the Vector Base Register (VBR)

The reset vector address is fixed at H'A000 0000. Start addresses for general exceptions and interrupts other than the reset are determined by adding an offset (H'400 for TLB miss exceptions, H'100 for other exceptions in general, and H'600 for interrupts) for the specific event to the vector base address.
In the sample program, the start address of the general exception handler (_INTHandlerPRG) is exported by the exception handler (vhandler.src) and then used to set the VBR in the PowerON_Reset() function, the first to be called in the reset processing program (resetprg.c).
In the exception handler (vhandler.src), the start addresses of the TLB miss handler (_TLBmissHandler) and interrupt handler (_IRQ_Handler) are defined by “.org H'300” and “.org H'500”, respectively based on the offset (H'100) for other exceptions in general.

### 2.3.7 Memory Initialization (_INITSCT)

Although global variables with initial values are placed in the ROM area (section D) when the system is activated, they must be copied to the RAM area (section R) so that they can be handled as variables. Global variables without initial values are placed in the RAM area (section B) and must be initialized at the time of system activation. The High-performance Embedded Workshop automatically handles these processes. For details, see 3.2.5 dbsct.c in section 3.2, Description of the Sample Program.

### 2.3.8 Cache Settings

In the initial settings, please consider which of the cacheable areas (P0, P1, P3) are to be placed in the cache-enabled state, and the write mode (write-through or copy-back) in the cache-enabled state. For details, see the following SH7730 Group Application Notes: *Examples of Cache Memory Settings* (REJ06B0851) and *Example of Writing Back from the Operand Cache* (REJ06B0853).

### 2.3.9 Setting of the Status Register (SR)

The SR is used to select privileged mode or user mode, specify general register banks, and control exceptions and interrupts. Settings should be made in accord with the system design. In the sample program, the following settings are made.

- Privileged mode
- Selection of general register bank 0
- Release of exception/interrupt blocking (changing value of block bits)

## 3. Description of Sample Application

Based on the previous sections, this section describes the actual creation of an environment for SH7730 initialization with corrections and additions to the source code that is automatically generated by the High-performance Embedded Workshop.

Use of the sample program described in this document as a program for initialization is a precondition for using the sample code of the other application notes of the SH7730.

### 3.1 Changes to the Environment Automatically Generated by the High-performance Embedded Workshop

This sample program changes, adds to, and deletes from the environment that has been automatically generated by the High-performance Embedded Workshop in the following ways.

- Allocation of sections
  Changes and additions need to be made as given in table 1.
- Processing to enable the caches
  Function PowerON_Reset () (see 7 in figure 2) is used to enable the caches.
- Timing of initialization of the floating-point status/control register (FPSCR)
  Changes are made so that the values which have been set in the PowerON_Reset () function are set in the reset handler (see 4 in figure 2).
- Setting of the clock pulse generator (CPG)
  Reset handler (see 5 in figure 2) is used to set the CPG so that the applicable conditions given in section 1.3, Applicable Conditions, are in effect.
- Setting of the bus state controller (BSC)
  Processing for BSC initialization so that external memory (flash memory, SDRAM) can be used is added to the reset handler (see 6 in figure 2).
- Setting of the On-Chip Memory Control Register (RAMCR)
  In the code that is automatically generated by the High-performance Embedded Workshop, the RMD (on-chip memory access mode) bit in the on-chip memory control register (RAMCR) is set to 1 before return from the PowerON_Reset () function, enabling access to on-chip memory in user mode. This setting is skipped in the sample program because operation in privilege mode is a precondition for the rest of the processing.
- intprg.src
  The language for functions of general exceptions and interrupts is changed from assembler to C.

Note: The sample program sets up an environment where the MMU is not in use. If it is to be used, take the following points into consideration.
Addition of the TLB miss handler
Additional setting of a dedicated stack area for cases where a TLB miss occurs

## 3.2 Description of the Sample Program

The initialization program consists of the following eight source files.

1. vhandler.src
2. vecttbl.src
3. resetprg.c
4. stacksct.h
5. dbsct.c
6. sh7730.c
7. intprg.c
8. vect.inc

### 3.2.1 vhandler.src

When an exception (reset, general exception, or interrupt) occurs, the code in the exception handler (vhandler.src) is first to be executed. File vhandler.src contains the code for processing by the handlers for all exceptions and the processing for BSC initialization. Processing in handlers for the reset and for exceptions other than reset is different; for details, see *SuperH RISC engine C/C++ Compiler Package Application Note: [Introduction Guide] Sample File Guide for SH-3, SH-4, and SH-4A* (REJ06J0012).

The reset handler (from label _Reset_handler) is activated by a power-on reset. The reset handler used in this application program differs from that generated by the High-performance Embedded Workshop in the following ways: the instruction cache and operand cache are disabled (see 3 in figure 2), the FPSCR is set (see 4 in figure 2), the CPG is set (see 5 in figure 2), and the BSC is initialized (see 6 in figure 2). The TLB miss handler has also been changed for the reasons described below.

The initial setting of FPSCR selects 32 bits as the transfer size for floating-point instructions. Please change this setting if this is required by the specifications of your application. In the code that is automatically generated by the High-performance Embedded Workshop, the set_fpscr (FPSCR_Init) function handles this initialization and is called from the PowerOn_Reset() function. This processing has been shifted to the reset handler so that it proceeds in response to other kinds of reset (manual reset).

The stack area is placed in external SDRAM, which requires initialization. When a function written in the C language is called, the stack area can be used. To avoid access to the stack area before BSC initialization, the BSC is initialized in the early section of the reset handler.

If the MMU is to be used, TLB misses must also be taken into consideration. The TLB-miss handler which is automatically generated by the High-performance Embedded Workshop uses the same stack area as the other exception handlers and other programs. If the stack area is allocated to the P0 or P3 area where address translation by the TLB is enabled, generation of a TLB-miss exception will lead to a further TLB-miss exception every time the TLB-miss handler places a value on the stack, leading to the generation of a manual reset.

A stack area (H'200) for exclusive use in cases where a TLB miss occurs is set up in area P1 where address translation by the TLB is disabled. The TLB-miss handler in this sample program uses the stack area (H'200) for exclusive use until return from the TLB-miss handler. This prevents the generation of TLB-miss exceptions by execution of the TLB-miss handler.

When the interrupt operating mode switching bit in the CPU operating mode register (CPUOPM) is in use along with automatic setting of the threshold interrupt level for acceptance in SR.IMASK or multiple interrupts, modify the processing in the respective exception handlers accordingly (in vhandler.src).

In event handling by the source program which is automatically generated by the High-performance Embedded Workshop, a common vector table (_INT_Vectors) for exception handling (resets, general exceptions, and interrupts) is looked up, and the general exception function or interrupt function is determined in accord with the value in the exception event register (EXPEVT) or the interrupt event register (INTEVT), respectively. However, the general FPU illegal exception and DMA (DEI0), and the slot FPU illegal exception and DMAC (DEI1), share exception codes. These exceptions thus cannot be distinguished in processing by the event handler that is automatically generated by the High-performance Embedded Workshop. In response to this, a countermeasure has been adopted for the general FPU illegal exception and slot FPU illegal exception so that even the event handler of the source program which is automatically generated by the High-performance Embedded Workshop can distinguish between said exceptions.

In processing by the event handler which is automatically generated by the High-performance Embedded Workshop, the BL bit is cleared so that multiple interrupts can be handled before any exception handler. Accordingly, a non-maskable interrupt (NMI) will be accepted even if a previous NMI is being processed. As a countermeasure against this, processing to clear the BL bit is not executed when the exception code corresponds to the NMI.

### 3.2.2 vecttbl.src

This file contains definitions for the vector table for exception handling (resets, general exceptions, interrupts) and interrupt mask table. The tables are looked up in processing by the code in vhandler.src (described above), and processing continues in the corresponding exception handling function (resetprg.c or intprg.c).

### 3.2.3 resetprg.c

This file contains the code for the PowerON_Reset() function, i.e. the reset processing program (see 7 in figure 2). The PowerON_Reset() function described in this application note differs from the file that the High-performance Embedded Workshop automatically generates in that the HardwareSetup() function is not called. Since sections B, R, and S are allocated to the external SDRAM, which requires initialization, BSC initialization is handled by code in the reset handler (vhandler.src).

The PowerOn_Reset() function contains code that sets the vector base register (VBR) (see 8 in figure 2), calls the _INITSCT() function (see 9 in figure 2), calls the function that enables the cache (see 10 in figure 2), sets the status register (SR) (see 11 in figure 2), and calls the main function (see 12 in figure 2).

If the sample program is extended to include settings for the internal registers of peripheral modules, the main function is intended to be the source of calls to the corresponding functions. Therefore the status register (SR) is set in privileged mode in the PowerON_Reset() function. If peripheral modules are to be used in user mode, be sure to exclude instructions which are only available in privileged mode.

Furthermore, in the file that the High-performance Embedded Workshop automatically generates, RAMCR is set before exit from the PowerON_Reset() function. This setting is not made by the sample program. Accordingly, if the processing mode is changed to user mode, subsequent access to the on-chip memory will generate an address error exception due to the function of protection against access to on-chip memory. In cases where the processing mode is changed to user mode, be sure to set the RMD bit in RAMCR to 1.

### 3.2.4 stacksct.h

This file specifies the size of the stack (initial value: H'400). Do not change the stack size by directly making changes to the stacksct.h file (to change the size or address of the stack, select **Project (P)** in the High-performance Embedded Workshop menu bar → **Edit Project Configuration (E)** → the **Stack** tab).

### 3.2.5 dbsct.c

The dbsct.c file is automatically generated by the High-performance Embedded Workshop and handles part of the initialization of sections: specifically, definition of the addresses where the initialized data sections (sections D and R) and non-initialized data section (section B) start and end. Clearing of section B to 0 and copying of data from section D to section R are handled by the call of the _INITSCT() function from within the PowerOn_Reset() function, which is in resetprg.c (see 9 in figure 2).

When the ROM support function is used to run a program in RAM, the address to which the program will be transferred should be added to the dbsct.c file so that the corresponding section is copied by the _INITSCT() function.

### 3.2.6 sh7730.c

This file contains the main function, which is called after completion of initialization (see 12 in figure 2). Code for user programs should be written in the main routine. In source programs that the High-performance Embedded Workshop automatically generates, hwsetup.c is used to make settings for the operation of peripheral modules. In this sample program, on the other hand, calls to functions that make such settings are supposed to be in the main function.

### 3.2.7 intprg.c

The programs (dummy functions) in this file are called by the handler (vhandler.src) for general exceptions and interrupts other than resets. When interrupts for peripheral functions are used, alter the dummy function by creating new functions on the basis of this sample program (and altering vect.inc and vecttbl.src in accord with any changes of function name), or include a call to a separate function within the dummy function.

Processing for tasks such as clearing interrupt request flags should be written in accord with the descriptions in the *SH7730 Group Hardware Manual* (REJ09B0359).

### 3.2.8 vect.inc

To enable reference from vhandler.src to the individual processing routines for general exceptions and interrupts in intprg.c, declarations of symbols for external reference are made in vect.inc. When a dummy function of intprg.c is rewritten as a new interrupt function, change the function name in the corresponding entry of this file accordingly. If separate functions are called from within the dummy function, changes to this file are not necessary.

Figure 2 shows the flow of processing from a power-on reset.

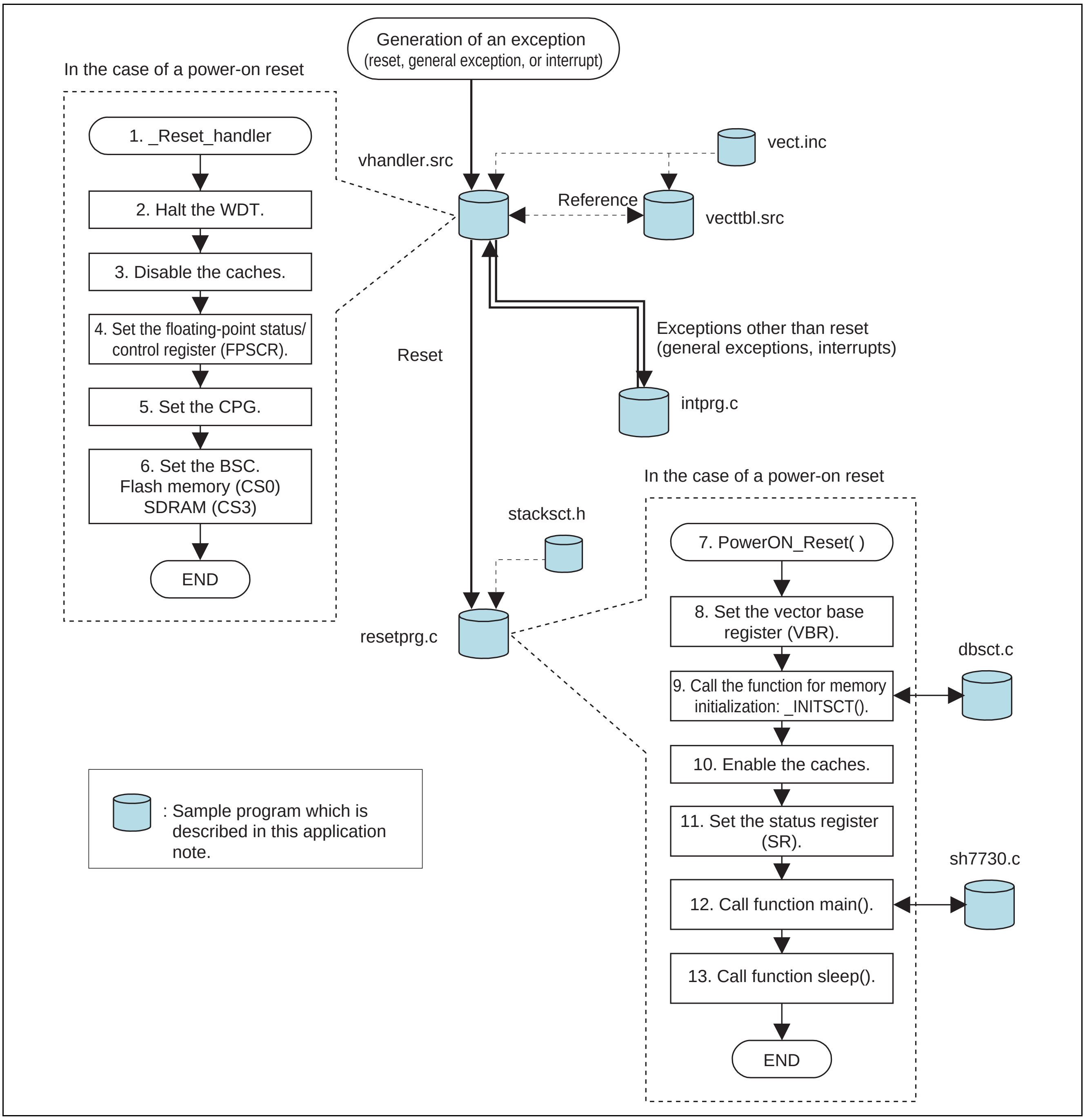

Generation of an exception
(reset, general exception, or interrupt)
In the case of a power-on reset
1. _Reset_handler
2. Halt the WDT.
3. Disable the caches.
4. Set the floating-point status/ control register (FPSCR).
5. Set the CPG.
6. Set the BSC.
Flash memory (CS0)
SDRAM (CS3)
END
vhandler.src
vect.inc
Reference
vecttbl.src
Exceptions other than reset
(general exceptions, interrupts)
Reset
intprg.c
In the case of a power-on reset
stacksct.h
7. PowerON_Reset( )
8. Set the vector base register (VBR).
resetprg.c
dbsct.c
9. Call the function for memory initialization: _INITSCT().
10. Enable the caches.
11. Set the status register (SR).
sh7730.c
12. Call function main().
13. Call function sleep().
END
: Sample program which is described in this application note.

**Figure 2 Flow of Processing from Power-On Reset**

## 3.3 Description of Settings in the Sample Program

Table 2 is a list of the settings in the sample program.

**Table 2 Settings in the Sample Program**

| Module | Description | |
|---|---|---|
| CPG | Internal clock: 266.66 MHz | |
| | SuperHyway bus clock: 133.33 MHz | |
| | Bus clock: 66.66 MHz | |
| | Peripheral clock: 33.33 MHz | |
| BSC | CS0 | NOR-type flash memory |
| | | Data bus width: 16-bit (fixed)[*1] |
| | | Cycles of delay from address/CSn assertion to RD/WEn assertion: 1.5 |
| | | Cycles of waiting for access: 4 |
| | | Cycles of delay from RD/WEn negation to address/CSn negation: 1.5 |
| | CS3 | SDRAM |
| | | Data bus width: 32 bits |
| | | Row address bits: 12 |
| | | Column address bits: 9 |
| | | CAS latency: 2 cycles |
| PFC | All the multiplexed pins can be used at initial setting. | |
| Cache | Instruction/operand cache enabled | |

Note: 1. Data bus width of area 0 is determined by the level on pin MD3.

## 3.4 Precautions Regarding the Sample Program

### 3.4.1 Allocation of Sections B, R, and S to External Memory and Initialization of Sections by the _INITSCT() Function

In this sample program, the bus state controller (BSC) is initialized before the initialization of sections B, R, and S. This is so that the sections can be allocated to external SDRAM and then initialized.

For initialization of the sections, the _INITSCT() function that copies data from section D to section R, and relocates symbol to addresses in the R section is used. Therefore, in any function which is executed before the sections are initialized (i.e. before the _INITSCT() function), avoid variables, including global variables, which are to be placed in sections to be initialized by the _INITSCT() function.

### 3.4.2 Faster Initialization of Sections

If the caching is enabled before the _INITSCT() function that handles the copying of sections is called, execution of the _INITSCT() function can be sped up. In this case, however, after the sections have been copied, data in the operand cache for sections B and R must be written back to ensure that external memory reflects the data in the cache.

For details on writing-back operations, see the following SH7730 Group Application Note: *Example of Writing Back from the Operand Cache* (REJ06B0853).

### 3.4.3 Running a Program in RAM

Follow the procedures below if you are using the _INITSCT() function with the ROM support function to develop the user programs in RAM rather than ROM for execution of the program at a higher speed.

- Organize the user program at the source for transfer under an explicit name such as “PROM section”.
- Refer to the destination for transfer by an explicit name such as “PRAM section”.
- Add the addresses of the PROM section and PRAM section to the structure under C$DSEC in dbsct.c.
- Since the _INITSCT() function is placed in section P, section P should be allocated to ROM.
- To add PROM and PRAM to sections in ROM and RAM, respectively, select **Build (B)** in the High-performance Embedded Workshop menu bar → **SuperH RISC engine Standard Toolchain** → **Link/Library** → **Category (Y)** “Output” → **Show entries for (S)** “ROM to RAM mapped sections”.

### 3.4.4 Stack Pointer Addresses

The address of the stack pointer at the start of the PowerOn_Reset() function (specified as the entry point by the #pragma entry directive that immediately precedes it) is the address specified by the High-performance Embedded Workshop at the time of project generation.

To change the address and size of stack area, select **Project (P)** in the High-performance Embedded Workshop menu bar → **Edit Project Configuration (E)** → the **Stack** tab. Do not directly change the address allocation of section S. If this is directly changed, the dialog box might not be activated by selection of the **Edit** menu item.

## 4. Listing of the Sample Program

1. Sample Program Listing: "vhandler.src"

```
;------------------------------------------------------------------------
;                                                                          |
;   FILE             :vhandler.src                                     |
;   DATE             :Tue, Oct 07, 2008                                |
;   DESCRIPTION      :Reset/Interrupt Handler                          |
;   CPU TYPE         :SH7730                                           |
;                                                                          |
;   This file is generated by Renesas Project Generator (Ver.4.9).        |
;                                                                          |
;------------------------------------------------------------------------
;/*********************************************************************
;*
;* Device        : SH-4A/SH7730
;*
;* File Name     : vhandler.src
;*
;* Abstract      : Reset/Interrupt Handler.
;*
;* History       : 1.00  (2008-10-01)  [Hardware Manual Revision : 1.00]
;*
;* Copyright(c) 2008 Renesas Technology Corp.
;*               And Renesas Solutions Corp.,All Rights Reserved.
;*
;*********************************************************************/

             .include    "env.inc"
             .include    "vect.inc"

;
ILLEGALFPU_CODE:            .equ    H'800
DUMMY_ILLEGALFPU_CODE:          .equ    H'880
ILLEGALSLOTFPU_CODE:        .equ    H'820
DUMMY_ILLEGALSLOTFPU_CODE:      .equ    H'8A0
INT_NMI_CODE:               .equ    H'1C0
;
IMASKclr:    .equ    H'FFFFFF0F
RBBLclr:     .equ    H'CFFFFFFF
MDRBBLset:       .equ    H'70000000
MDRBset:     .equ    H'60000000
RBclr:           .equ    H'DFFFFFFF

                 .import        _RESET_Vectors
                 .import        _INT_Vectors
                 .import        _INT_MASK

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;     macro definition                                              ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                     .macro  PUSH_EXP_BASE_REG
                 stc.l   ssr,@-r15          ; save ssr
                 stc.l   spc,@-r15          ; save spc
                 sts.l   pr,@-r15           ; save context registers
                 sts.l   fpscr,@-r15        ; save fpscr registers
                 stc.l   r7_bank,@-r15
                 stc.l   r6_bank,@-r15
                 stc.l   r5_bank,@-r15
                 stc.l   r4_bank,@-r15
                 stc.l   r3_bank,@-r15
                 stc.l   r2_bank,@-r15
                 stc.l   r1_bank,@-r15
                 stc.l   r0_bank,@-r15
                         .endm
```

```
;
                        .macro  POP_EXP_BASE_REG
                 ldc.l  @r15+,r0_bank             ; recover registers
                 ldc.l  @r15+,r1_bank
                 ldc.l  @r15+,r2_bank
                 ldc.l  @r15+,r3_bank
                 ldc.l  @r15+,r4_bank
                 ldc.l  @r15+,r5_bank
                 ldc.l  @r15+,r6_bank
                 ldc.l  @r15+,r7_bank
                 lds.l  @r15+,fpscr
                 lds.l  @r15+,pr
                 ldc.l  @r15+,spc
                 ldc.l  @r15+,ssr
                        .endm
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;     reset                                                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                 .section   RSTHandler,code
_ResetHandler:
                        mov.l   #H'A4520004,r0 ;set RWTCSR address
                        mov.l   #H'0000A507,r1 ;RWDT disable
                        mov.w   r1,@r0

                        mov.l   #H'FF00001C,r0 ;set CCR address
                        mov.l   #H'00000808,r1 ;IC,OC Invalidate
                        mov.l   r1,@r0

                        mov.l   #H'00040001,r0 ;set single precision mode
                        ;mov.l  #H'000C0001,r0 ;set double precision mode
                        lds.l   r0,fpscr

                        mov.l   #H'A4150000,r0 ;set CPG address
                        mov.l   #H'07002508,r1 ;  * Clockin = 33.333 MHz, CKIO = 66.6 MHz
                                               ;  * I Clock = 266 MHz, B Clock = 66.6 MHz,
                                               ;  * P Clock = 33.3 MHz
                        mov.l   r1,@r0

                        mov.l   #CS0_INIT,r0
                        jmp            @r0
                        nop

CS0_INIT_END:
                        mov.l   #SDRAM_INIT,r0
                        jmp            @r0
                        nop

SDRAM_INIT_END:
                        mov.l   #EXPEVT,r0
                        mov.l   @r0,r0
                        shlr2   r0
                        shlr    r0
                        mov.l   #_RESET_Vectors,r1
                        mov.l   @(r0,r1),r0 ;set Reset function address
                        jmp     @r0
                        nop
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;     exceptional interrupt                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                 .section   INTHandler,code
                 .export    _INTHandlerPRG
_INTHandlerPRG:
_ExpHandler:
                    PUSH_EXP_BASE_REG
```

```
;
                    mov.l   #EXPEVT,r0                    ; set event address
                    mov.l   @r0,r1                        ; set exception code
;
                    mov.l   #ILLEGALFPU_CODE,r2           ; H'800
                    cmp/eq  r1,r2
                    bf              exp_01
                    mov.l   #DUMMY_ILLEGALFPU_CODE,r1     ; H'800 -> H'880
                    bra             exp_10
                    nop
exp_01:
                    mov.l   #ILLEGALSLOTFPU_CODE,r2       ; H'820
                    cmp/eq  r1,r2
                    bf              exp_10
                    mov.l   #DUMMY_ILLEGALSLOTFPU_CODE,r1 ; H'820 -> H'8A0
exp_10:
;
                    mov.l   #_INT_Vectors,r0              ; set vector table address
                    add             #-(h'40),r1           ; exception code - h'40
                    shlr2   r1
                    shlr    r1
                    mov.l   @(r0,r1),r3                   ; set interrupt function addr
;
                    mov.l   #_INT_MASK,r0                 ; interrupt mask table addr
                    shlr2   r1
                    mov.b   @(r0,r1),r1                   ; interrupt mask
                    extu.b  r1,r1
;
                    stc             sr,r0                 ; save sr
                    mov.l   #(RBBLclr&IMASKclr),r2        ; RB,BL,mask clear data
                    and             r2,r0                 ; clear mask data
                    or              r1,r0                 ; set interrupt mask
                    ldc             r0,ssr                ; set current status
;
                    ldc.l   r3,spc
                    mov.l   #__int_term,r0                ; set interrupt terminate
                    lds             r0,pr
;
                    rte
                    nop
;
                    .pool
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;     Interrupt terminate                                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             .align  4
__int_term:
                    mov.l   #MDRBBLset,r0         ; set MD,BL,RB
                    ldc.l   r0,sr                 ;
                    POP_EXP_BASE_REG
                    rte                           ; return
                    nop
;
                    .pool
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;     TLB miss interrupt                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                 .org   H'300
_TLBmissHandler:

                    mov.l   #(SP_STACK+H'200),r15 ;set SP_STACK(for only TLBmiss) pointer
                    stc.l   sgr,@-r15

                    PUSH_EXP_BASE_REG
```

```
;
                     mov.l   #EXPEVT,r0                ; set event address
                     mov.l   @r0,r1                    ; set exception code
                     mov.l   #_INT_Vectors,r0          ; set vector table address
                     add            #-(h'40),r1        ; exception code - h'40
                     shlr2   r1
                     shlr    r1
                     mov.l   @(r0,r1),r3               ; set interrupt function addr
;
                     mov.l   #_INT_MASK,r0             ; interrupt mask table addr
                     shlr2   r1
                     mov.b   @(r0,r1),r1               ; interrupt mask
                     extu.b r1,r1
;
                     stc            sr,r0              ; save sr
                     mov.l   #(RBBLclr&IMASKclr),r2    ; RB,BL,mask clear data
                     and            r2,r0              ; clear mask data
                     or             r1,r0              ; set interrupt mask
                     ldc            r0,ssr             ; set current status
;
                     ldc.l   r3,spc
                     mov.l   #__TLBMISS_INT_TERM,R0    ;set interrupt terminate
                     lds            r0,pr
;
                     rte
                     nop

                     .align 4

__TLBMISS_INT_TERM:
                     mov.l   #MDRBBLset,r0             ;set MD,BL,RB
                     ldc.l   r0,sr

                     POP_EXP_BASE_REG

                     ldc.l   @r15+,sgr
                     stc.l   sgr,r15
                     rte
                     nop
;
                     .pool
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;     IRQ                                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             .org    H'500
_IRQHandler:
                     PUSH_EXP_BASE_REG
;
                     mov.l   #INTEVT,r0                ; set event address
                     mov.l   @r0,r1                    ; set exception code

                     mov.l   #INT_NMI_CODE,r2          ; H'1C0
                     cmp/eq r1,r2
                     bf  no_nmi

;add NMI CASE =========================================

                     mov.l   #_INT_NMI,r3
                     jsr        @r3
                     nop

                     POP_EXP_BASE_REG

                     rte
                     nop
```

```
 no_nmi:
;add end NMI CASE =====================================

                     mov.l   #_INT_Vectors,r0              ; set vector table address
                     add            #-(h'40),r1            ; exception code - h'40
                     shlr2   r1
                     shlr    r1
                     mov.l   @(r0,r1),r3                   ; set interrupt function addr
;
                     mov.l   #_INT_MASK,r0                 ; interrupt mask table addr
                     shlr2   r1
                     mov.b   @(r0,r1),r1                   ; interrupt mask
                     extu.b  r1,r1
;
                     stc            sr,r0                  ; save sr
                     mov.l   #(RBBLclr&IMASKclr),r2        ; RB,BL,mask clear data
                     and            r2,r0                  ; clear mask data
                     or             r1,r0                  ; set interrupt mask
                     ldc            r0,ssr                 ; set current status
;
                     ldc.l   r3,spc
                     mov.l   #__int_term,r0                ; set interrupt terminate
                     lds            r0,pr
;
                     rte
                     nop
;
                     .pool

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; CS0 INIT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CS0_INIT:
                     mov.l   #H'FEC10004,r0        ;set CS0BCR address
                     mov.l   #H'10480400,r1        ;set for FLASHROM(spansion
S29AL032D70TFI04)
                     mov.l   r1,@r0

                     mov.l   #H'FEC10024,r0        ;set CS0WCR address
                     mov.l   #H'00000A41,r1
                     mov.l   r1,@r0

                     mov.l   #CS0_INIT_END,r0
                     jmp     @r0
                     nop

                     .pool

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; SDRAM INIT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SDRAM_INIT:
             mov.l   #H'FEC1000C,r0     ;set CS3BCR address
             mov.l   #H'10004600,r1     ;set for SDRAM(Samsung K4S281632F-UC75)
                                        ;32bit bus-width, IWW 1cyc
             mov.l   r1,@r0

             mov.l   #H'FEC1002C,r0 ;set CS3WCR address
             mov.l   #H'00002492,r1 ;tRP 2cyc
                                    ;tRCD 2cyc
                                    ;A3CL 2cyc
                                    ;tRWL 2cyc
                                    ;tRC 6cyc
             mov.l   r1,@r0

             mov.l   #H'FEC10044,r0 ;set SDCR address
             mov.l   #H'00000809,r1 ;auto refresh mode, row 12bit, column 9bit
             mov.l   r1,@r0

             mov.l   #H'FEC10050,r0 ;set RTCOR address
             mov.l   #H'a55a003E,r1 ;refresh rate
             mov.l   r1,@r0


             mov.l   #H'000030d4,r0
LOOP1:
             dt      r0
             bf      LOOP1 ;200µs wait
             nop
             nop

             mov.l   #H'FEC10048,r0 ;set RTCSR address
             mov.l   #H'a55a0010,r1
             mov.l   r1,@r0

             mov.l   #H'FEC15080,r0 ;set SDMR3(32bit bus-width, CL=2, burstR/W(burst length=1))
             mov.l   #H'00000000,r1
             mov.w   r1,@r0

             mov.l   #SDRAM_INIT_END,r0
             jmp     @r0
             nop

             .pool

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; SPECIAL STACK(for TLBmiss Handler)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                     .section SP_S,data
SP_STACK:
                     .res.b  H'200
                     .end
```

2. Sample Program Listing: "vecttbl.src"

```
;------------------------------------------------------------------------
;                                                                        |
;   FILE            :vecttbl.src                                         |
;   DATE            :Fri, Aug 01, 2008                                   |
;   DESCRIPTION     :Initialization of Vector Table                      |
;   CPU TYPE        :SH7730                                              |
;                                                                        |
;   This file is generated by Renesas Project Generator (Ver.4.9).       |
;                                                                        |
;------------------------------------------------------------------------
;/*********************************************************************
;*
;* Device        : SH-4A/SH7730
;*
;* File Name     : vecttbl.src
;*
;* Abstract      : Initialize of Vector Table.
;*
;* History       : 1.00  (2008-10-01)  [Hardware Manual Revision : 1.00]
;*
;* Copyright(c) 2008 Renesas Technology Corp.
;*               And Renesas Solutions Corp.,All Rights Reserved.
;*
;*********************************************************************/

                        .include    "vect.inc"

                        .section    VECTTBL,data
                        .export             _RESET_Vectors

_RESET_Vectors:
;<<VECTOR DATA START (POWER ON RESET)>>
                        ;H'000              Power On Reset(H-UDI RESET)
                        .data.l             _PowerON_Reset
;<<VECTOR DATA END (POWER ON RESET)>>
;<<VECTOR DATA START (MANUAL RESET)>>
                        ;H'020              Manual Reset
                        .data.l             _Manual_Reset
;<<VECTOR DATA END (MANUAL RESET)>>
                        ;H'040-120          Reserved
                        .datab.l    8,H'00000000
                        ;H'140              TLB Reset(DATA TLB Reset)
                        .data.l             _TLB_Reset

                        .section    INTTBL,data
                        .export             _INT_Vectors

_INT_Vectors:
                        ; H'040     Data TLB miss exception(read)
                        .data.l             _INT_TLB_MISS_READ_EXP
                        ; H'060     Data TLB miss exception(write)
                        .data.l             _INT_TLB_MISS_WRITE_EXP
                        ; H'080     Initial page write exception
                        .data.l             _INT_TLB_INIT_PAGE_EXP
                        ; H'0A0     Data TLB protection violation exception (read)
                        .data.l             _INT_TLB_PROTECT_READ_EXP
                        ; H'0C0     Data TLB protection violation exception (write)
                        .data.l             _INT_TLB_PROTECT_WRITE_EXP
                        ; H'0E0     Data address error(read)
                        .data.l             _INT_ADR_ERROR_READ
                        ; H'100     Data address error(write)
                        .data.l             _INT_ADR_ERROR_WRITE
                        ; H'120     FPU exception
                        .data.l             _INT_FPU_EXP
                        ; H'140     Instruction TLB multiple-hit exception
                        .data.l             _TLB_Reset
                        ; H'160     Unconditional trap(TRAPA)
                        .data.l             _INT_TRAP
                        ; H'180     General illegal instruction exception
                        .data.l             _INT_ILLEGAL_INST_EXP
                        ; H'1A0     Slot illegal instruction exception
                        .data.l             _INT_ILLEGAL_SLOT_EXP
;EXTERNAL INTERRUPT
                        ; H'1C0     NMI
                        .data.l             _INT_NMI
                        ; H'1E0     USER_BREAK
                        .data.l             _INT_USER_BREAK
                        ; H'200     IRL_LEVEL15
                        .data.l             _INT_IRL_LEVEL15
                        ; H'220     IRL_LEVEL14
                        .data.l             _INT_IRL_LEVEL14
                        ; H'240     IRL_LEVEL13
                        .data.l             _INT_IRL_LEVEL13
                        ; H'260     IRL_LEVEL12
                        .data.l             _INT_IRL_LEVEL12
                        ; H'280     IRL_LEVEL11
                        .data.l             _INT_IRL_LEVEL11
                        ; H'2A0     IRL_LEVEL10
                        .data.l             _INT_IRL_LEVEL10
                        ; H'2C0     IRL_LEVEL9
                        .data.l             _INT_IRL_LEVEL9
                        ; H'2E0     IRL_LEVEL8
                        .data.l             _INT_IRL_LEVEL8
                        ; H'300     IRL_LEVEL7
                        .data.l             _INT_IRL_LEVEL7
                        ; H'320     IRL_LEVEL6
                        .data.l             _INT_IRL_LEVEL6
                        ; H'340     IRL_LEVEL5
                        .data.l             _INT_IRL_LEVEL5
                        ; H'360     IRL_LEVEL4
                        .data.l             _INT_IRL_LEVEL4
                        ; H'380     IRL_LEVEL3
                        .data.l             _INT_IRL_LEVEL3
                        ; H'3A0     IRL_LEVEL2
                        .data.l             _INT_IRL_LEVEL2
                        ; H'3C0     IRL_LEVEL1
                        .data.l             _INT_IRL_LEVEL1
                        ;H'3E0              Reserved
                        .data.l             H'00000000
;TMU-ch0
                        ;H'400              TMU_TUNI0
                        .data.l             _INT_TMU0_TUNI0
;TMU-ch1
                        ;H'420              TMU_TUNI1
                        .data.l             _INT_TMU1_TUNI1
;TMU-ch2
                        ;H'440              TMU_TUNI2
                        .data.l             _INT_TMU2_TUNI2
                        ;H'460              Reserved
                        .data.l             H'00000000
;RTC
                        ;H'480              RTC ATI
                        .data.l             _INT_RTC_ATI
                        ;H'4A0              RTC PRI
                        .data.l             _INT_RTC_PRI
                        ;H'4C0                  RTC CUI
                        .data.l                 _INT_RTC_CUI
;PINT
                        ;H'4E0 PINT PINTA
                        .data.l                 _INT_PINT_PINTA
                        ;H'500 PINT PINTB
                        .data.l                 _INT_PINT_PINTB
                        ;H'520-5C0              Reserved
                        .datab.l    6,H'00000000
;H-UDI
                        ;H'5E0                  H-UDII
                        .data.l                 _INT_H_UDII
;IRQ
                        ;H'600                  IRQ IRQ0
                        .data.l                 _INT_IRQ_IRQ0
                        ;H'620                  IRQ IRQ1
                        .data.l                 _INT_IRQ_IRQ1
                        ;H'640                  IRQ IRQ2
                        .data.l                 _INT_IRQ_IRQ2
                        ;H'660                  IRQ IRQ3
                        .data.l                 _INT_IRQ_IRQ3
                        ;H'680                  IRQ IRQ4
                        .data.l                 _INT_IRQ_IRQ4
                        ;H'6A0                  IRQ IRQ5
                        .data.l                 _INT_IRQ_IRQ5
                        ;H'6C0                  IRQ IRQ6
                        .data.l                 _INT_IRQ_IRQ6
                        ;H'6E0                  IRQ IRQ7
                        .data.l                 _INT_IRQ_IRQ7
;SIM
                        ;H'700                  SIM ERI
                        .data.l                 _INT_SIM_ERI
                        ;H'720                  SIM RXI
                        .data.l                 _INT_SIM_RXI
                        ;H'740                  SIM TXI
                        .data.l                 _INT_SIM_TXI
                        ;H'760                  SIM TEI
                        .data.l                 _INT_SIM_TEI
                        ;H'780-7C0              Reserved
                        .datab.l    3,H'00000000
;IIC1
                        ;H'7E0                  IIC1 IICI1
                        .data.l                 _INT_IIC1_IICI1
;DMAC(1)
                        ;H'800                  DMAC DEI0 ; Illegal FPU -> Dummy Code H'880
                        .data.l                 _INT_DMAC_DEI0
                        ;H'820                  DMAC DEI1 ; Illegal slot FPU -> Dummy Code H'8A0
                        .data.l                 _INT_DMAC_DEI1
                        ;H'840                  DMAC DEI2
                        .data.l                 _INT_DMAC_DEI2
                        ;H'860                  DMAC DEI3
                        .data.l                 _INT_DMAC_DEI3
;Illegal FPU
                        ;H'880                  Reserved -> Used as Illegal FPU
                        .data.l                 _INT_ILLEGAL_FPU
                        ;H'8A0                  Reserved -> Used as Illegal slot FPU
                        .data.l                 _INT_ILLEGAL_SLOT_FPU
                        ;H'8C0-920              Reserved
                        .datab.l    4,H'00000000
;IRDA
                        ;H'940 IRDA IRDAI0
                        .data.l                 _INT_IRDA_IRDAI0
                        ;H'960 IRDA IRDAI1
                        .data.l            _INT_IRDA_IRDAI1
;ADC
                        ;H'980 ADC ADI
                        .data.l            _INT_ADC_ADI
;TPU
                        ;H'9A0 TPU TPUI0
                        .data.l            _INT_TPU_TPUI0
                        ;H'9C0 TPU TPUI1
                        .data.l            _INT_TPU_TPUI1
                        ;H'9E0-B60         Reserved
                        .datab.l    13,H'00000000
;DMAC(2)
                        ;H'B80             DMAC DEI4
                        .data.l            _INT_DMAC_DEI4
                        ;H'BA0             DMAC DEI5
                        .data.l            _INT_DMAC_DEI5
                        ;H'BC0             DMAC DADERR
                        .data.l            _INT_DMAC_DADERR
                        ;H'BE0             Reserved
                        .data.l            H'00000000
;SCIF
                        ;H'C00             SCIF SCIFI0
                        .data.l            _INT_SCIF_SCIFI0
                        ;H'C20             SCIF SCIFI1
                        .data.l            _INT_SCIF_SCIFI1
                        ;H'C40             SCIF SCIFI2
                        .data.l            _INT_SCIF_SCIFI2
                        ;H'C60             SCIF SCIFI3
                        .data.l            _INT_SCIF_SCIFI3
                        ;H'C80             SCIFA SCIFI4
                        .data.l            _INT_SCIFA_SCIFI4
                        ;H'CA0             SCIFA SCIFI5
                        .data.l            _INT_SCIFA_SCIFI5
                        ;H'CC0-E40         Reserved
                        .datab.l    13,H'00000000
;IIC0
                        ;H'E60             IIC0 IICI0
                        .data.l            _INT_IIC0_IICI0
                        ;H'E80-EE0         Reserved
                        .datab.l    4,H'00000000
;CMT
                        ;H'F00             CMTI
                        .data.l            _INT_CMT_CMTI
;SIOF
                        ;H'F20             SIOFI
                        .data.l            _INT_SIOF_SIOFI
                        ;H'F40-FE0         Reserved
                        .datab.l    6,H'00000000

                 .export    _INT_MASK
_INT_MASK:
                        ; interrupt priority mask level(31 to 0)

                        ;H'040             Data TLB miss exception(read)
                        .data.b            H'00
                        ;H'060             Data TLB miss exception(write)
                        .data.b            H'00
                        ;H'080             Initial page write exception
                        .data.b            H'00
                        ;H'0A0             Data TLB protection violation exception (read)
                        .data.b            H'00
                        ;H'0C0             Data TLB protection violation exception (write)
                        .data.b            H'00
```

```
                         ;H'0E0             Data address error(read)
                         .data.b            H'00
                         ;H'100             Data address error(write)
                         .data.b            H'00
                         ;H'120             FPU exception
                         .data.b            H'00
                         ;H'140             Instruction TLB multiple-hit exception
                         .data.b            H'00
                         ;H'160 TRAPA
                         .data.b            H'00
                         ;H'180             ILLEGAL_INST
                         .data.b            H'00
                         ;H'1A0             ILLEGAL_SLOT
                         .data.b            H'00
;EXTERNAL INTERRUPT
                         ;H'1c0             NMI
                         .data.b            H'00
                         ;H'1E0             USER_BREAK
                         .data.b            H'00
                         ;H'200-3c0         IRL
                         .datab.b   15,H'00
                         ;H'3e0             Reserved
                         .data.b            H'00
;TMU
                         ;H'400             TMU TUNI0
                         .data.b            H'00
                         ;H'420             TMU TUNI1
                         .data.b            H'00
                         ;H'440             TMU TUNI2
                         .data.b            H'00
                         ;H'460             Reserved
                         .data.b            H'00
;RTC
                         ;H'480             RTC ATI
                         .data.b            H'00
                         ;H'4A0             RTC PRI
                         .data.b            H'00
                         ;H'4C0             RTC CUI
                         .data.b            H'00
;PINT
                         ;H'4E0             PINT PINTA
                         .data.b            H'00
                         ;H'500             PINT PINTB
                         .data.b            H'00
                         ;H'520-5C0         Reserved
                         .datab.b   6,H'00
;HUDI
                         ;H'5E0             _INT_H_UDII
                         .data.b            H'00
;IRQ
                         ;H'600             IRQ IRQ0
                         .data.b            H'00
                         ;H'620             IRQ IRQ1
                         .data.b            H'00
                         ;H'640             IRQ IRQ2
                         .data.b            H'00
                         ;H'660             IRQ IRQ3
                         .data.b            H'00
                         ;H'680             IRQ IRQ4
                         .data.b            H'00
                         ;H'6A0             IRQ IRQ5
                         .data.b            H'00
                         ;H'6C0             IRQ IRQ6
                         .data.b            H'00
                         ;H'6E0             IRQ IRQ7
                         .data.b            H'00
;SIM
                         ;H'700             SIM ERI
                         .data.b            H'00
                         ;H'720             SIM RXI
                         .data.b            H'00
                         ;H'740             SIM TXI
                         .data.b            H'00
                         ;H'760             SIM TEI
                         .data.b            H'00
                         ;H'780-7C0         Reserved
                         .datab.b    3,H'00
;IIC1
                         ;H'7E0             IIC1 IICI1
                         .data.b            H'00
;DMAC(1)
                         ;H'800             DMAC DEI0
                         .data.b            H'00
                         ;H'820             DMAC DEI1
                         .data.b            H'00
                         ;H'840             DMAC DEI2
                         .data.b            H'00
                         ;H'860             DMAC DEI3
                         .data.b            H'00
                         ;H'880             Reserved -> Used as Illegal FPU
                         .data.b            H'00
                         ;H'8A0             Reserved -> Used as Illegal slot FPU
                         .data.b            H'00
                         ;H'8C0-920         Reserved
                         .datab.b    4,H'00
;IRDA
                         ;H'940             IRDA IRDAI0
                         .data.b            H'00
                         ;H'960             IRDA IRDAI1
                         .data.b            H'00
;ADC
                         ;H'980             ADC ADI
                         .data.b            H'00
;TPU
                         ;H'9A0             TPU TPUI0
                         .data.b            H'00
                         ;H'9C0             TPU TPUI1
                         .data.b            H'00
                         ;H'9E0-B60         Reserved
                         .datab.b    13,H'00
;DMAC
                         ;H'B80             DMAC DEI4
                         .data.b            H'00
                         ;H'BA0             DMAC DEI5
                         .data.b            H'00
                         ;H'BC0             DMAC DADERR
                         .data.b            H'00
                         ;H'BE0             Reserved
                         .data.b            H'00
;SCIF
                         ;H'C00             SCIF SCIFI0
                         .data.b            H'00
                         ;H'C20             SCIF SCIFI1
                         .data.b            H'00
                         ;H'C40             SCIF SCIFI2
                         .data.b            H'00
                         ;H'C60             SCIF SCIFI3
                         .data.b            H'00
                         ;H'C80             SCIFA SCIFI4
                         .data.b            H'00
                         ;H'CA0             SCIFA SCIFI5
                         .data.b            H'00
                         ;H'CC0-E40         Reserved
                         .datab.b   13,H'00
;IIC0
                         ;H'E60 IIC0        IICI0
                         .data.b            H'00
                         ;H'E80-EE0         Reserved
                         .datab.b   4,H'00
;CMT
                         ;H'F00             CMT
                         .data.b            H'00
;SIOF
                         ;H'F20             SIOFI
                         .data.b            H'00
                         ;H'F40-FE0         Reserved
                         .datab.b   6,H'00

                         .end
```

## 3. Sample Program Listing: "resetprg.c"

```
/***********************************************************************/
/*                                                                    */
/*  FILE             :resetprg.c                                      */
/*  DATE             :Wed, Dec 24, 2008                               */
/*  DESCRIPTION      :Reset Program                                   */
/*  CPU TYPE         :SH7730                                          */
/*                                                                    */
/*  This file is generated by Renesas Project Generator (Ver.4.9).    */
/*                                                                    */
/***********************************************************************/
/*""FILE COMMENT""*********** Technical reference data ****************
* System Name    : SH7730 Sample Program
* File Name      : resetprg.c
* Abstract       : Sample Program of the SH7730 Initialization
* Version        : Ver 1.00
* Device         : SH7730
* Tool-Chain : SuperH RISC engine Standard Toolchain Ver.9.1.1.0
* OS             : None
* H/W Platform   : The SH-4A evaluation board AP-SH4A-1A is
*                 available from AlphaProject Co., Ltd.
* Description    : Sample program for the SH7730 initialization
*                :
* Operation      :
* Disclaimer     :
*                :
* Copyright (C) 2008. Renesas Technology Corp., All Rights Reserved.
*
***********************************************************************
* History        : 27.May.2008 Ver. 1.00 First Release
*""FILE COMMENT END""**************************************************/
#include  <machine.h>
#include  <_h_c_lib.h>
#include  "typedefine.h"
#include  "stacksct.h"
#include  "iodefine.h"
#include  "cache.h"         /* Add cache function */


#define SR_Init         0x40000000
#define INT_OFFSET      0x100UL

#ifdef __cplusplus
extern "C" {
#endif
extern void INTHandlerPRG(void);
void PowerON_Reset(void);
void Manual_Reset(void);
void main(void);
#ifdef __cplusplus
}
#endif

//#ifdef __cplusplus            // Enable I/O in the application(both SIM I/O and hardware I/O)
//extern "C" {
//#endif
//extern void _INIT_IOLIB(void);
//extern void _CLOSEALL(void);
//#ifdef __cplusplus
//}
//#endif

//extern void srand(_UINT);     // Remove the comment when you use rand()
```

```
//extern _SBYTE *_s1ptr;        // Remove the comment when you use strtok()

#ifdef __cplusplus                  // Use Hardware Setup
extern "C" {
#endif
extern void HardwareSetup(void);
#ifdef __cplusplus
}
#endif

//#ifdef __cplusplus            // Remove the comment when you use global class object
//extern "C" {                      // Sections C$INIT and C$END will be generated
//#endif
//extern void _CALL_INIT(void);
//extern void _CALL_END(void);
//#ifdef __cplusplus
//}
//#endif


/* = = = = Changing section name to ResetPRG = = = = */
#pragma section ResetPRG

/* = = = = Specification of entry function = = = = */
#pragma entry PowerON_Reset
/*""FUNC COMMENT""***************************************************
* ID                     :
* Outline                : Function for CPU Initialization
* Include                :
* Declaration            : void PowerON_Reset(void)
* Description            : CPU initialization routine. Its address is registered in
*                        : the vector table entry for power-on reset exception handling.
*                        : This is the first function executed after a power-on reset.
*                        :
* Disclaimer             : Enable processing which has been commented out as required.
* Argument               : none
* Return Value           : none
* Calling Functions      :
*""FUNC COMMENT END""**********************************************/
void PowerON_Reset(void)
{

    set_vbr((void *)((_UINT)INTHandlerPRG - INT_OFFSET));

    /* = = = = Initialization of sections B and D = = = = */
    _INITSCT();

//    errno=0;                      // Remove the comment when you use errno
//    srand((_UINT)1);              // Remove the comment when you use rand()
//    _s1ptr=NULL;                  // Remove the comment when you use strtok()


    /* ==== Cache setting ==== */
    /* ==== For details on this function, see the SH7730 Group Application Note: Examples
       of Cache Memory Settings (REJ06B0851).  ==== */
    cache_set_ccr(CACHE_I_ON | CACHE_O_ON );

    /* ==== Setting the status register (privileged mode) ==== */
    set_cr(SR_Init);

    main();

//    _CLOSEALL();                  // Close I/O in the application(both SIM I/O and hardware I/O)

//    _CALL_END();                  // Remove the comment when you use global class object

    sleep();
}

//#pragma entry Manual_Reset    // Remove the comment when you use Manual Reset
/*""FUNC COMMENT""***************************************************
* ID                     :
* Outline                : Manual reset processing
* Include                :
* Declaration            : void Manual_Reset_PC (void)
* Description            : The address of this function is registered in the vector
*                        : table entry for manual reset exception handling.
*                        :
* Disclaimer             : No processing is defined in this sample program.
*                        : Add processing as required.
* Argument               : none
* Return Value           : none
* Calling Functions      :
*""FUNC COMMENT END""**********************************************/
void Manual_Reset (void)
{
/* NOP */
}
/* END of File */
```

4. Sample Program Listing: "dbsct.c"

```
/***************************************************************************/
/*                                                                         */
/*  FILE:        dbsct.c                                                   */
/*  DATE:        Wed, Dec 24, 2008                                         */
/*  DESCRIPTION: Setting of the B and R sections                           */
/*  CPU TYPE:    SH7730                                                    */
/*                                                                         */
/*  This file is generated by Renesas Project Generator (Ver.4.9).         */
/*                                                                         */
/***************************************************************************/



#include "typedefine.h"

#pragma section $DSEC
static const struct {
    _UBYTE *rom_s;       /* First address of initialized data section in ROM */
    _UBYTE *rom_e;       /* Last address of initialized data section in ROM  */
    _UBYTE *ram_s;       /* First address of initialized data section in RAM */
}   DTBL[] = {
    { __sectop("D"), __secend("D"), __sectop("R") }
};
#pragma section $BSEC
static const struct {
    _UBYTE *b_s;         /* First address of non-initialized data section */
    _UBYTE *b_e;         /* Last address of non-initializaed data section */
}   BTBL[] = {
    { __sectop("B"), __secend("B") }
};
```

## 5. Sample Program Listing: "sh7730.c"

```
/*""FILE COMMENT""*********** Technical reference data ****************
 * System Name    : SH7730 Sample Program
 * File Name      : sh7730.c
 * Abstract       : Sample Program for the SH7730 Initialization
 * Version        : Ver 1.00
 * Device         : SH7730
 * Tool-Chain     : SuperH RISC engine Standard Toolchain Ver.9.1.1.0
 * OS             : None
 * H/W Platform   : The SH-4A evaluation board AP-SH4A-1A is
 *                 available from AlphaProject Co., Ltd.
 * Description    : Sample program for SH7730 initialization
 *                :
 * Operation      :
 * Disclaimer     :
 *                :
 * Copyright (C) 2008. Renesas Technology Corp., All Rights Reserved.
 *
*********************************************************************
 * History        : 27.May.2008 Ver. 1.00 First Release
 *""FILE COMMENT END""**********************************************/
#include <machine.h>
#include "iodefine.h"

//#include "typedefine.h"
#ifdef __cplusplus
//#include <ios>                        // Remove the comment when you use ios
//_SINT ios_base::Init::init_cnt;           // Remove the comment when you use ios
#endif

void main(void);

#ifdef __cplusplus
extern "C" {
void abort(void);
}
#endif

/*""FUNC COMMENT""**************************************************
 * ID                     :
 * Outline                : ‶main″ function
 * Include                :
 * Declaration            : void main(void)
 * Description            : “main” function of the sample program
 *                            :
 * Argument               : none
 * Return Value           : none
 * Calling Functions      :
 *""FUNC COMMENT END""**********************************************/
void main(void)
{

}

#ifdef __cplusplus
void abort(void)
{

}
#endif
```

6. Sample Program Listing: "intprg.c"

```
/*""FILE COMMENT""*********** Technical reference data ****************
* System Name:  SH7730 Sample Program
* File Name:    intprg.c
* Abstract:     Sample Program of the SH7730 Initialization
* Version:      Ver 1.00
* Device:       SH7730
* Tool-Chain:   SuperH RISC engine Standard Toolchain Ver.9.1.1.0
* OS:           None
* H/W Platform: The AP-SH4A-1A board from AlphaProject Co., Ltd.
* Description:  This is a sample program for the SH7730 initialization.
*               intprg.src has been changed to the C language.
*
* Operation:
* Disclaimer:
*
* Copyright (C) 2008. Renesas Technology Corp., All Rights Reserved.
*
***********************************************************************
* History:      27.May.2008 Ver. 1.00 First Release
*""FILE COMMENT END""**************************************************/
#include <machine.h>
#include "iodefine.h"

/* --- RAM allocation variable declaration --- */

#pragma section IntPRG
/* H'040 Data TLB miss exception(read) */
void INT_TLB_MISS_READ_EXP(void)
{
}

…Snip…

/* H'1C0 NMI */
void INT_NMI(void)
{
}

…Snip…
```

## 5. Documents for Reference

- Software Manual
  SH-4A Software Manual (REJ09B0003)

- Hardware Manual
  SH7730 Group Hardware Manual (REJ09B0359)

- Application Note
  SuperH RISC engine C/C++ Compiler Package Application Note: [Introduction guide] Sample File Guide for SH-3, SH-4, and SH-4A (REJ06J0012)

- Development Tool Manuals
  Application Note: Flash Memory Download Program for the E10A-USB Emulator (REJ10J1221)

  User's Manual: SuperH RISC engine C/C++ Compiler, Assembler, Optimizing Linkage Editor Compiler Package V.9.01 (REJ10J1571)

The most up-to-date versions of the documents are available on the Renesas Technology Website.

## Website and Support

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

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

## Revision Record

| | | Description | |
|---|---|---|---|
| Rev. | Date | Page | Summary |
| 1.00 | Mar.27.09 | — | First edition issued |
| 2.00 | Dec.24.09 | 3 | The content of compiler options is corrected. |
| | | 6 | The allocation address is corrected. |
| | | | |
| | | | |
| | | | |

## Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.