HITACHI SEMICONDUCTOR TECHNICAL UPDATE

Classification of Production	Development Environment		No	TN-OS*-066A/E	Rev	1
THEME	HI7750/4 Notes on using the FPU	Classification of Information	1. Spec change 2. Supplement of Documents (3.) Limitation of Use 4. Change of Mask 5. Change of Production Line			
PRODUCT NAME	HS0775ITI41SRE, HS0775ITI41SRE-E, HS0775ITI41SRB, HS0775ITI41SRB-E, HS0775ITI41SRS, HS0775ITI41SRS-E	Lot No. V1.00r1, V1.01r1, V1.0Ar1, V1.0Br1, V1.0Cr1	Reference Documents	HI7000/4 series User's Manual ADE-702-248 (Rev.1.0)	Effective Date Forever	

When using the FPU on the HI7750/4, please note the contents on the following document.

[Attached document]

" Notes on using the FPU on the HI7750/4" (HI7750/4-NOTE-FPU-021105(E))"

HI7750/4-NOTE-FPU-021105(E)

Notes on Using the FPU on the HI7750/4

Read this document carefully before using the FPU, and take note of the points on usage described herein.

System Software Development Dept.
Semiconductor & Integrated Circuits
Hitachi, Ltd.

1. Task and Task Exception Processing Routine

1.1 Attributes TA_COP1, TA_COP2

The SH-4 has two FPU register banks, 0 and 1. Attributes TA_COP1 and TA_COP2 select use of FPU bank 0 and FPU bank 1, respectively.

Specify attributes TA_COP1 and TA_COP2 as described in Table 1. The FPU register bank at initiation is 0 (FPSCR.FR = 0).

Table 1 Specifying Attributes TA_COP1, TA_COP2

Case	Attribute	Remarks
Matrix calculation (both FPU register banks used)	TA_COP1|TA_COP2	
Floating-point calculation	TA_COP1	The normal floating-point calculation uses a single FPU register bank. Specification of TA_COP2 is not recommended because FPSCR.FR = 1 must be set at the top of the entry function of the task and task exception processing routine.
No floating-point calculation	No specification required	

1.2 Compiler Options

The FPSCR at initiation is shown in 4.1, States on the Initiation of Tasks and Handlers.

When a floating-point calculation is executed, the FPU will not operate normally with these initial values and some compiler options. When any of the options in Table 2 is selected, the corresponding bit of FPSCR must be set to the value shown. This setting is made at the top of the entry function of the task and of the task exception processing routine.

Table 2 Value to be Set in FPSCR

Bits in FPSCR	Compiler Option		Value to be Set
Precision mode (FPSCR.PR)	FPU option	Double	1
		Other than double	(Not required)
Denormalization mode (FPSCR.DN)	Denormalize option	OFF	(Not required)
		ON	0
Rounding mode (FPSCR.RM)	Round option	Zero	(Not required)
		Nearest	B'00

The following shows an example for setting FPSCR in the conditions below:

[Compiler Options]

- FPU =Double
- Denormalize = ON
- Round = Nearest

```
#include <machine.h>   /* Included to use built-in function set_fpscr(). */
#define INI_FPSCR 0x00080000   /* PR=1, DN=0, SZ=0, RM=B'00 */
#pragma noregsave(Task)
void Task(VP_INT exinf)
{
    set_fpscr(INI_FPSCR);    /* Sets FPR at the top of function.  */
    /* Task processing */
    ext_tsk();
}
```

2. Interrupt Handler, CPU Exception Handler, Time Event Handler, and Initialization Routine

When a floating-point calculation is executed by these handlers, make the required initial settings for the FPSCR at the start of these handlers, with reference to section 4.14, Using the FPU in Programs (HI7750/4), of the HI7000/4 Series User's Manual (ADE-702-248). When leaving the handler, the FPSCR needs not be restored.

Table 3 shows values to be set in the bits of FPSCR.

When no floating-point calculation is to be executed, this initial setting of FPSCR is not necessary.

Table 3 Value to be Set in FPSCR

Bits in FPSCR	Compiler Option		Value to be Set
Precision mode (FPSCR.PR)	FPU option	Double	1
		Other than double	0
Denormalization mode (FPSCR.DN)	Denormalize option	OFF	1
		ON	0
Rounding mode (FPSCR.RM)	Round option	Zero	B'01
		Nearest	B'00
Transfer size mode (FPSCR.SZ)			0
FPU register bank (FPSCR.FR)			0
Other bits of FPSCR			0

3. Extended Service Call Routine

3.1 Execution of Floating-Point Calculation

The compiler handles the issuing of extended service calls as the calling of functions. However, when the caller and extended service call routine have been defined in different linkage units, different compiler options may have been used. In this case, make any required initial settings for FPSCR at the top of the entry function to the extended service call routine, as specified in Table 3. Restore FPSCR to its previous state on leaving the extended service call routine.

3.2 No Execution of Floating-Point Calculation

As a basic principle, set the same values of these compiler options in compiling the extended service call routine and the caller:

- FPU option
- FPSCR option
- Denormalize option
- Round option

In particular, when the compiler options for the extended service call routine are

- FPU option not specified (MIX)
- FPSCR option = Aggressive

and different compiler options were used with the caller and the extended service call routine, the state of the FPSCR register must be saved at the top of the extended service call routine and restored at the end of that routine.

An example is given below:

```
#include <machine.h>   /* Included to use built-in function set_fpscr(). */
ER ExtendedSVCRoutine(VP_INT par1)
{
    UW old_fpscr;
    old_fpscr = get_fpscr(); /* Saves FPSCR. */
    /* Processing of the extended service-call routine */
    set_fpscr(old_fpscr);     /* Restores FPSCR.  */
    return E_OK;
}
```

4. Information for Reference

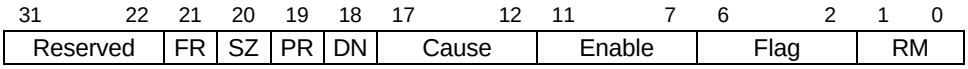
4.1 States on the Initiation of Tasks and Handlers

Table 4 shows the FPSCR states on the initiation of tasks and handlers

Table 4 States on the Initiation of Tasks and Handlers

State at Initiation		Task and Task Exception Processing Routine	Extended Service Call Routine	Interrupt and Time Event Handlers, and Initialization Routine	CPU Exception Handlers (Including TRAPA)
Value of FPSCR		H'00040001	Same as before the extended service call was issued	Undefined	Same as before the exception occurred
	Precision mode (FPSCR.PR)	Single precision (0)			
	Denormalization mode (FPSCR.DN)	Handled as 0 (1)			
	Rounding mode (FPSCR.RM)	Rounded to 0 (B'01)			
	Transfer size mode (FPSCR.SZ)	32 bits (0)			
	FPU regsiter bank(FPSCR.FR)	Bank 0 (0)			
	Other bits of FPSCR	0			

4.2 FPSCR Structure of SH-4



bit		Meaning		
21	FR	FPU register bank	0	Bank 0
			1	Bank 1
20	SZ	Transfer size mode	0	The data size of the FMOV instruction is 32 bits.
			1	The data size of the FMOV instruction is a 32-bit register pair (64 bits).
19	PR	Precision mode	0	Single precision
			1	Double precision
18	DN	Denormalization mode	0	A denormalized number is treated as such.
			1	A denormalized number is treated as zero.
17-12	Cause	FPU exception cause field		
11-7	Enable	FPU exception enable field		
6-2	Flag	FPU exception flag field		
1,0	RM	Rounding mode	B'00	Round to Nearest
			B'01	Round to Zero

4.3 Handling by the Compiler

This section explains handling of the FPU by V5.1, V6, and V7.1 of the compiler. The compiler never generates any object code to change the FPSCR when "Single" or "Double" has been specified as the FPU option.

(1) FPSCR.PR (Precision mode)

Table 5 Handling of the FPSCR.PR Bit by the Compiler

Compiler Option		Precision Mode Assumed by the Compiler on Entry to Functions (FPSCR.PR Bit)*1	Precision Mode at the End of the Function*2	Remarks
FPU option	FPSCR option *3			
Single	(Specification disabled)	Single precision (0)	Single precision (0)	The compiler does not generate any object code to change the PR bit.
Double	(Specification disabled)	Double precision (1)	Double precision (1)	
No specification (Mix)	Safe	Single precision (0)	Single precision (0)	
	Aggressive	Single precision (0)	Undefined	

Note: *1 The compiler assumes this precision mode in generating code at the top of the function.
*2 The compiler generates code to select this precision mode at the end of the function.
*3 Compiler V5.1 does not support this FPSCR option; treatment is the same as 'aggressive'.

(2) FPSCR.DN (Denormalization mode)

Table 6 Handling of the FPSCR.DN Bit by the Compiler

Compiler Option	Denormalization Mode Assumed by the Compiler (FPSCR.DN Bit)*	Remarks
Denormalize Option		
OFF	A denormalized number is treated as zero. (1)	The compiler does not generate any object code to change the DN bit.
ON	A denormalized number is treated as such. (0)	

Note: *1 The compiler assumes this denormalization mode in generating code at the top of the function.

(3) FPSCR.RM (Rounding mode)

Table 7 Handling of the FPSCR.RM Bits by the Compiler

Compiler Option	Rounding Mode Assumed by the Compiler (FPSCR.RM Bit)*	Remarks
Round Option		
Zero	Round to Zero (B'01)	The compiler does not generate any object code to change the RM bits.
Nearest	Round to Nearest (B'00)	

Note: *1 The compiler assumes this rounding mode in generating code at the top of the function.

(4) FPSCR.SZ (transfer size mode)

The compiler always assumes SZ = 0 (the data size of the FMOV instruction is 32 bits.) and does not generate any object code to change the SZ bit.

(5) FPSCR.FR (FPU register bank)

The compiler does not generate any object code to change the FR bit.

However, in the built-in functions st_ext() and ld_ext(), the FR bit is temporarily changed within these function. The value of the FR bit on return from these function is the same as the value when the function was called.