4.6 Floating point unit (FPU)

The Cortex-M4F FPU implements the FPv4-SP floating-point extension.

The FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed xxxxx-point and floating-point data formats, and floating-point constant instructions.

The FPU provides floating-point computation functionality that is compliant with the ANSI/IEEE standard 754-2008, IEEE standard for Binary Floating-Point Arithmetic, referred to as the IEEE 754 standard.

The FPU contains 32 single-precision extension registers, which you can also access as 16 doubleword registers for load, store, and move operations.

Table 55 shows the floating-point system registers in the Cortex-M4F system control block (SCB). The base address of the additional registers for the FP extension is 0xE000 ED00.

Table 55. Cortex-M4F floating-point system registers

Address	Name	Type	Reset	Description
0xE000ED88	CPACR	RW	0x00000000	*Section 4.6.1: Coprocessor access control register (CPACR) on page 252*
0xE000EF34	FPCCR	RW	0xC0000000	*Section 4.6.2: Floating-point context control register (FPCCR) on page 252*
0xE000EF38	FPCAR	RW	-	*Section 4.6.3: Floating-point context address register (FPCAR) on page 254*
0xE000EF3C	FPDSCR	RW	0x00000000	*Section 4.6.5: Floating-point default status control register (FPDSCR) on page 256*
-	FPSCR	RW	-	*Section 4.6.4: Floating-point status control register (FPSCR) on page 254*

The following sections describe the floating-point system registers whose implementation is specific to this processor.

Note: For more details on the IEEE standard and floating-point arithmetic (IEEE 754), refer to the AN4044 Application note. Available from website www.st.com.

4.6.1 Coprocessor access control register (CPACR)

Address offset (from SCB): 0x88

Reset value: 0x0000000

Required privilege: Privileged

The CPACR register specifies the access privileges for coprocessors.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CP11		CP10		Reserved			
								rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Bits 31:24 Reserved. Read as Zero, Write Ignore.

Bits 23:20 **CPn:** [2n+1:2n] for n values 10 and 11. Access privileges for coprocessor n. The possible values of each field are:

0b00: Access denied. Any attempted access generates a NOCP UsageFault.
0b01: Privileged access only. An unprivileged access generates a NOCP fault.
0b10: Reserved. The result of any access is Unpredictable.
0b11: Full access.

Bits 19:0 Reserved. Read as Zero, Write Ignore.

4.6.2 Floating-point context control register (FPCCR)

Address offset: 0x04

Reset value: 0xC000000

Required privilege: Privileged

The FPCCR register sets or returns FPU control data.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ASPEN	LSPEN	Reserved													
rw	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							MONRDY	Reserved	BFRDY	MMRDY	HFRDY	THREAD	Reserved	USER	LSPACT
							rw		rw	rw	rw	rw		rw	rw

Bit 31 **ASPEN**: Enables CONTROL<2> setting on execution of a floating-point instruction. This results in automatic hardware state preservation and restoration, for floating-point context, on exception entry and exit.

0: Disable CONTROL<2> setting on execution of a floating-point instruction.

1: Enable CONTROL<2> setting on execution of a floating-point instruction.

Bit 30 **LSPEN**:

0: Disable automatic lazy state preservation for floating-point context.

1: Enable automatic lazy state preservation for floating-point context.

Bits 29:9 Reserved.

Bit 8 **MONRDY**:

0: DebugMonitor is disabled or priority did not permit setting MON_PEND when the floating-point stack frame was allocated.

1: DebugMonitor is enabled and priority permits setting MON_PEND when the floating-point stack frame was allocated.

Bit 7 Reserved.

Bit 6 **BFRDY**:

0: BusFault is disabled or priority did not permit setting the BusFault handler to the pending state when the floating-point stack frame was allocated.

1: BusFault is enabled and priority permitted setting the BusFault handler to the pending state when the floating-point stack frame was allocated.

Bit 5 **MMRDY**:

0: MemManage is disabled or priority did not permit setting the MemManage handler to the pending state when the floating-point stack frame was allocated.

1: MemManage is enabled and priority permitted setting the MemManage handler to the pending state when the floating-point stack frame was allocated.

Bit 4 **HFRDY**:

0: Priority did not permit setting the HardFault handler to the pending state when the floating-point stack frame was allocated.

1: Priority permitted setting the HardFault handler to the pending state when the floating-point stack frame was allocated.

Bit 3 **THREAD**:

0: Mode was not Thread Mode when the floating-point stack frame was allocated.

1: Mode was Thread Mode when the floating-point stack frame was allocated.

Bit 2 Reserved.

Bit 1 **USER**:

0: Privilege level was not user when the floating-point stack frame was allocated.

1: Privilege level was user when the floating-point stack frame was allocated.

Bit 1 **LSPACT**:

0: Lazy state preservation is not active.

1: Lazy state preservation is active. floating-point stack frame is allocated but saving state to it is deferred.

4.6.3 Floating-point context address register (FPCAR)

Address offset: 0x08

Reset value: 0x0000000

Required privilege: Privileged

The FPCAR register holds the location of the unpopulated floating-point register space allocated on an exception stack frame.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:3]													Reserved		
rw															

Bits 31:3 ADDRESS: Location of unpopulated floating-point register space allocated on an exception stack frame.

Bits 2:0 Reserved. Read as Zero, Writes Ignored.

4.6.4 Floating-point status control register (FPSCR)

Address offset: Not mapped

Reset value: 0x0000000

Required privilege: Privileged

The FPSCR register provides all necessary user level control of the floating-point system.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
N	Z	C	V	Reserved	AHP	DN	FZ	RMode		Reserved					
rw	rw	rw	rw		rw	rw	rw	rw	rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								IDC	Reserved		IXC	UFC	OFC	DZC	IOC
								rw			rw	rw	rw	rw	rw

Bit 31 **N**: Negative condition code flag. Floating-point comparison operations update these flags. For more details on the result, refer to *Table 56*.

0: Operation result was positive, zero, greater than, or equal.

1: Operation result was negative or less than.

Bit 30 **Z**: Zero condition code flag. Floating-point comparison operations update these flags. For more details on the result, refer to *Table 56*.

0: Operation result was not zero.

1: Operation result was zero.

Bit 29 **C**: Carry condition code flag. Floating-point comparison operations update these flags. For more details on the result, refer to *Table 56*.

0: Add operation did not result in a carry bit or subtract operation resulted in a borrow bit.

1: Add operation resulted in a carry bit or subtract operation did not result in a borrow bit.

Bit 28 **V**: Overflow condition code flag. Floating-point comparison operations update this flag. For more details on the result, refer to *Table 56*.
0: Operation did not result in an overflow
1: Operation resulted in an overflow.

Bit 27 Reserved.

Bit 26 **AHP**: Alternative half-precision control bit:
0: IEEE half-precision format selected.
1: Alternative half-precision format selected.

Bit 25 **DN**: Default NaN mode control bit:
0: NaN operands propagate through to the output of a floating-point operation.
1: Any operation involving one or more NaNs returns the Default NaN.

Bit 24 **FZ**: Flush-to-zero mode control bit:
0: Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.
1: Flush-to-zero mode enabled.

Bits 23:22 **RMode**: Rounding Mode control field. The specified rounding mode is used by almost all floating-point instructions:
0b00: Round to nearest (RN) mode
0b01: Round towards plus infinity (RP) mode
0b10: Round towards minus infinity (RM) mode
0b11: Round towards zero (RZ) mode.

Bit 21:8 Reserved.

Bit 7 **IDC**: Input denormal cumulative exception bit. Cumulative exception bit for floating-point exception.
1: Indicates that the corresponding exception occurred since 0 was last written to it.

Bit 6:5 Reserved

Bit 4 **IXC**: Inexact cumulative exception bit. Cumulative exception bit for floating-point exception.
1: Indicates that the corresponding exception occurred since 0 was last written to it.

Bit 3 **UFC**: Underflow cumulative exception bit. Cumulative exception bit for floating-point exception.
1: Indicates that the corresponding exception occurred since 0 was last written to it.

Bit 2 **OFC**: Overflow cumulative exception bit. Cumulative exception bit for floating-point exception.
1: Indicates that the corresponding exception occurred since 0 was last written to it.

Bit 1 **DZC**: Division by zero cumulative exception bit. Cumulative exception bit for floating-point exception. 1: Indicates that the corresponding exception occurred since 0 was last written to it.

Bit 0 **IOC**: Invalid operation cumulative exception bit. Cumulative exception bit for floating-point exception. 1: Indicates that the corresponding exception occurred since 0 was last written to it.

Table 56. Effect of a Floating-point comparison on the condition flags

Comparison result	N	Z	C	V
Equal	0	1	1	0
Less than	1	0	0	0
Greater than	0	0	1	0
Unordered	0	0	1	1

4.6.5 Floating-point default status control register (FPDSCR)

Address offset: 0x0C

Reset value: 0x0000000

Required privilege: Privileged

The FPDSCR register holds the default values for the floating-point status control data.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					AHP	DN	FZ	RMode		Reserved					
					rw	rw	rw	rw	rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Bits 31:27 Reserved, must be kept cleared.

Bit 26 **AHP:** Default value for FPSCR.AHP

Bit 25 **DN:** Default value for FPSCR.DN

Bit 24 **FZ:** Default value for FPSCR.FZ

Bits 23:22 **RMode:** Default value for FPSCR.RMode

Bits 21:0 Reserved, must be kept cleared.

4.6.6 Enabling the FPU

The FPU is disabled from reset. You must enable it before you can use any floating-point instructions.

The example shows an example code sequence for enabling the FPU in both privileged and user modes. The processor must be in privileged mode to read from and write to the CPACR.

Example

```
                ; CPACR is located at address 0xE000ED88
LDR.W   R0, =0xE000ED88
                ; Read CPACR
LDR     R1, [R0]
                ; Set bits 20-23 to enable CP10 and CP11 coprocessors
ORR     R1, R1, #(0xF << 20)
                ; Write back the modified value to the CPACR
STR     R1, [R0]; wait for store to complete
DSB
;reset pipeline now the FPU is enabled
ISB
```

4.6.7 Enabling and clearing FPU exception interrupts

The FPU exception flags are generating an interrupt through the interrupt controller. The FPU interrupt is globally controlled through the interrupt controller.

A mask bit is also provided in the System Configuration Controller (SYSCFG), allowing to enable/disable individually each FPU flag interrupt generation.

Note: In STM32F4xx devices there is no individual mask and the enable/disable of the FPU interrupts is done at interrupt controller level. As it occurs very frequently, the IXC exception flag is not connected to the interrupt controller in these devices , and cannot generate an interrupt. If needed, it must be managed by polling.

Clearing the FPU exception flags depends on the FPU context save/restore configuration:

- No floating-point register saving: when Floating-point context control register (FPCCR) Bit 30 LSPEN=0 and Bit 31 ASPEN=0.
  You must clear interrupt source in Floating-point Status and Control Register (FPSCR).
  Example:

```
register uint32_t fpscr_val = 0;
fpscr_val  = __get_FPSCR();
{ check exception flags }
fpscr_val &= (uint32_t)~0x8F; // Clear all exception flags
__set_FPSCR(fpscr_val);
```

- Lazy save/restore: when Floating-point context control register (FPCCR)
  Bit 30 LSPEN=1 and Bit 31 ASPEN=X.
  In the case of lazy floating-point context save/restore, a dummy read access should be made to Floating-point Status and Control Register (FPSCR) to force state preservation and FPSCR clear.
  Then handle FPSCR in the stack.
  Example:

```
register uint32_t fpscr_val = 0;
register uint32_t reg_val = 0;
reg_val =  __get_FPSCR();        //dummy access
fpscr_val=*(__IO uint32_t*)(FPU->FPCAR +0x40);
{ check exception flags }
fpscr_val &= (uint32_t)~0x8F ;   // Clear all exception flags
*(__IO uint32_t*)(FPU->FPCAR +0x40)=fpscr_val;
__DMB() ;
```

- Automatic floating-point registers save/restore: when Floating-point context control register (FPCCR)
  Bit 30 LSPEN=0 and Bit 31 ASPEN=1.
  In case of automatic floating-point context save/restore, a read access should be made to Floating-point Status and Control Register (FPSCR) to force clear.
  Then handle FPSCR in the stack.
  Example:

```
// FPU Exception handler
void FPU_ExceptionHandler(uint32_t lr, uint32_t sp)
{
register uint32_t fpscr_val;
   if(lr == 0xFFFFFFE9)
   {
```

```
    sp = sp + 0x60;
  }
  else if(lr == 0xFFFFFFED)
  {
    sp = __get_PSP() + 0x60 ;
  }
fpscr_val = *(uint32_t*)sp;
{ check exception flags }
fpscr_val &= (uint32_t)~0x8F ;   // Clear all exception flags
*(uint32_t*)sp = fpscr_val;
__DMB() ;
}
// FPU IRQ Handler
void __asm FPU_IRQHandler(void)
{
IMPORT  FPU_ExceptionHandler
MOV   R0, LR             // move LR to R0
MOV   R1, SP             // Save SP to R1 to avoid any modification to
                         // the stack pointer from FPU_ExceptionHandler
VMRS R2, FPSCR           // dummy read access, to force clear
B     FPU_ExceptionHandler
BX    LR
}
```