# FPU HCI

## Specifications

This floating-point unit (FPU) supports three different representations of floating-point numbers according to IEEE754-2019 standard which are binary32, decimal representation-binary format and decimal representation-decimal format, the following arithmetic operations are supported for each of the different representations, addition, subtraction and multiplication between operand A and operand B and fusedMultiplyAdd represented as follows . SIMD instruction is also supported for any of the supported operations with a maximum number of 16 similar operations per SIMD instruction.

## Abbreviations

* **Reserved**: These registers/bits are reserved and should be set to zero
* **RO-Read Only**: If a register/bit is read only, this means that only the FPU can write into it, writes by the software have no effect and reads by the FPU return zeros.
* **WO-Write Only**: If a register/bit is write only, this means that only the software can write into it, writes by the FPU have no effect and reads by the software return zeros.
* **R/W-Read/Write**: If a register/bit is read/write, this means that both the software and the FPU can write into it and read from it. Note that individual bits in R/W registers may be RO or WO.
* **Single operation instruction**: instruction where the SIMD of thecontrol bit of the FPU command register is set to zero.
* **SIMD instruction**: instruction where the SIMD of thecontrol bit of the FPU command register is set to one.

## Memory-mapped FPU Host Controller Registers

|  |  |  |  |
| --- | --- | --- | --- |
| **Configuration offset** | **Register Set** | **Number of registers** | **Register Access** |
| 000 | FPU Command register | 1 | R/W |
| 0x004-0x00C | Reserved | 3 | -- |
| 0x010-0x04C | Operand A | 16 | WO |
| 0x050-0x08C | Operand B | 16 | WO |
| 0x090-0x0CC | Operand C | 16 | WO |
| 0x0D0-0x10C | Reserved | 16 | -- |
| 0x110-0x11C | FPU Status registers | 4 | R/W |
| 0x120-0x12C | Reserved | 4 | -- |
| 0x130-0x16C | Output | 16 | RO |
| 0x170-0x17C | Reserved | 4 | -- |

## FPU Command register:

|  |  |
| --- | --- |
| **Bit** | **Description** |
| 21-31 | **Reserved** |
| 18-20 | **Number of SIMD operations – WO.**   * Default 000b (1 operation). This field identifies the number of SIMD instruction operations, the FPU checks this field if it’s a SIMD instruction. * The FPU can carry out a maximum of 16 similar operations (value = 111b) at each SIMD instruction. |
| 17 | **SIMD – WO.**   * This control bit is used by the software to tell the FPU whether this instruction is a single operation instruction or a SIMD instruction, if the software sets this bit to one then it’s a SIMD instruction else if it sets it to zero then this is a single operation instruction. |
| 13-16 | **Reserved** |
| 11-12 | **Operation – WO.**   * Default 00b.This field identifies which operation will be performed. * Values mean: * 00b Addition * 01b Subtraction * 10b Multiplication * 11b Fused multiply add |
| 7-10 | **Reserved** |
| 5-6 | **Floating-point format – WO.**   * Default 00b.This field identifies which floating-point format is to be used. * Values mean: * 00b Binary32 (Single-precision) * 01b Decimal representation – Binary format * 10b Decimal representation – Decimal format * 11b Reserved |
| 4 | **Reserved** |
| 3 | **Interrupt Enable – WO.**   * This control bit is set to one by the software to tell the FPU to issue an interrupt in the Interrupt bit of the FPU Status register (register 0x110-bit 0) when it finishes an operation. |
| 2 | **Doorbell – R/W.**   * This control bit is set to one by software to tell the FPU that there is a new operation. * When the FPU finishes the operation, it sets it to zero. |
| 1 | **FPU Enable – WO.**   * This control bit is used by software to enable the FPU. * The FPU executes the operations as long as this bit is one. When the software sets this bit to zero, the FPU completes the current operation and then halts until the software sets this bit to one again. |
| 0 | **FPU Reset – R/W.**   * This control bit is used by software to reset the FPU. * When software writes a one to this bit, the FPU terminates any operation in progress. * This bit is set to zero by the FPU when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. |

## FPU Status registers:

### Register (0x110)

|  |  |
| --- | --- |
| **Bit** | **Description** |
| 6-31 | **Reserved** |
| 5 | **Inexact flag – RO.**   * Default 0b. This control bit is set to one by the FPU when an operation delivers a numerical result that signal no other exception and its rounded result differs from what would have been computed were both exponent range and precision unbounded. * More details in IEEE754-2019 standard section 7.6 |
| 4 | **Underflow flag – RO.**   * Default 0b. This control bit is set to one by the FPU when a tiny non-zero result is detected. * More details in IEEE754-2019 standard section 7.5 |
| 3 | **Overflow flag – RO.**   * Default 0b. This control bit is set to one by the FPU if and only if the destination format’s largest finite number is exceeded in magnitude by what would have been the rounded floating-point result were the exponent range unbounded. * More details in IEEE754-2019 standard section 7.4 |
| 2 | **Division by zero flag– RO.**   * **Reserved** and set to zero as the current FPU doesn’t support division or logarithmic operations. * More details in IEEE754-2019 standard section 7.3 |
| 1 | **Invalid operation flag – RO.**   * Default 0b. This control bit is set to one by the FPU if and only if there is no usefully definable result in the cases where the operands are invalid for the operation to be performed. * More details in IEEE754-2019 standard section 7.2 |
| 0 | **Interrupt– R/W.**   * This control bit is set to one by The FPU to tell the software that the operation has terminated. * In case that the Interrupt Enable bit of the FPU (bit 4) is set to one, the software notices when the FPU sets this bit to one and it sets it to zero again. |

Note: The five flags (bits 1 -5) in this register are the output flags in case of a single operation instruction, in case of a SIMD instruction they are the output flags of the first SIMD instruction operation.

### Register (0x114)

|  |  |
| --- | --- |
| **Bit** | **Description** |
| 31 | **Reserved** |
| 16-30 | **SIMD Division by zero flags– RO.**   * **Reserved** and set to zero as the current FPU doesn’t support division or logarithmic operations. |
| 15 | **Reserved** |
| 0-14 | **SIMD Invalid operation flags – RO.**   * Each of these 15 bits have a default value of 0b, in case of a SIMD instruction, they are the invalid operation output flags, as explained for bit 1 in FPU status register (0x110), of the second to the SIMD instruction operations in order. |

### Register (0x118)

|  |  |
| --- | --- |
| **Bit** | **Description** |
| 31 | **Reserved** |
| 16-30 | **SIMD Underflow flags– RO.**   * Each of these 15 bits have a default value of 0b, in case of a SIMD instruction, they are the underflow output flags, as explained for bit 4 in FPU status register (0x110), of the of the second to the SIMD instruction operations in order. |
| 15 | **Reserved** |
| 0-14 | **SIMD Overflow flags– RO.**   * Each of these 15 bits have a default value of 0b, in case of a SIMD instruction, they are the overflow output flags, as explained for bit 3 in FPU status register (0x110), of the of the second to the SIMD instruction operations in order. |

### Register (0x11C)

|  |  |
| --- | --- |
| **Bit** | **Description** |
| 15-31 | **Reserved** |
| 0-14 | **SIMD Inexact flags– RO.**   * Each of these 15 bits have a default value of 0b, in case of a SIMD instruction, they are the inexact output flags, as explained for bit 5 in FPU status register (0x110), of the of the second to the SIMD instructions operations in order. |

## Operands A, B & C and Output registers:

Each operand has 16 registers and so does the output, the first register of each operand is used in single operations and the result is written in the first output register, the different operations are carried out as follows:

* Addition:
* Subtraction:
* Multiplication:
* FusedMultiplyAdd:

In case of SIMD instructions, the Number of SIMD operations field of the FPU command register determines how many similar operations are carried out which also determines the number of registers of each operand and the output that is to be used, operations are carried out on the registers of each operand and the output in order, to illustrate how this works with a simple example, given that the operation is addition and the number of SIMD operations is three, the addition SIMD operations are carried out as follows: