uM-FPU Floating Point Coprocessor

Datasheet

Micromega Corporation

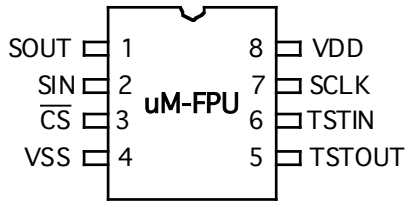
Introduction

The uM-FPU is a 32-bit floating point coprocessor that can be easily interfaced with microcontrollers to provide support for 32-bit IEEE 754 floating point operations and long integer operations. The uM-FPU is easy to connect using an SPI compatible interface.

uM-FPU Features

- 8-pin integrated circuit.
- No additional external components
- SPI compatible interface
- Sixteen 32-bit general purpose registers for storing floating point or long integer values
- Five 32-bit temporary registers with support for nested calculations (i.e. parenthesis)
- Floating Point Operations
  - Set, Add, Subtract, Multiply, Divide
  - Sqrt, Log, Log10, Exp, Exp10, Power, Root
  - Sin, Cos, Tan
  - Asin, Acos, Atan, Atan2
  - Floor, Ceil, Round, Min, Max, Fraction
  - Negate, Abs, Inverse
  - Convert Radians to Degrees
  - Convert Degrees to Radians
  - Compare, Status
- Long Integer Operations
  - Set, Add, Subtract, Multiply, Divide, Unsigned Divide
  - Negate, Abs
  - Compare, Unsigned Compare, Status
- Conversion Functions
  - Convert 8-bit and 16-bit integers to floating point
  - Convert 8-bit and 16-bit integers to long integer
  - Convert long integer to floating point
  - Convert floating point to long integer
  - Convert floating point to ASCII
  - Convert floating point to formatted ASCII
  - Convert long integer to ASCII
  - Convert long integer to formatted ASCII
  - Convert ASCII to floating point
  - Convert ASCII to long integer

Pin Diagram and Pin Description

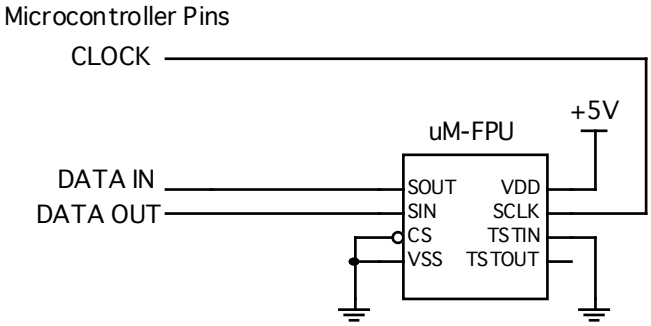


Pin	Name	Type	Description
1	SOUT	Output	SPI Output
2	SIN	Input	SPI Input
3	/CS	Input	Chip Select
4	VSS	Power	Ground
5	TSTOUT	Output	Test Output
6	TSTIN	Input	Test Input
7	SCLK	Input	SPI Clock
8	VDD	Power	Supply Voltage

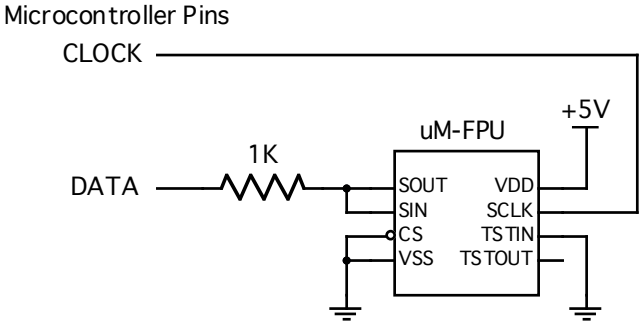
Connecting the uM-FPU

The uM-FPU can be connected using either a 2-wire or 3-wire interface depending on the capabilities of the microcontroller. The 3-wire connection uses separate data input and data output pins on the microcontroller, while the 2-wire connection uses a single bidirectional pin for both data input and data output. The Chip Select (/CS) pin on the uM-FPU is normally tied to ground.

3-wire Connection



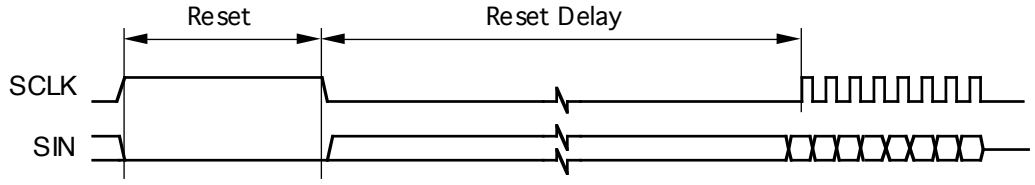
2-wire Connection



Resetting the uM-FPU

The uM-FPU should be reset at the beginning of every program to ensure that the microcontroller and the uM-FPU are synchronized. To cause a Reset, the SIN line must be Low while the SCLK line is held High for a minimum of 100 microseconds. A delay of 2 milliseconds is required after the Reset to ensure that the Reset is complete and the uM-FPU is ready to receive commands. All uM-FPU registers are reset to the special value NaN (Not a Number), which is equal to hexadecimal value 7FC00000.

Reset Timing Diagram



Item	Min	Max	Unit
Reset - SCLK Output High	100		usec
Reset - SIN Output Low	100		usec
Reset Delay	2		msec

Executing Instructions

The uM-FPU is configured as a Serial Peripheral Interconnect (SPI) slave device. Data is transmitted and received with the most significant bit (MSB) first using SPI mode 0.

SCLK is active High (idle state is Low)
Data latched on leading edge of SCLK
Data changes on trailing edge of SLCK
Data is transmitted most significant bit first

There are three types of instructions executed by the uM-FPU:

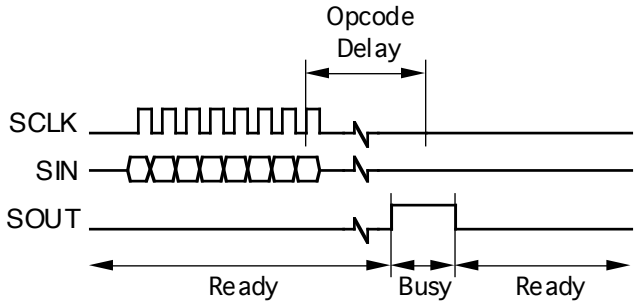
1. Opcode.
2. Opcode followed by an additional argument.
3. Opcode followed by a return value.

Opcodes are either one or two bytes in length. Some opcodes require an additional argument, which can be a byte value, word value (2 bytes), long value (4 bytes) or string value (up to 16 bytes terminated by a zero byte). Some opcodes have a return value, which can be a byte value, long value (4 bytes), or string value (up to 16 bytes terminated by a zero byte). Refer to Appendix A for a list of opcodes, additional arguments and return values.

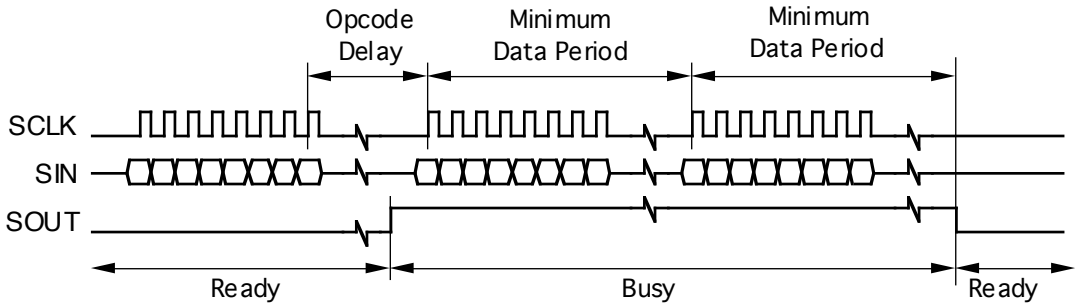
Before an instruction is sent to the uM-FPU, the SOUT line must be checked to ensure that the uM-FPU is ready. When the uM-FPU is ready to receive the next instruction, the SOUT pin is held Low by an internal 5.6K resistor. If the uM-FPU is still executing the last instruction, or the debug monitor is enabled and a Break has occurred, the SOUT pin is held High through an internal 5.6K pull-up resistor. The minimum data period must have elapsed since the last byte was transmitted before the SOUT status is checked.

The maximum SCLK frequency is 1 MHz, but there must be a delay after the first opcode byte, and a minimum data period between bytes. The opcode delay is measured from the rising edge of the last bit of the first opcode byte to the rising edge of the first bit of the next data byte. The minimum data period is measured from the rising edge of the first bit of one date byte to the rising edge of the next data byte. In the debug tracing is enabled, the opcode delay and minimum data period must be extended to accommodate the trace overhead (see timing chart below).

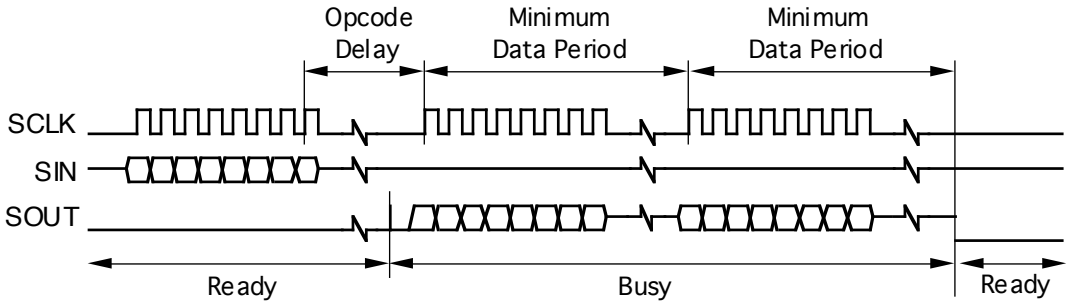
Instruction Timing Diagram
(Opcode)



Instruction Timing Diagram
(Opcode followed by an additional argument)



Instruction Timing Diagram
(Opcode followed by return value)



Item	Min	Max	Unit
SCLK Frequency		1	MHz
SCLK Output Low	0.5		usec
SCLK Output High	0.05	100	usec
Opcode Delay – Trace Enabled	100		usec
Opcode Delay – Trace Disabled	25		usec
Minimum Data Period – Trace Enabled	100		usec
Minimum Data Period – Trace Disabled	25		usec

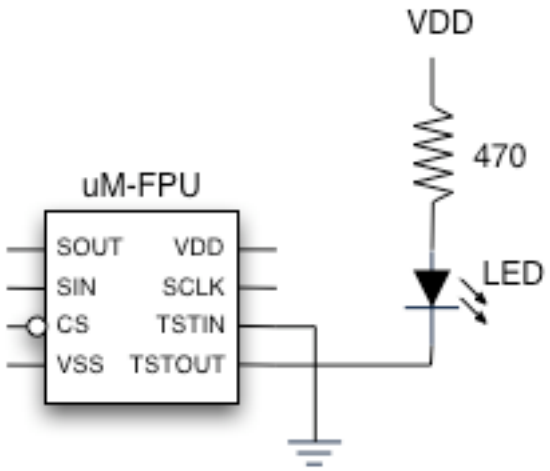
Using the TSTIN and TSTOUT Pins

The TSTIN and TSTOUT pins can be configured as an activity monitor or as a serial interface for the built-in debug monitor. The mode of operation is selected by the logic value of the TSTIN pin whenever the uM-FPU is reset. If the serial interface is not being used, the TSTIN pin should be tied to ground.

Activity Monitor

If the TSTIN pin is Low when the uM-FPU is reset, the TSTOUT pin is configured to generate an activity monitor signal. In this mode TSTOUT will be High when the uM-FPU is executing an instruction, and will be Low when it is idle. TSTOUT can be connected to an LED to provide a visual activity indicator, used as an input to an oscilloscope or logic analyzer during testing, or left unconnected.

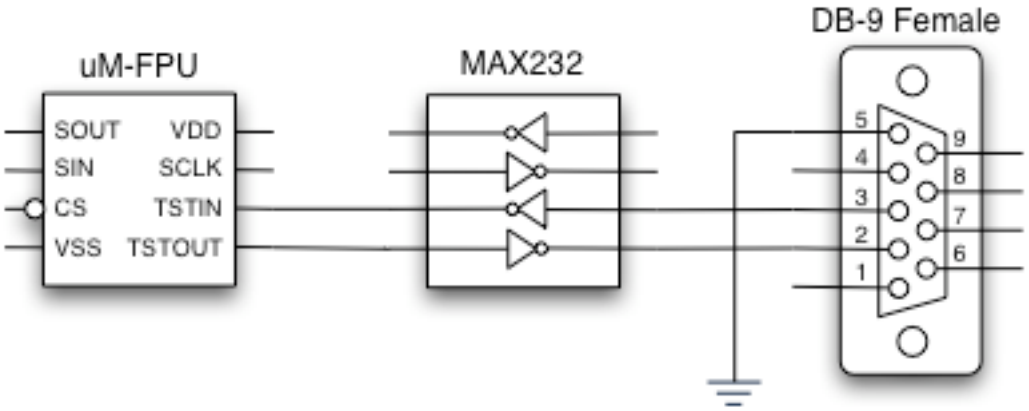
Activity Indicator



Serial Interface

If the TSTIN pin is High when the uM-FPU is reset, the TSTIN pin is configured as a serial input and the TSTOUT pin is configured as a serial output. The uM-FPU has a built-in debug monitor that is accessed through the TSTIN and TSTOUT serial connection. This enables the uM-FPU to be easily connected to a PC for debugging. The serial connection is configured as 57,600 baud, 8 bits, no parity, and one stop bit. There is no flow control. Note: The idle state of an RS-232 connection will assert a high level on the TSTIN pin, so provided the uM-FPU is connected to an active idle RS-232 port when the uM-FPU is reset, TSTIN and TSTOUT will be properly configured as a serial interface.

Serial Interface Connection



Debug Monitor

The built-in debug monitor provides support for tracing the execution of uM-FPU instructions, setting breakpoints for debugging, displaying the contents of uM-FPU registers, and programming stored functions.
Whenever the uM-FPU is reset and the serial interface is enabled the following message is displayed:

```
{RESET}
```

Commands are specified by typing a single uppercase or lowercase character followed by a return key. The command is not processed (or echoed) until the return key is pressed. Once the return key is pressed, the command prompt and command is displayed, and the command is executed. If the command is not recognized, a question mark will be displayed. Special commands are prefixed with a dollar sign. These commands are used to program the stored functions and to check the contents of the uM-FPU. They are not generally used when debugging a running application. The commands are listed below:

`B`	Break – stop execution after next opcode
`G`	Go – continue execution
`R`	Display Registers
`T`	Toggle Trace On/Off
`V`	Display Version
`/`	Comment
`$C`	Display Checksum
`$D`	Display Stored Functions
`$P`	Program Stored Functions

Break – stop execution after next opcode

The Break command is used to interrupt operation of the uM-FPU. The break will not occur until after the next opcode that is not a SELECTA or SELECTB is executed by the uM-FPU. The debug monitor displays the hex value of the last opcode executed and any additional data. Entering another break command, or simply pressing the return key, will single step to the next opcode. Entering the Go command will continue execution.

```
>B
 53                              (i.e. SET R3)
{BREAK}

>
 F6:0028                         (i.e. LOADWORD $0028)
{BREAK}
>

 80                              (i.e. FMUL R0)
{BREAK}

>
02 51                            (i.e. SELECT R2; SET R1)
{BREAK}
```

Go – continue execution

The Go command is used to continue normal execution after a Break command.

```
>G
```

Display Registers

The Register command displays the current contents of all uM-FPU registers.

```
>R
{A=R2, B=R0
 R0:41200000  R1:3F16791A  R2:3F16791A  R3:40490FDB
 R4:41200000  R5:41A00000  R6:7FC00000  R7:7FC00000
 R8:7FC00000  R9:7FC00000 R10:7FC00000 R11:7FC00000
R12:7FC00000 R13:7FC00000 R14:7FC00000 R15:7FC00000
 T1:7FC00000  T2:7FC00000  T3:7FC00000  T4:7FC00000
 T5:7FC00000}
```

Toggle Trace On/Off

The Trace command toggles the trace mode. The current state of the trace mode is displayed. When trace mode is on, each opcode that is executed by the uM-FPU is displayed.

```
>T
{TRACE ON}
 01 FA:55 F8" 1.00000" 53 F6:0002 80 95 E6 02 51 F6:000A 80 EA
F2 50 42:0000000A 01 FA:55 F8" 0.95106" 53 F6:0004 80 95 E6 02
51 F6:000A 80 EA F2 50 42:00000008 01 FA:55 F8" 0.80902" 53 F6:
0006 80 95 E6 02 51 F6:000A 80 EA F2 50 42:00000006 01 FA:55 F8
" 0.58779" 53 F6:0008 80 95 E6 02 51 F6:000A 80 EA F2 50 42:000
00003 01 FA:55 F8" 0.30902" 53 F6:000A 80 95 E6 02 51 F6:000A 8
0 EA F2 50 42:00000000 01 FA:55 F8" 0.00000" 53 F6:000C 80 95 E
6 02 51 F6:000A 80 EA F2 50 42:FFFFFFFD 01 FA:55 F8"-0.30902" 5
3 F6:000E 80 95 E6 02 51 F6:000A 80 EA F2 50 42:FFFFFFFA 01 FA:
55 F8"-0.58778" 53 F6:0010 80 95 E6 02 51 F6:000A 80 EA F2 50 4
2:FFFFFFF8 01 FA:55 F8"-0.80902" 53 F6:0012 80 95 E6 02 51 F6:0
00A 80 EA F2 50 42:FFFFFFF6 01 FA:55 F8"-0.95106" 53 F6:0014 80
 95 E6 02 51 F6:000A 80 EA F2 50 42:FFFFFFF6 01 FA:55 F8"-1.000
00" 53 F6:0016 80 95 E6 02 51 F6:000A 80 EA F2 50 42:FFFFFFF6 0
1 FA:55 F8"-0.95106" 53 F6:0018 80 95 E6 02 51 F6:000A 80 EA F2
 50 42:FFFFFFF8
>T
{TRACE OFF}
```

Display Version

The Version command displays the version string for the uM-FPU chip.

```
>V
uM-FPU V1.0
```

Comment

The comment command is used to insert short comment strings (up to six characters) in the debug session. This can be useful to provide some notations to refer to when analyzing debug results.

```
>/test1
```

Display Checksum

The Checksum command displays a checksum for the uM-FPU chip, excluding the stored function area. This can be used to confirm that the chip is valid.

```
>$C:000EC52B
```

Display Stored Functions

The Display command displays the contents of the stored function memory in Intel Hex format. This can be used to confirm the contents of the stored function memory.

```
>$D
:103C0000200C230725413648483F588B7B107F1041
:103C10008310874498079A0F9E64B70FBB080000CE
:103C2000000000000000000000000000000000FF
:103C3000000000000000000000000000000000FF
:103C4000000000000000000000000000000000FF
:103C5000000000000000000000000000000000FF
:103C6000000000000000000000000000000000FF
:103C7000000000000000000000000000000000FF
:103C800005FEF05006FEF05007FEF050073041F0CB
      .
      .
      .
:103F1000000000000000000000000000000000FF
:103F2000000000000000000000000000000000FF
:103F3000000000000000000000000000000000FF
```

Program Stored Functions

The Program command is used to program the stored function memory. Once in program mode, then uMFPU looks for valid Intel Hex format records of a specified format. The records must have an address between 0x0000 and 0x03C0 and start on a 64-byte boundary and the length must be 1 to 64 bytes. The data is not echoed, but an acknowledge character is sent after each record. The acknowledge characters are as follows:

`+`	The record was programmed successfully.
`F`	A format error occurred.
`A`	An address error occurred.
`C`	A checksum error occurred.
`P`	A programming error occurred.

An application program would normally be used to send the required data for the program command. (See documentation for the uM-FPU IDE application program.) To exit the program mode, an escape character is sent. The program command will reset the uM-FPU on exit.

```
>$P
{*** PROGRAM MODE ***}
+++

{RESET}
```

Debug Opcodes

There are several opcodes that are designed to work in conjunction with the debug monitor. If the serial interface was not selected by the TSTIN pin at the last Reset, these commands are NOPs. The opcodes are as follows:

BREAK

When this opcode is encountered a Break occurs. Execution will only resume when a command is issued to the debug monitor.

TRACEOFF

Turns the Trace mode OFF.

TRACEON
Turns the Trace mode ON. All opcodes will be traced on the debug terminal until the trace mode is turned off by TRACEOFF opcode or the trace mode is turned off using a debug command.

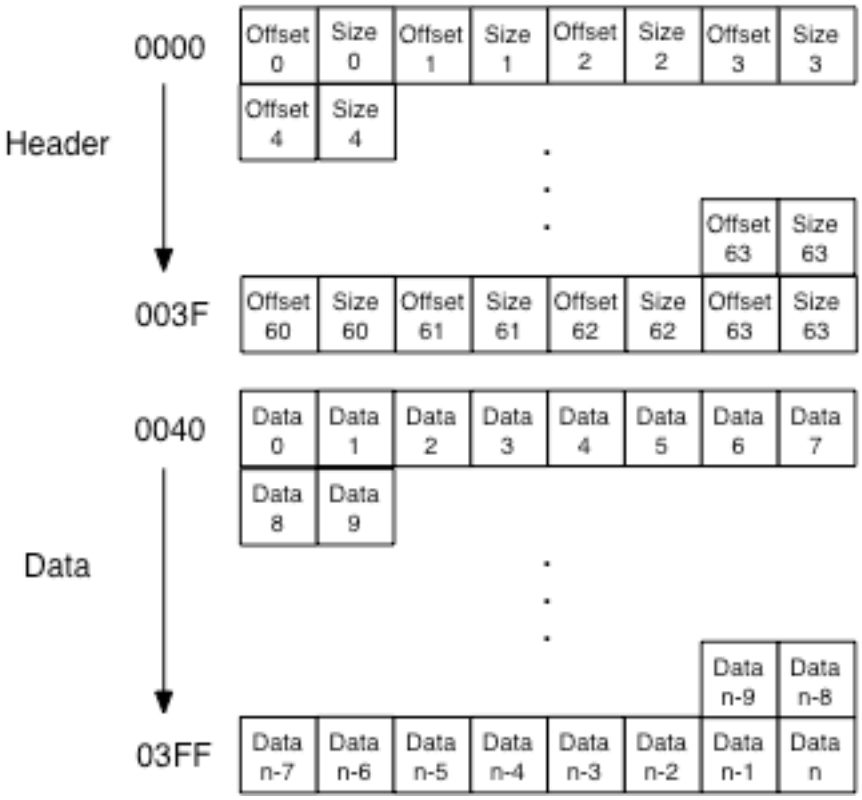
TRACESTR
Displays a trace string to the debug monitor output. This can be useful for keeping track a debug session. The trace strings are always output; they are not affected by the Trace mode.

Trace Buffer

Since there is no flow control on the serial interface, when the debug monitor is enabled, care must be taken to ensure that the trace buffer does not overflow. If the trace buffer overflows, the uM-FPU can lose synchronization with the microcontroller while waiting for space in the trace buffer. On many microcontrollers the normal code overhead allows ample time for the trace data, but on faster microcontrollers it may be necessary to insert a delay after sending or receiving data bytes. This is only required if the debug monitor is enabled. The trace buffer on the uM-FPU is 16 bytes in length. Each data byte that is transmitted or received produces about three trace characters on average and the serial interface runs at 57,600 baud, therefore approximately 521 microseconds is required to trace each data byte. The 16-byte trace buffer can accommodate short data bursts at a high data rate, but if the microcontroller can output data at a sustained rate faster than one byte every 521 microseconds, a suitable delay must be used to reduce the data rate while debugging.

Stored Functions

There are 1024 bytes of flash memory reserved on the uM-FPU for stored functions. Up to 64 stored functions can be defined by the user. Stored functions have the advantage of conserving space on the microcontroller and greatly reducing the communications overhead between the microcontroller and the uM-FPU. Opcodes FE00 through FE3F are used to execute the stored functions 0 through 63. The busy condition remains set while all of the opcodes in the stored function execute, so it appears to the microcontroller like a single instruction execution.
Function memory is divided into two sections: the header section and the data section. The header section is located at program address 0x0000 and consists of 64 pairs of bytes that specify the offset to the data section and the length of the stored function. The offset is specified as the address divided by 4, therefore all stored functions must start on an even 4-byte boundary.



Absolute Maximum Ratings

Parameter	Minimum	Typical	Maximum	Units
Storage Temperature	-55	-	+100	° Celsius
Ambient Temperature with Power Applied	-40	-	+70	° Celsius
Supply Voltage on VDD relative to VSS	-0.5	-	+6.0	V
DC Input Voltage VSS	-0.5	-	VDD+0.5	V
Maximum Current into any I/O Pin	-25	-	+50	mA
Recommended VDD Operating Range	4.75	-	5.25	V
Supply Current	-	7	-	mA

Appendix A
uM-FPU Opcode Summary

Opcode Name	Data Type	Opcode	Arguments	Returns	B Reg	Description
SELECTA		0x				Select A register
SELECTB		1x			x	Select B register
WRITEA	Either	2x	yyyy zzzz			Write register and select A
WRITEB	Either	3x	yyyy zzzz		x	Write register and select B
READ	Either	4x		yyyy zzzz		Read register
SET	Either	5x				A = B
FADD	Float	6x			x	A = A + B
FSUB	Float	7x			x	A = A - B
FMUL	Float	8x			x	A = A * B
FDIV	Float	9x			x	A = A / B
LADD	Long	Ax			x	A = A + B
LSUB	Long	Bx			x	A = A -B
LMUL	Long	Cx			x	A = A * B
LDIV	Long	Dx			x	A = A / B
SQRT	Float	E0				A = sqrt(A)
LOG	Float	E1				A = ln(A)
LOG10	Float	E2				A = log(A)
EXP	Float	E3				A = e ** A
EXP10	Float	E4				A = 10 ** A
SIN	Float	E5				A = sin(A) radians
COS	Float	E6				A = cos(A) radians
TAN	Float	E7				A = tan(A) radians
FLOOR	Float	E8				A = nearest integer <= A
CEIL	Float	E9				A = nearest integer >= A
ROUND	Float	EA				A = nearest integer to A
NEGATE	Float	EB				A = -A
ABS	Float	EC				A = |A|
INVERSE	Float	ED				A = 1 / A
DEGREES	Float	EE				Convert radians to degrees A = A / (PI / 180)
RADIANS	Float	EF				Convert degrees to radians A = A * (PI / 180)
SYNC		F0		5C		Synchronization
FLOAT	Long	F1			0	Copy A to Register 0 Convert long to float
FIX	Float	F2			0	Copy A to Register 0 Convert float to long
FCOMPARE	Float	F3		ss		Compare A and B (floating point)

Opcode Name	Data Type	Opcode	Arguments	Returns	B Reg	Description
LOADBYTE	Float	F4	bb		0	Write signed byte to Register 0 Convert to float
LOADUBYTE	Float	F5	bb		0	Write unsigned byte to Register 0 Convert to float
LOADWORD	Float	F6	wwww		0	Write signed word to Register 0 Convert to float
LOADUWORD	Float	F7	wwww		0	Write unsigned word to Register 0 Convert to float
READSTR		F8		aa … 00		Read zero terminated string from string buffer
ATOF	Float	F9	aa … 00		0	Convert ASCII to float Store in A
FTOA	Float	FA	ff			Convert float to ASCII Store in string buffer
ATOL	Long	FB	aa … 00		0	Convert ASCII to long Store in A
LTOA	Long	FC	ff			Convert long to ASCII Store in string buffer
FSTATUS	Float	FD		ss		Get floating point status of A
FUNCTION		FE0n				User functions 0-15
FUNCTION		FE1n				User functions 16-31
FUNCTION		FE2n				User functions 32-47
FUNCTION		FE3n				User functions 48-63
LWRITEA	Long	FEAx	yyyy zzzz			Write register and select A
LWRITEB	Long	FEBx	yyyy zzzz		0	Write register and select B
LREAD	Long	FECx		yyyy zzzz		Read register
LUDIV	Long	FEDx			0	A = A / B (unsigned long)
POWER	Float	FEE0				A = A ** B
ROOT	Float	FEE1				A = the Bth root of A
MIN	Float	FEE2				A = minimum of A and B
MAX	Float	FEE3				A = maximum of A and B
FRACTION	Float	FEE4			0	Load Register 0 with the fractional part of A
ASIN	Float	FEE5				A = asin(A) radians
ACOS	Float	FEE6				A = acos(A) radians
ATAN	Float	FEE7				A = atan(A) radians
ATAN2	Float	FEE8				A = atan(A/B)
LCOMPARE	Long	FEE9		ss		Compare A and B (signed long integer)
LUCOMPARE	Long	FEEA		ss		Compare A and B (unsigned long integer)
LSTATUS	Long	FEEB		ss		Get long status of A
LNEGATE	Long	FEEC				A = -A
LABS	Long	FEED				A = |A|
LEFT		FEEE				Right parenthesis
RIGHT		FEEF			0	Left parenthesis

Opcode Name	Data Type	Opcode	Arguments	Returns	B Reg	Description
LOADZERO	Either	FEF0			0	Load Register 0 with zero
LOADONE	Float	FEF1			0	Load Register 0 with 1.0
LOADE	Float	FEF2			0	Load Register 0 with e
LOADPI	Float	FEF3			0	Load Register 0with pi
LONGBYTE	Long	FEF4	bb		0	Write signed byte to Register 0 Convert to long
LONGUBYTE	Long	FEF5	bb		0	Write unsigned byte to Register 0 Convert to long
LONGWORD	Long	FEF6	wwww		0	Write signed word to Register 0 Convert to long
LONGUWORD	Long	FEF7	wwww		0	Write unsigned word to Register 0 Convert to long
IEEEMODE		FEF8				Set IEEE mode (default)
PICMODE		FEF9				Set PIC mode
BREAK		FEFB				Debug breakpoint
TRACEOFF		FEFC				Turn debug trace off
TRACEON		FEFD				Turn debug trace on
TRACESTR		FEFE	aa … 00			Send debug string to trace buffer
CHECKSUM		FEFF			0	Calculate code checksum
VERSION		FF				Copy version string to string buffer

Notes:

Data Type	data type required by opcode
Opcode	hexadecimal opcode value
Aruments	additional data required by opcode
Returns	data returned by opcode
B Reg	value of B register after opcode executes
x	register number (0-15)
n	function number (0-63)
yyyy	most significant 16 bits of 32-bit value
zzzz	least significant 16 bits of 32-bit value
ss	status byte
bb	8-bit value
wwww	16-bit value
aa … 00	zero terminated ASCII string