Micromega Corporation

uM-FPU Floating Point Coprocessor

V2 Datasheet

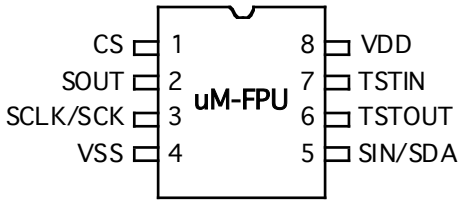
Introduction

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 either an I2C or SPI compatible interface.

uM-FPU Features

- 8-pin integrated circuit.
- I2C compatible interface up to 400 kHz
- SPI compatible interface up to 4 Mhz
- 32 byte instruction buffer
- 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
  - Read, Compare, Status
- Long Integer Operations
  - Set, Add, Subtract, Multiply, Divide, Unsigned Divide
  - Increment, Decrement, Negate, Abs
  - And, Or, Xor, Not, Shift
  - Read 8-bit, 16-bit, and 32-bit
  - 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 formatted ASCII
  - Convert long integer to formatted ASCII
  - Convert ASCII to floating point
  - Convert ASCII to long integer
- User Defined Functions can be stored in Flash memory
  - Conditional execution
  - Table lookup
  - N^{th} order polynomials

Pin Diagram and Pin Description

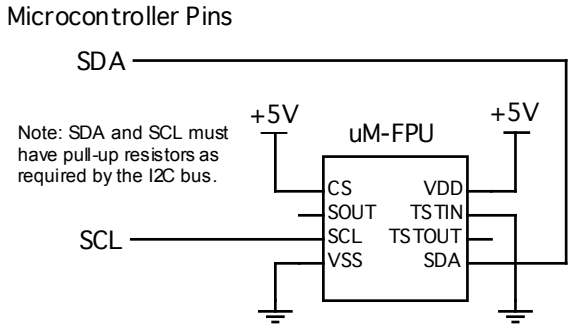


Pin	Name	Type	Description
1	CS	Input	Chip Select
2	SOUT	Output	SPI Output Busy/Ready
3	SCLK SCK	Input	SPI Clock I2C Clock
4	VSS	Power	Ground
5	SIN SDA	Input In/Out	SPI Input I2C Data
6	TSTOUT	Output	Test Output
7	TSTIN	Input	Test Input
8	VDD	Power	Supply Voltage

Connecting the uM-FPU to the I2C compatible interface

If the CS pin is a logic high at reset (e.g. tied to +5V), the uM-FPU will be configured as an I2C slave device. Using an I2C interface allows the uM-FPU to share the I2C bus with other peripheral chips. The connection diagram is shown below.

I2C Connection



I2C Slave Address

The slave address is 7 bits long, followed by an 8th bit which specifies whether the master wishes to write to the slave (0), or read from the slave(1). The default slave address for the uM-FPU is 1100100x (binary).

- expressed as a 7-bit value, the default slave address is 100 (decimal), or 0x64 (hex).
- expressed as a left justified 8-bit value the default slave address is 200 (decimal) or 0xC8 (hex).

The slave address can be changed using the built-in serial debug monitor and stored in nonvolatile flash memory.

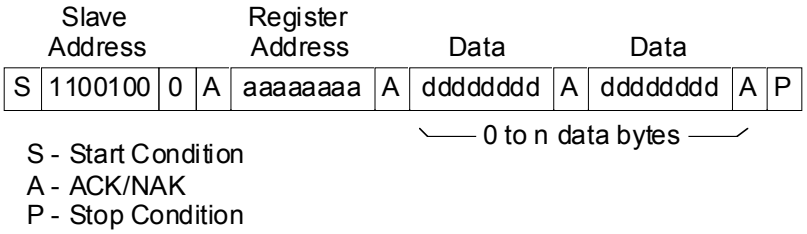
I2C Bus Speed

The uM-FPU can handle I2C data speeds up to 400 kHz.

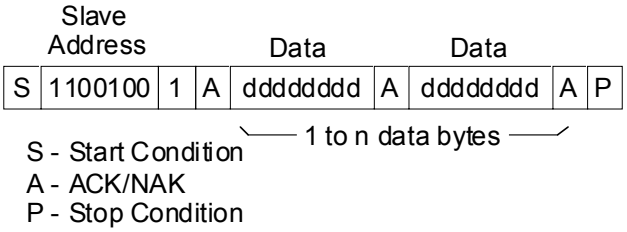
I2C Data Transfers

The following diagrams show the write and read and data transfers. The write transfer consists of a slave address, followed by a register address, followed by 0 to n data bytes. The read transfer is normally proceeded by a write transfer to select the register to read from.

I2C Write Data Transfer



I2C Read Data Transfer



I2C Registers

Register Address	Write	Read
0	Data	Data / Status
1	Reset	Buffer Space

Item	Min	Max	Unit
I2C transfer speed		400	kHz
Read Delay – normal operation	50	90	usec
Read Delay – debug enabled	100	180	usec

I2C Reset Operation

The uM-FPU should be reset at the beginning of every program to ensure that the microcontroller and the uM-FPU are synchronized. The uM-FPU is reset by writing a zero byte to I2C register address 1. A delay of 8 milliseconds is recommended after the reset operation 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 0x7FC00000.

I2C Reading and Writing Data

uM-FPU instructions and data are written to I2C register 0. Reading I2C register 0 will return the next data byte, if data is waiting to be transferred. If no data is waiting to be transferred the Busy/Ready status is returned. A read operation is normally preceded by a write operation to select the I2C register to read from.

I2C Busy/Ready Status

The Busy/Ready status must always be checked to confirm that the uM-FPU is Ready prior to any read operation. The Busy status is asserted as soon as a valid opcode is received. The Ready status is asserted when both the instruction buffer and trace buffer are empty. If the uM-FPU is Ready, a zero byte is returned. If the uM-FPU is Busy, either executing instructions, or because the debug monitor is active, a 0x80 byte is returned. If more than 32 bytes of data are sent between read operations, the Ready status must also be checked at least once every 32 bytes to ensure that the instruction buffer does not overflow.

I2C Buffer Space

Reading I2C register 1 will return the number of bytes of free space in the instruction buffer. This can be used by more advanced interface routines to ensure that the instruction buffer remains fully utilized. It is only used to determine if there is space to write data to the uM-FPU. The Busy/Ready status must still be used to confirm the Ready status prior to any read operation.

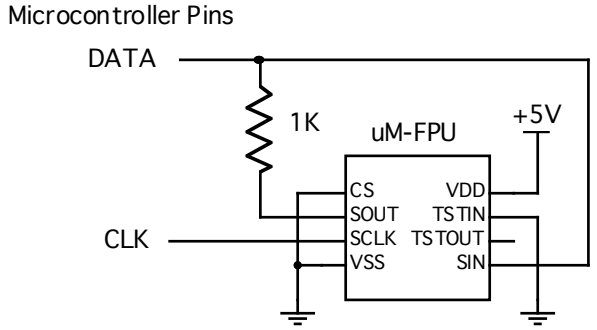
Read Delay

There is a minimum delay required from the end of a read instruction opcode until the first data byte is ready to be read. If debug tracing is active, this delay is longer (see table). With many microcontrollers the call overhead for the interface routines is long enough that no additional delay is required. On faster microcontrollers a suitable delay must be inserted after a read instruction to ensure that data is valid before the first byte is read. A 180 microsecond read delay will handle all circumstances (including the debug mode).

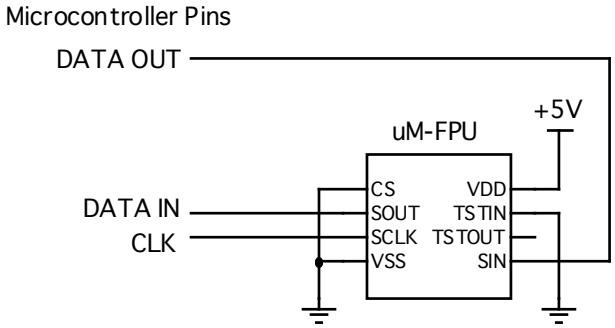
Connecting the uM-FPU using the SPI compatible interface

If the CS pin is a logic low at reset (e.g. tied to GND), the uM-FPU will be configured as a SPI slave device. The uM-FPU can be connected using either a 2-wire or 3-wire SPI interface depending on the capabilities of the microcontroller. The 3-wire SPI connection uses separate data input and data output pins on the microcontroller. The 2-wire SPI connection uses a single bidirectional pin for both data input and data output. If a 2-wire SPI interface is used, the SOUT and SIN pins should not be connected directly together, ***they must be connected through a 1K resistor.*** The microcontroller data pin is connected to the SIN pin. The connection diagrams are shown below.

2-wire SPI Connection



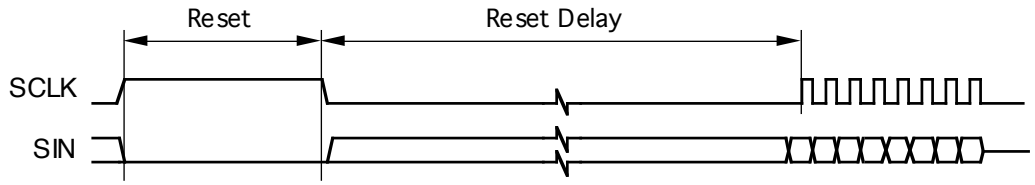
3-wire SPI Connection



SPI Reset Operation

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 200 microseconds. The reset operation will not occur until the SCLK line returns Low. A delay of 8 milliseconds is recommended 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	Typical	Max	Unit
Reset - SCLK Output High	500			usec
Reset - SIN Output Low	100			usec
Reset Delay	3	8		msec

SPI Reading and Writing Data

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, summarized as follows:

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

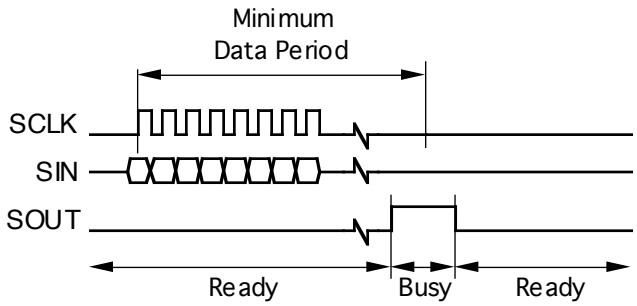
The maximum SCLK frequency is 4 MHz, but there must be minimum data period between bytes. The minimum data period is measured from the rising edge of the first bit of one date byte to the rising edge of the first bit of the next data byte. The minimum data period must elapse before the Busy/Ready status is checked.

Read Delay

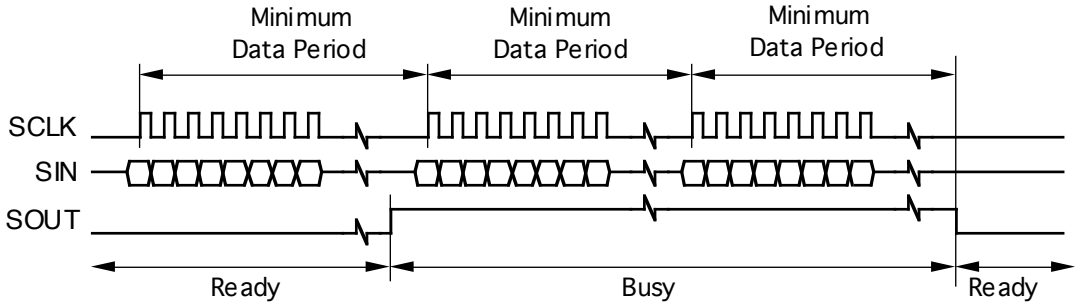
There is a minimum delay required from the end of a read instruction opcode until the first data byte is ready to be read. If debug tracing is active, this delay is longer (see table). With many microcontrollers the call overhead for the interface routines is long enough that no additional delay is required. On faster microcontrollers a suitable delay must be inserted after a read instruction to ensure that data is valid before the first byte is read. A 180 microsecond

read delay will handle all circumstances (including the debug mode).

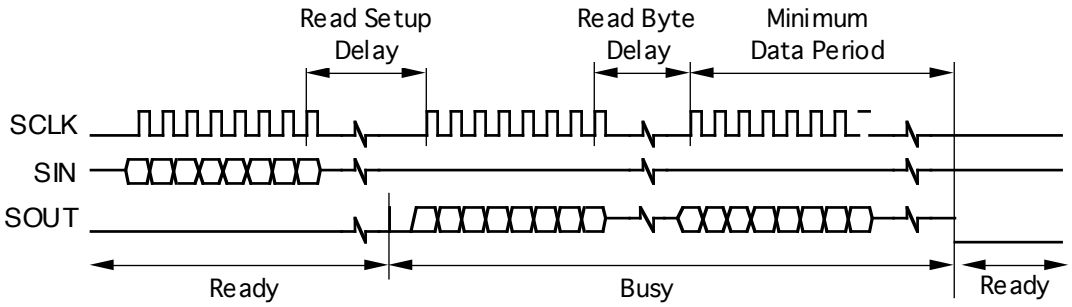
SPI Instruction Timing Diagram
(Opcode)



Instruction Timing Diagram
(Opcode followed by an additional argument)



SPI Instruction Timing Diagram
(Opcode followed by return value)



Item	Min	Max	Unit
SCLK Frequency		4	MHz
SCLK Output Low (per bit)	.25		usec
SCLK Output High (per bit)	.25	60	usec
Minimum Data Period – normal operation	10		usec
Minimum Data Period – debug trace enabled	15		usec
Read Setup Delay – normal operation	90		usec
Read Setup Delay – debug trace enabled	180		usec
Read Byte Delay – normal operation	10		usec
Read Byte Delay – debug trace enabled	15		usec

SPI Busy/Ready Status

The busy/ready status must always be checked to confirm the Ready status prior to any read operation.
The Busy status is asserted as soon as a valid opcode is received. The Ready status is asserted when both the instruction buffer and trace buffer are empty. If the uM-FPU is Ready the SOUT pin is held Low. If the uM-FPU is Busy, either executing instructions, or because the debug monitor is active, the SOUT pin is held High. The minimum data period must have elapsed since the last byte was transmitted before the SOUT status is checked. If more than 32 bytes of data are sent between read operations, the Ready status must also be checked at least once every 32 bytes to ensure that the instruction buffer does not overflow.

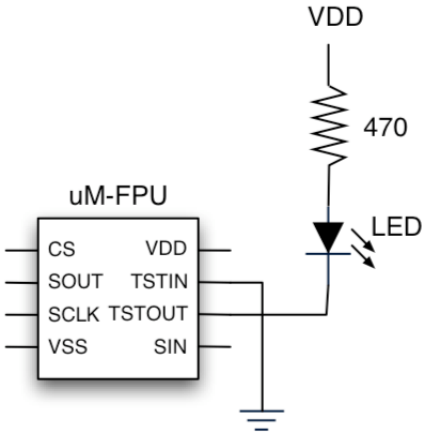
Using the TSTIN and TSTOUT Pins

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 GND.

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 Busy, and will be Low when it is Ready. 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.

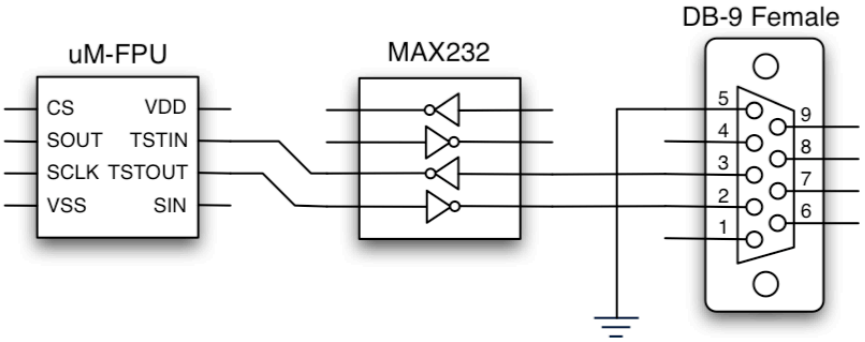
Activity Indicator



Serial Interface for Debug Monitor

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 displaying the contents of uM-FPU registers, tracing the execution of uM-FPU instructions, setting breakpoints for debugging, and programming user functions. Whenever the uM-FPU is reset and the serial interface is enabled the following message is displayed:

```
{RESET}
```

Commands are specified by typing an 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 are displayed, and the command is executed. If the command is not recognized, a question mark is displayed. Special commands are prefixed with a dollar sign. These commands are used to program the user functions and to check the contents of the uM-FPU. They are not generally used when debugging a running application. The $M and $P will reset the uM-FPU on completion. The commands are listed below:

B	Break	stop execution after next opcode
G	Go	continue execution
R	Register	display registers
T	Trace	toggle trace mode on/off
V	Version	display version information
/	Comment	add comment to debug trace
$C	Checksum	display checksum value
$D	Display	display user function memory
$M	Mode	set mode parameters
$P	Program	program user function memory

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
```

Registers – 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}
```

Trace – toggle trace mode 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}
```

Version – display version information

The Version command displays the version string for the uM-FPU chip and the selected interface. If the selected interface is I2C the I2C address is also shown.

```
>V
uM-FPU V2.0 I2C C8
```

Comment – add comment to debug trace

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
```

Checksum – display checksum value

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 – display user function memory

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

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

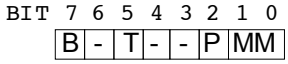
Mode – set mode parameters

The Mode command is used to set the four interface mode parameter bytes that are stored in Flash memory. The factory setting of the parameter bytes is all zeros. The parameter bytes are read at reset to determine the mode of operation. The mode command displays the current parameter values and the user is prompted to enter new values. (The values are entered as hexadecimal values.) The new values are programmed into Flash memory and the uM-FPU is Reset.

```
>$M
 00000000
:00CA0000
```

Two hexadecimal digits represent each parameter byte. The mode parameter bytes are interpreted as follows:

Byte 0:



Bit 7	Break on Reset (if debug mode is enabled)
Bit 5	Trace on Reset (if debug mode is enabled)
Bit 2	PIC mode enabled (see PICMODE instruction)
Bits 1:0	Mode 00 – CS pin determines interface mode (default) 01 – I2C mode selected (CS pin ignored) 10 – SPI mode selected (CS pin ignored) 11 – SPI mode selected (CS pin is active as chip select for data transfers)

Byte 1: I2C Address (if zero, the default address (0xC8) is used.
The 7-bit address is entered as a left justified 8-bit value. The last bit is ignored.

Byte 2: reserved

Byte 3: reserved

Program – program user function memory

The Program command is used to program the user function memory. Once in program mode, the uM-FPU looks for valid Intel Hex format records. The records must have an address between 0x0000 and 0x03C0, start on a 64-byte boundary, and have a length of 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.

The uM-FPU IDE program (or another PC based 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 debug monitor 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 and the debug monitor is entered. Execution will only resume when a Go command is issued to the debug monitor.

TRACEOFF

Turns the debug trace mode off.

TRACEON

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

TRACESTR

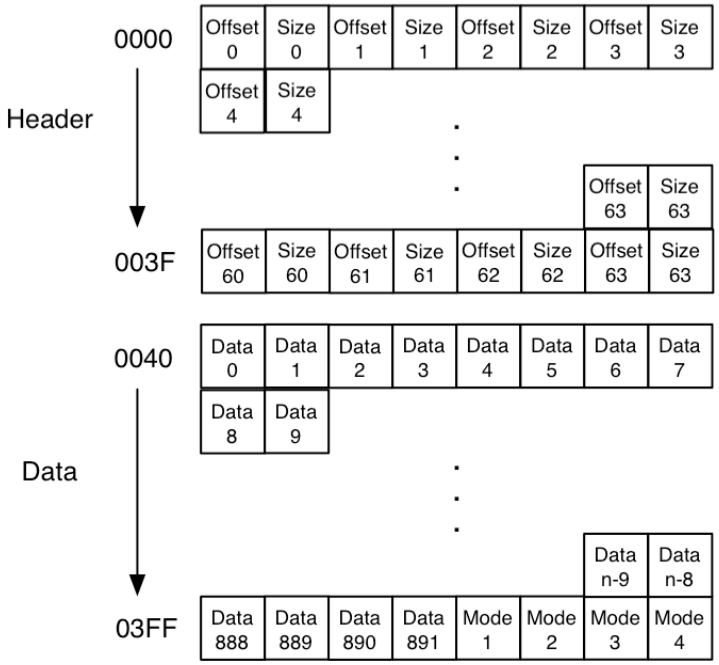
Displays a trace string to the debug monitor output. This can be useful for keeping track of a debug session. Trace strings are always output; they are not affected by the trace mode.

Stored Functions

There are 1024 bytes of flash memory reserved on the uM-FPU for storing user functions and the mode paramaters. Up to 64 user functions can be defined and saved by the user. Stored user functions have the advantage of conserving space on the microcontroller and greatly reducing the communications overhead between the microcontroller and the uM-FPU. In addition, certain instructions (e.g. IF_XXX, TABLE, POLY) are only valid in user defined functions. Opcodes FE00 through FE3F are used to execute the stored user functions 0 through 63. The Busy condition remains set while all of the opcodes in the stored function execute.

User 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.

The data section contains the opcodes and data for the instructions that make up the user defined function. User functions are programmed using the serial debug monitor. The uM-FPU IDE application supports the definition of user functions. (Refer to uM-FPU IDE documentation.)



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 V2 Instruction Summary

Opcode Name	Data Type	Opcode	Arguments	Returns	B Reg	Description
SELECTA		0x				Select A register
SELECTB		1x			x	Select B register
FWRITEA	Float	2x	yyyy zzzz			Write register and select A
FWRITEB	Float	3x	yyyy zzzz		x	Write register and select B
FREAD	Float	4x		yyyy zzzz		Read register
FSET/LSET	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 Remainder stored in register 0
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)
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
XOP		FE				Extended opcode prefix (extended opcodes are listed below)
NOP		FF				No Operation
FUNCTION		FE0n FE1n FE2n FE3n			0	User defined functions 0-15 User defined functions 16-31 User defined functions 32-47 User defined functions 48-63
IF_FSTATUSA	Float	FE80	ss			Execute user function code if FSTATUSA conditions match
IF_FSTATUSB	Float	FE81	ss			Execute user function code if FSTATUSB conditions match
IF_FCOMPARE	Float	FE82	ss			Execute user function code if FCOMPARE conditions match
IF_LSTATUSA	Long	FE83	ss			Execute user function code if LSTATUSA conditions match
IF_LSTATUSB	Long	FE84	ss			Execute user function code if LSTATUSB conditions match
IF_LCOMPARE	Long	FE85	ss			Execute user function code if LCOMPARE conditions match
IF_LUCOMPARE	Long	FE86	ss			Execute user function code if LUCOMPARE conditions match
IF_LTST	Long	FE87	ss			Execute user function code if LTST conditions match
TABLE	Either	FE88				Table Lookup (user function)
POLY	Float	FE89				Calculate n^{th} degree polynomial (user function)
READBYTE	Long	FE90		bb		Get lower 8 bits of register A
READWORD	Long	FE91		bb		Get lower 16 bits of register A
READLONG	Long	FE92		bb		Get long integer value of register A
READFLOAT	Float	FE93		bb		Get floating point value of register A
LINCA	Long	FE94				A = A + 1
LINCB	Long	FE95				B = B + 1
LDECA	Long	FE96				A = A - 1
LDECB	Long	FE97				B = B - 1
LAND	Long	FE98				A = A AND B
LOR	Long	FE99				A = A OR B
LXOR	Long	FE9A				A = A XOR B
LNOT	Long	FE9B				A = NOT A
LTST	Long	FE9C	ss			Get the status of A AND B
LSHIFT	Long	FE9D				A = A shifted by B bit positions
LWRITEA	Long	FEAx	yyyy zzzz			Write register and select A
LWRITEB	Long	FEBx	yyyy zzzz		x	Write register and select B
LREAD	Long	FECx		yyyy zzzz		Read register
LUDIV	Long	FEDx			x	A = A / B (unsigned long) Remainder stored in register 0
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
LOADZERO	Float	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
CHECKSUM		FEFA			0	Calculate checksum for uM-FPU code
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
VERSION		FEFF				Copy version string to string buffer

Notes:

Data Type	data type required by opcode
Opcode	hexadecimal opcode value
Arguments	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