uM-FPU V3.1 Datasheet

32-bit Floating Point Coprocessor

Micromega Corporation

Introduction

The uM-FPU V3.1 chip easily interfaces to virtually any microcontroller using a SPI™ or I^{2}C™ interface.
Many microcontrollers used in embedded systems lack floating point support, but a wide range of sensors available today require additional computations or data transformation to provide accurate results.

Advanced operations and fast execution allows the uM-FPU V3.1 chip to outperform comparable software math libraries. It also provides Flash memory and EEPROM for storing user-defined functions and data, and 128 32-bit registers for floating point and integer data.

Software math libraries often use large amounts of memory on microcontrollers, particularly as more complex library functions are used. The uM-FPU V3.1 chip offloads this overhead, and provides a comprehensive set of floating point operations, including advanced functions such as FFT, matrix operations and NMEA sentence parsing.

Development support is provided by the uM-FPU V3 IDE which takes traditional math expressions and automatically produces uM-FPU code targeted for one of the many microcontrollers and compilers supported. The IDE also interacts with the built-in debugger on the uM-FPU V3.1 chip to assist in debugging and testing the uM-FPU code.

Applications

- **sensor data processing**
- **GPS data input and processing**
- **robotic control**
- **data transformations**
- **embedded systems**

Features

- **32-bit IEEE 754 floating point**
- **32-bit integer operations**
- **GPS serial input**
- **NMEA sentence parsing**
- **FFT operations**
- **12-bit A/D Converters**
- **Serial input/output**
- **String handling**
- **Matrix operations**
- **SPI™ or I^{2}C™ interface**
- **2.7V, 3.3V, 5V supply**
- **low power modes**
- **18-pin DIP, SOIC-18, QFN-44**
- **RoHS compliant**

Features

32-bit Floating Point and 32-bit Integer
A comprehensive set of 32-bit floating point and 32-bit integer operations are provided. See the *uM-FPU V3.1 Instruction Set* document for details.

User-defined Functions
User-defined functions can be stored in Flash and EEPROM. Flash functions are programmed through the SERIN/SEROUT pins using the uM-FPU V3 IDE. The EEPROM functions can be programmed at run-time. Conditional execution is supported using conditional branch and jump instructions.

Matrix Operations
A matrix can be defined as any set of sequential registers. The MOP instruction provides scalar operations, element-wise operations, matrix multiply, inverse, determinant, count, sum, average, min, max, copy and set operations.

FFT Instruction
Provides support for Fast Fourier Transforms. Used as a single instruction for data sets that fit in the available registers, or as a multi-pass instruction for working with larger data sets.

Serial Input / Output
When not used for debugging, the SERIN and SEROUT pins can be used for serial I/O. For example, SERIN can be used to read data from a GPS, and SEROUT can be used to drive an LCD.

NMEA Sentence Parsing
The serial input can be set to scan for valid NMEA sentences with optional checksum. Multiple sentences can be buffered for further processing.

String Handling
String instructions are provided to insert and append substrings, search for fields and substrings, convert from floating point or long integer to a substring, or convert from a substring to floating point or long integer. For example, the string instructions could be used to parse a GPS NMEA sentence, or format multiple numbers in an output string.

Table Lookup Instructions
Instructions are provided to load 32-bit values from a table or find the index of a floating point or long integer table entry that matches a specified condition.

MAC Instructions
Instructions are provided to support multiply and accumulate and multiply and subtract operations.

A/D Conversion
Two 12-bit A/D channels are provided. The A/D conversion can be triggered manually, through an external input, or from a built-in timer. The A/D values can be read as raw values or automatically scaled to a floating point value. Data rates of up to 10,000 samples per second are supported.

Timers
Timers can be used to trigger the A/D conversion, or to track elapsed time. A microsecond and second timer are provided.

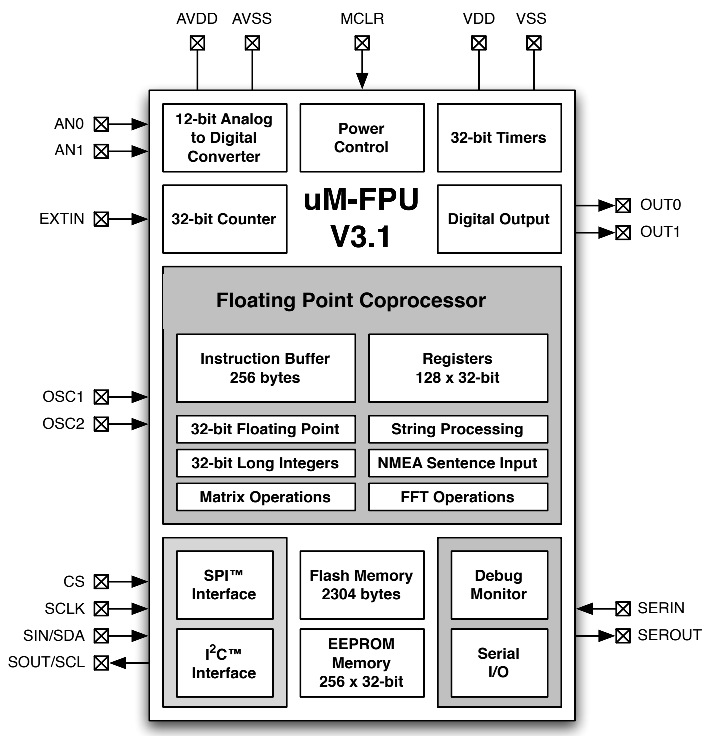
External Input
An external input can be used to trigger an A/D conversion, or to count external events.

Low Power Modes
When the uM-FPU V3.1 chip is not busy it automatically enters a power saving mode. It can also be configured to enter a sleep mode which turns the device off while preserving register contents. In sleep mode the uM-FPU V3.1 chip consumes negligible power.

Internal Oscillator
Operates at full speed from internal oscillator. No external components required.

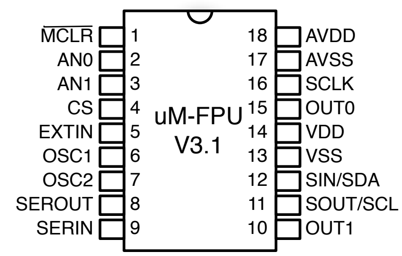
Core Features
- Packages: 18-pin DIP, SOIC-18, QFN-44
- Supply voltages: 5V, 3.3V, 2.7V
- Operating temperature: -40°C to +85°C
- RoHS compliant
- I^2C compatible interface up to 400 kHz
- SPI compatible interface up to 15 MHz
- internal oscillator
- no external components required
- supports optional external oscillator
- 256 byte instruction buffer
- 128 general purpose 32-bit registers
- 8 temporary 32-bit registers
- 2304 bytes Flash memory for user-defined functions
- 1024 bytes EEPROM for data storage or user-defined functions

Block Diagram



Pin Diagram

PDIP-18, SOIC-18



Pin Descriptions

Pin	Name	Type	Description
1	/MCLR	Input	Master Clear (Reset)
2	AN0	Input	Analog Input 0
3	AN1	Input	Analog Input 1
4	CS	Input	Chip Select / Interface Select
5	EXTIN	Input	External Input
6	OSC1	Input	Oscillator Crystal (optional)
7	OSC2	Output	Oscillator Crystal (optional)
8	SEROUT	Output	Serial Output, Debug Monitor - Tx
9	SERIN	Input	Serial Input, Debug Monitor - Rx
10	OUT1	Output	Digital Output 1, Ready/Busy Status
11	SOUT SCL	Output Input	SPI Output, Busy/Ready Status I^2C Clock
12	SIN SDA	Input In/Out	SPI Input I^2C Data
13	VSS	Power	Digital Ground
14	VDD	Power	Digital Supply Voltage
15	OUT0	Output	Digital Output 0
16	SCLK	Input	SPI Clock
17	AVSS	Power	Analog Ground
18	AVDD	Power	Analog Supply Voltage

Connecting to the uM-FPU V3.1 chip

The uM-FPU V3.1 chip can be interfaced using one of several different types of SPI interface, or an I^2C interface. The different types are as follows:

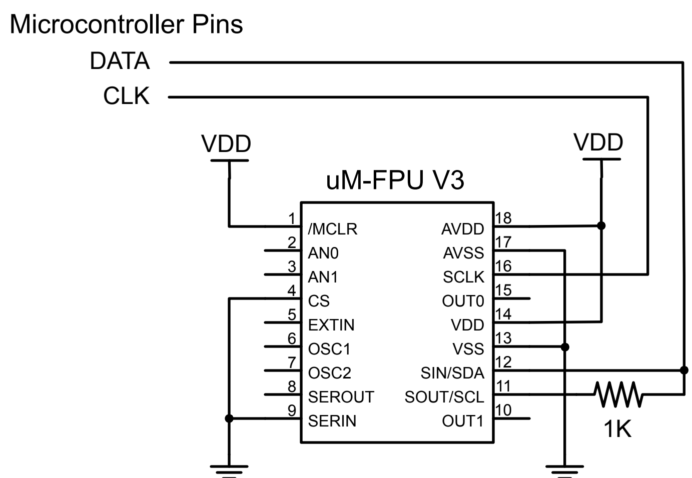
- 2-wire SPI interface, single device
- 3-wire SPI interface, single device
- SPI bus interface, multiple devices
- I^2C interface, multiple devices

By default, the `CS` pin is used to select between SPI or I^2C interfaces. To use the `CS` pin as a chip select, as required by the SPI bus interface, a parameter byte stored in Flash must be set. This is described below, in the section called *SPI Bus Interface*.

2-wire SPI interface

When the uM-FPU V3.1 chip is connected directly to the microcontroller as a single device, no chip select is required, and either a 2-wire or 3-wire SPI interface can be used depending on the capabilities of the microcontroller. The 2-wire SPI connection uses a single bidirectional pin for both data input and data output. When 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 `CS` pin is tied low to select SPI mode at Reset, and must remain low during operation. The connection diagrams are shown below.

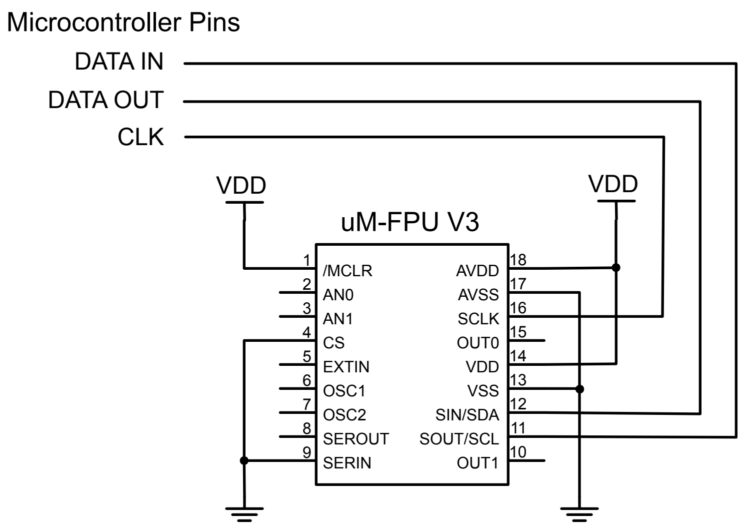
2-wire SPI Connection



3-wire SPI interface

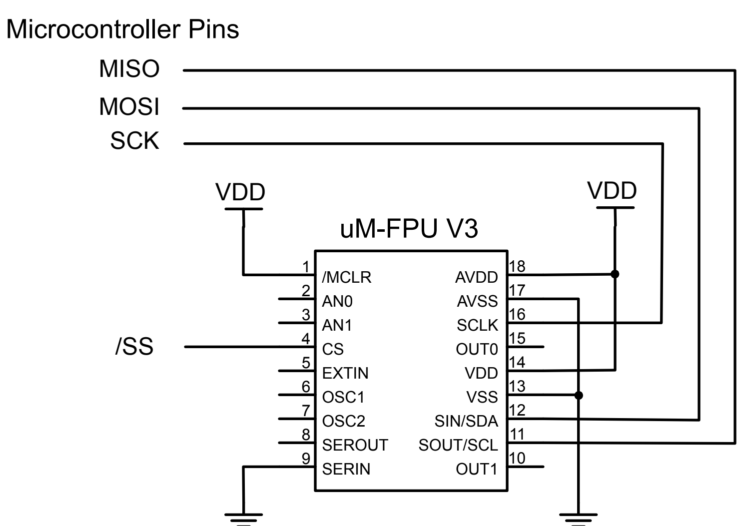
The 3-wire SPI connection uses separate data input and data output pins on the microcontroller. The `CS` pin is tied low to select SPI mode at Reset, and must remain low during operation.

3-wire SPI Connection



SPI Bus Interface

In order for the uM-FPU V3.1 chip to be used on a SPI bus with multiple devices, the `CS` pin must be enabled as a chip select. This is accomplished by programming mode parameter bits stored in Flash memory on the uM-FPU V3.1 chip. Bits 1:0 of mode parameter byte 0 must be set to 11 to select SPI bus mode. When this mode is set, the SPI interface is automatically selected at Reset, and the `CS` pin is enabled as a standard active low slave select. The `SOUT` pin is a tri-state output and is high impedance when the uM-FPU V3.1 chip is not selected. The connection diagram is shown below:

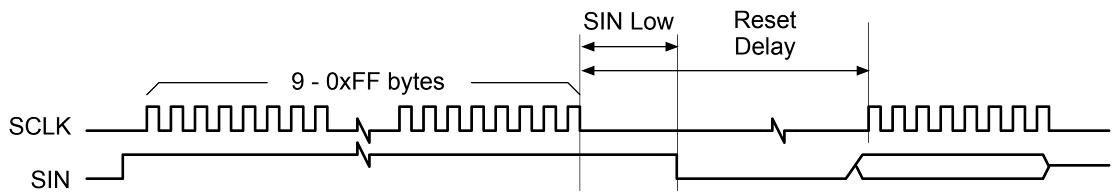


The clock signal is idle low and data is read on the rising edge of the clock (often referred to as SPI Mode 0).

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. The uM-FPU will prepare for a reset after nine consecutive 0xFF bytes are read, but it is recommended that ten 0xFF bytes be sent by the microcontroller to ensure that at least nine 0xFF bytes are recognized even if the microcontroller and uM-FPU are out of sync. The reset does not occur until the `SIN` signal goes Low. If `SIN` remains High after sending the ten 0xFF bytes, a 0x00 byte must be sent (or `SIN` set Low) to trigger the reset. Note: If SIN does not go Low within 100 milliseconds of receiving nine 0xFF bytes, a reset will be triggered by default. A delay of 10 milliseconds is recommended after the reset is triggered to ensure that the reset sequence 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 7FFFFFFF.

Reset Timing Diagram



Item	Min	Typical	Max	Unit
Reset - 0xFF bytes	9	10		bytes
Reset - SIN Low			100	msec
Reset Delay	10			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 15 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 (Read Setup Delay) required from the end of a read instruction opcode until the first data byte is ready to be read. 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.

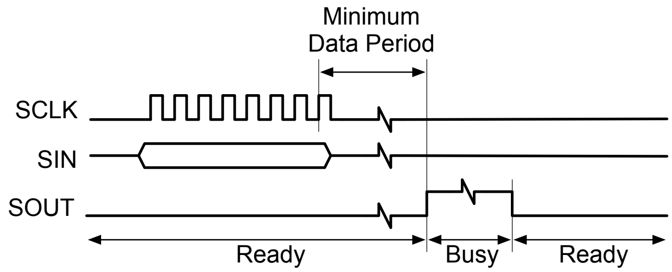
SPI Busy/Ready Status

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 an instruction byte 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 256 bytes of data are sent between read operations, the Ready status must also be checked at least once

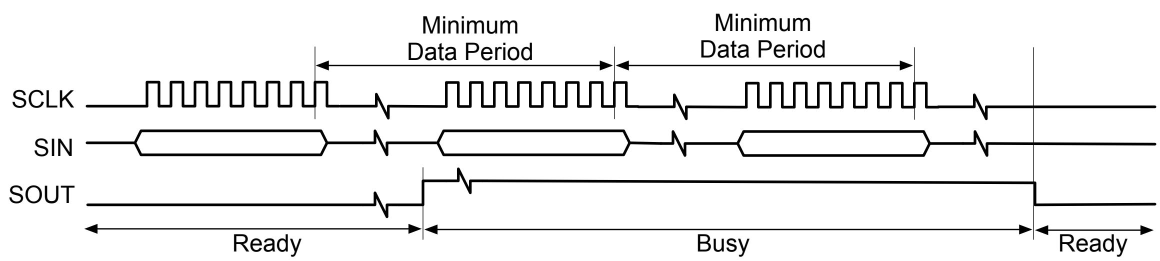
every 256 bytes to ensure that the instruction buffer does not overflow. The `OUT1` pin can also be used to check the Busy/Ready Status, see the section entitled *Using `OUT1` as a Ready/Busy Status*.

SPI Instruction Timing Diagrams

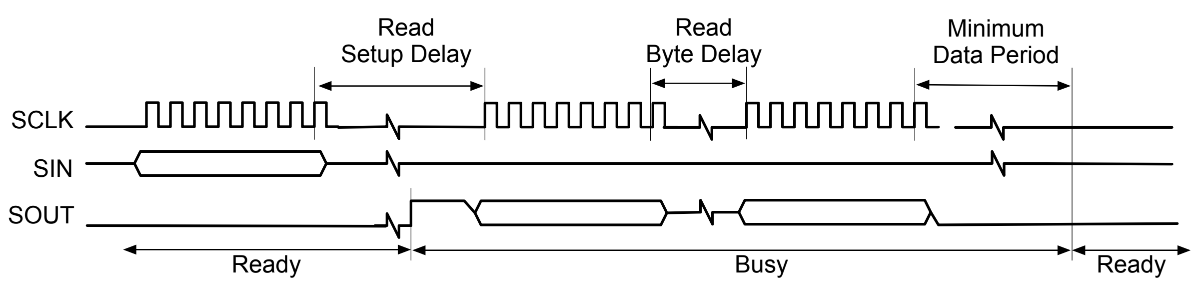
Single Byte Opcode



Multiple Byte Opcode



Opcode followed by return value

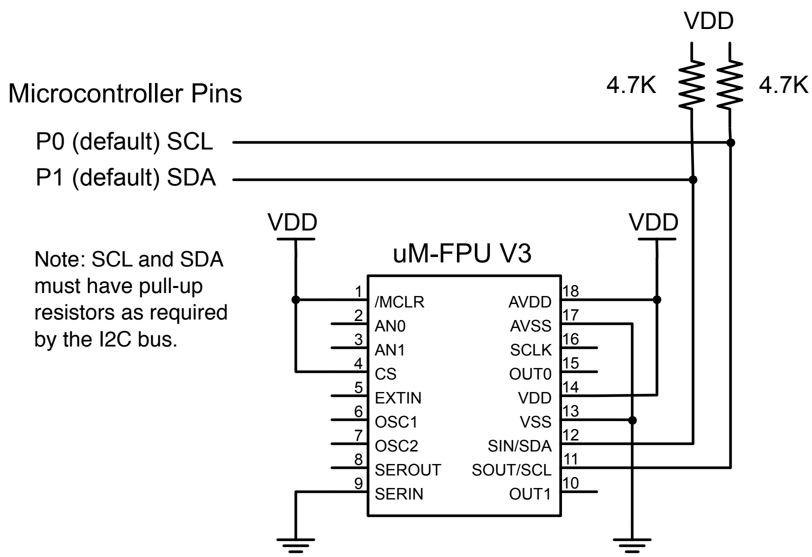


Item	Min	Max	Unit
SCLK Output Low	30		nsec
SCLK Output High	30		nsec
SCLK Frequency - single byte		15	MHz
SCLK Frequency - continuous		5	MHz
Minimum Data Period	1.6		usec
Read Setup Delay	15		usec
Read Byte Delay	1		usec
Falling Edge of CS to Rising Edge of SCLK	120		nsec
Falling Edge of CS to Busy/Ready Check	1		usec
Rising Edge of CS to Bus Released		500	nsec

I²C interface

If the CS pin is a logic high at reset (e.g. tied to VDD), 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.

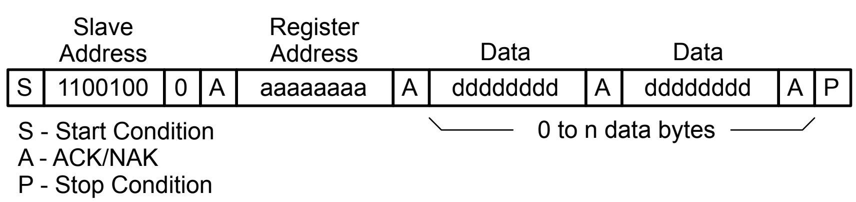
I²C Bus Speed

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

I²C Data Transfers

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

I²C Write Data Transfer



I^{2}C Read Data Transfer

	Slave Address			Data		Data		
S	1100100	1	A	dddddddd	A	dddddddd	N	P

1 to n data bytes

S - Start Condition
A - ACK
N - NAK
P - Stop Condition

I^{2}C Registers

I^{2}C Register Address	Write	Read
0	Data	Data / Status
1	Reset	Buffer Space

Item	Min	Max	Unit
I^{2}C transfer speed		400	kHz
Read Delay – normal operation	TBD	TBD	usec
Read Delay – debug enabled	TBD	TBD	usec

I^{2}C 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 I^{2}C 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.

I^{2}C Reading and Writing Data

uM-FPU instructions and data are written to I^{2}C register 0. Reading I^{2}C 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 I^{2}C register to read from.

I^{2}C 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 an instruction byte 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 256 bytes of data are sent between read operations, the Ready status must also be checked at least once every 256 bytes to ensure that the instruction buffer does not overflow.

I^{2}C Buffer Space

Reading I^{2}C 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 (Read Setup Delay) required from the end of a read instruction opcode until the first data byte is ready to be read. 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.

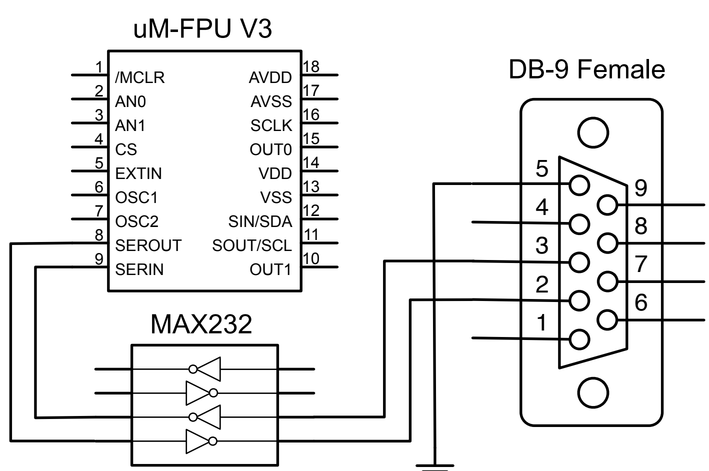
Using OUT1 as a Ready/Busy Status

By default, the uM-FPU V3.1 chip outputs the Busy/Ready status on the `SOUT` pin, when the `SOUT` pin is not being used for data input. Some microcontroller applications are not able to access this pin when the Busy/Ready status is valid. As an alternative, the `OUT1` pin can be configured as a Ready/Busy status (note: `OUT1` is High for Ready and Low for Busy). This is accomplished by programming bit 6 of mode parameter byte 0. See the section entitled *Mode - set mode parameters*. When `OUT1` is set to output the Ready/Busy status, the `SOUT` pin will no longer output the Busy/Ready status. The `OUT1` pin can also be used as an activity indicator by connected it to an LED with a pull-up resistor.

Using the SERIN and SEROUT Pins

The `SERIN` and `SEROUT` pins provide a serial interface for the built-in Debug Monitor, and can also be used for general purpose serial I/O when the Debug Monitor is not being used. The Debug Monitor communicates at 57,600 baud, using 8 data bits, no parity, one stop bit, and no flow control. The Debug Monitor is enabled if the `SERIN` pin is high when the uM-FPU is Reset. Note: The idle state of an RS-232 connection will assert a high level on the `SERIN` pin, so provided the uM-FPU is connected to an active idle RS-232 port when the uM-FPU is reset, the Debug Monitor will be enabled. The `SEROUT,0` instruction can also be used to enable/disable the Debug Monitor.

When the Debug Monitor is not being used, the serial I/O pins can be used for other purposes. The `SEROUT,0` instruction is used to set the baud rate for the `SERIN` and `SEROUT` pins from 300 to 115,200 baud, using 8 data bits, no parity, one stop bit, and no flow control. The `SERIN` instruction supports reading serial data from the `SERIN` pin, and the `SEROUT` instruction supports sending serial data to the `SEROUT` pin. The uM-FPU V3.1 chip includes support for NMEA sentence parsing, making it easy to connect to a GPS or other NMEA compliant device. The serial output can be used to drive an LCD display or other serial device.



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 V3.1 chip is reset and debug mode is enabled, the following message is sent to the serial output (`SEROUT` pin):

```
{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 instruction
`E`	EEPROM	display EEPROM memory
`F`	Flash	display Flash stored function memory
`G`	Go	continue execution
`R`	Register	display registers
S	String	display string, length and selection point
`T`	Trace	toggle trace mode on/off
`V`	Version	display version information
`X`	Change	displays all register that have changed
`/`	Comment	add comment to debug trace
`$C`	Clock	select clock source
`$M`	Mode	set mode parameters
`$P`	Program	program user function memory
`$S`	Checksum	display checksum value

Break – stop execution after next instruction

The Break command is used to interrupt operation of the uM-FPU. The break will not occur until after the next instruction is executed by the uM-FPU. The debug monitor displays the hex value of the last instruction executed and any additional data. Entering another Break command, or simply pressing the return key, will single step to the next instruction. Entering the Go command will continue execution. Note: the uM-FPU V3 IDE includes a disassembler that translates the trace bytes into a readable instruction sequence.

```
{BREAK}
>
 0103               (i.e. SELECTA,3)
{BREAK}
>
 2001               (i.e. FSET,1)
{BREAK}
>
 3702               (i.e. FDIVI,2)
{BREAK}
>
 2403               (i.e. FMUL,3)
{BREAK}
>
```

EEPROM – display EEPROM memory

The EEPROM command displays the contents of the EEPROM memory in Intel Hex format.

```
>E
:100000000000000000000000000000000000000000F0
:100010000000000000000000000000000000000000E0
:10002000000000000099000000000000000000000037
:100030000000000000000000000000000000000000C0
:100040000102030405060708090A0B0C0000000062
:1000500007360A33057F168003330180000000055
:10006000000000000000000000000000000000000090
:10007000000000000000000000000000000000000080
:10008000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80
:10009000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF70
.
.
.
:1003D000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2D
:1003E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1D
:1003F000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0D
```

Flash – display Flash stored function memory

The Flash command displays the contents of the Flash stored function memory in Intel Hex format.

```
>$F
:10000000000000000100000E010E000801160006AD
:10001000011C0002011E0006012400050129001731
:10002000014000120152000D015F0016017500171A
:10003000018C001701A3000E00000000000000006A
:100040000000000000000000000000000000000000B0
:100050000000000000000000000000000000000000A0
:10006000000000000000000000000000000000000090
:10007000000000000000000000000000000000000080
:10008000000000000000000000000000000000000070
:10009000000000000000000000000000000000000060
:1000A000000000000000000000000000000000000050
:1000B000000000000000000000000000000000000040
:1000C000000000000000000000000000000000000030
:1000D000000000000000000000000000000000000020
:1000E000000000000000000000000000000000000010
:1000F000000000000000000000000000000000000000
:1001000014200124011420022402152A4115142070
:100110000124022103157E047E05330332017E068D
.
.
.
:1008D000000000000000000000000000000000000018
:1008E000000000000000000000000000000000000008
:1008F00000000000000000000000000010C8000020
```

Go – continue execution

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

```
>G
```

Registers – display registers

The Register command displays a header line showing the currently selected register A, register X, the internal status value, and if selected, matrix A, B and C. The current contents of all uM-FPU registers are then displayed.

```
>R
{A=R0, X=R57, S=80, MA=R16:3:3, MB=R32:3:3, MC=R48:3:3
R0:41900000 R1:7FFFFFFF R2:7FFFFFFF R3:7FFFFFFF
R4:40E00000 R5:BF800000 R6:40800000 R7:00000000
R8:C0400000 R9:40800000 R10:00000000 R11:41000000
R12:7FFFFFFF R13:7FFFFFFF R14:7FFFFFFF R15:7FFFFFFF
R16:40000000 R17:40800000 R18:40C00000 R19:41000000
R20:41200000 R21:41400000 R22:41600000 R23:41800000
R24:41900000 R25:7FFFFFFF R26:7FFFFFFF R27:7FFFFFFF
R28:7FFFFFFF R29:7FFFFFFF R30:7FFFFFFF R31:7FFFFFFF
R32:40000000 R33:40800000 R34:40C00000 R35:41000000
R36:41200000 R37:41400000 R38:41600000 R39:41800000
R40:41900000 R41:7FFFFFFF R42:7FFFFFFF R43:7FFFFFFF
R44:7FFFFFFF R45:7FFFFFFF R46:7FFFFFFF R47:7FFFFFFF
R48:40000000 R49:40800000 R50:40C00000 R51:41000000
R52:41200000 R53:41400000 R54:41600000 R55:41800000
R56:41900000 R57:7FFFFFFF R58:7FFFFFFF R59:7FFFFFFF
R60:7FFFFFFF R61:7FFFFFFF R62:7FFFFFFF R63:7FFFFFFF
R64:7FFFFFFF R65:7FFFFFFF R66:7FFFFFFF R67:7FFFFFFF
R68:7FFFFFFF R69:7FFFFFFF R70:7FFFFFFF R71:7FFFFFFF
R72:7FFFFFFF R73:7FFFFFFF R74:7FFFFFFF R75:7FFFFFFF
R76:7FFFFFFF R77:7FFFFFFF R78:7FFFFFFF R79:7FFFFFFF
R80:7FFFFFFF R81:7FFFFFFF R82:7FFFFFFF R83:7FFFFFFF
R84:7FFFFFFF R85:7FFFFFFF R86:7FFFFFFF R87:7FFFFFFF
R88:7FFFFFFF R89:7FFFFFFF R90:7FFFFFFF R91:7FFFFFFF
R92:7FFFFFFF R93:7FFFFFFF R94:7FFFFFFF R95:7FFFFFFF
R96:7FFFFFFF R97:7FFFFFFF R98:7FFFFFFF R99:7FFFFFFF
R100:7FFFFFFF R101:7FFFFFFF R102:7FFFFFFF R103:7FFFFFFF
R104:7FFFFFFF R105:7FFFFFFF R106:7FFFFFFF R107:7FFFFFFF
R108:7FFFFFFF R109:7FFFFFFF R110:7FFFFFFF R111:7FFFFFFF
R112:7FFFFFFF R113:7FFFFFFF R114:7FFFFFFF R115:7FFFFFFF
R116:7FFFFFFF R117:7FFFFFFF R118:7FFFFFFF R119:7FFFFFFF
R120:7FFFFFFF R121:7FFFFFFF R122:7FFFFFFF R123:7FFFFFFF
R124:7FFFFFFF R125:7FFFFFFF R126:7FFFFFFF R127:7FFFFFFF
T1:7FFFFFFF T2:7FFFFFFF T3:7FFFFFFF T4:7FFFFFFF
T5:7FFFFFFF T6:7FFFFFFF T7:7FFFFFFF T8:7FFFFFFF}
```

String – display string, length and selection point

The String command displays the current string buffer and selection point. The string length, selection start point and selection length are displayed, followed by the string. The following example shows an empty string.

```
>S
0,0,0
```

The following example shows the string buffer after the `VERSION` instruction has been executed.

```
>S
13,0,13
uM-FPU V3.1.3
```

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 instruction that is executed by the uM-FPU is displayed. Note: the uM-FPU V3 IDE includes a disassembler that translates the trace bytes into a readable instruction sequence.

```
>T
{TRACE ON}
 0101 5E 29 3600 3714 47 0102 2001 360A 53 61 97:00 0101 1F55 F2" 0.00
000" 0101 5E 29 3602 3714 47 0102 2001 360A 53 61 97:03 0101 1F55 F2"
0.30902" 0101 5E 29 3604 3714 47 0102 2001 360A 53 61 97:06 0101 1F55
F2" 0.58779" 0101 5E 29 3606 3714 47 0102 2001 360A 53 61 97:08 0101 1
F55 F2" 0.80902" 0101 5E 29 3608 3714 47 0102 2001 360A 53 61 97:0A 01
01 1F55 F2" 0.95106" 0101 5E 29 360A 3714 47 0102 2001 360A 53 61 97:0
A 0101 1F55 F2" 1.00000" 0101 5E 29 360C 3714 47 0102 2001 360A 53 61
97:0A 0101 1F55 F2" 0.95106" 0101 5E 29 360E 3714 47 0102 2001 360A 53
 61 97:08 0101 1F55 F2" 0.80902" 0101 5E 29 3610 3714 47 0102 2001 360
A 53 61 97:06 0101 1F55 F2" 0.58779"
>T
{TRACE OFF}
```

Version – display version information

The Version command displays the version string for the uM-FPU chip, the currently selected interface, and the current clock speed. If the selected interface is I^2C the device address is also shown.

```
>V
uM-FPU V3.1.3, SPI 29.48 MHz

>V
uM-FPU V3.1.3, I2C C8 29.48 MHz
```

Change – display changed registers

The Change command displays a header line showing the currently selected register A, register X, the internal status value, and if selected, matrix A, B and C. The current contents of all uM-FPU registers that have changed since the last Change command (or Reset) are then displayed.

```
>X
{A=R0, X=R57, S=80, MA=R16:3:3, MB=R32:3:3, MC=R48:3:3
R0:41900000 R4:40E00000 R5:BF800000 R6:40800000
R7:00000000 R8:C0400000 R9:40800000 R10:00000000
R11:41000000 R16:40000000 R17:40800000 R18:40C00000
R19:41000000 R20:41200000 R21:41400000 R22:41600000
R23:41800000 R24:41900000 R32:40000000 R33:40800000
R34:40C00000 R35:41000000 R36:41200000 R37:41400000
R38:41600000 R39:41800000 R40:41900000 R48:40000000
R49:40800000 R50:40C00000 R51:41000000 R52:41200000
R53:41400000 R54:41600000 R55:41800000 R56:41900000}

>X
{A=R0, X=R57, S=80, MA=R16:3:3, MB=R32:3:3, MC=R48:3:3}
```

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
```

Clock – select clock source

The Clock command allows you to change the clock source. The default clock speed is 29.48 MHz using an internal oscillator which provides the maximum execution speed. The clock speed would only need to be changed for special circumstances such as low-power applications. The clock source is stored in Flash memory as part of the device configuration bits. The clock selection indicates the clock source to use at power-up. If the selected clock source can't be validated at power-up, the uM-FPU V3.1 chip will fall back to an internal clock speed of 1.8425 MHz. The available clock speeds and clock sources are selected by entering one of the following values:

Value	Clock Speed	Clock Source
`20`	1.8425 MHz	internal oscillator
`E1`	7.37 MHz	internal oscillator
`EA`	14.74 MHz	internal oscillator
`E3`	29.48 MHz	internal oscillator (default clock speed)
`E5`	10.0 MHz	external 10.0 MHz crystal
`E6`	20.0 MHz	external 10.0 MHz crystal
`E7`	29.4912 MHz	external 7.3728 MHz crystal

The following example changes the clock selection from 29.48 MHz to 14.74 MHz.

```
>$C
E3
:EA
```

Note: It may be necessary to power the chip off and back on before the new clock source will take effect since some clock sources use an internal PLL that only resets at power up. You can check the clock speed that the chip is currently running at by using the Version command.

Checksum – display checksum value

The Checksum command displays a checksum for the uM-FPU V3.1 program code and user-defined functions stored in Flash. This can be used to check that the chip is valid, or that a particular set of user-defined functions is installed.

```
>$S:001AB76A
```

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	6	5	4	3	2	1	0
	B	R	T	I	S	P	Mode	

Bit 7 Break on Reset (if debug mode is enabled)
Bit 6 use `OUT1` pin for Ready/Busy status
Bit 5 Trace on Reset (if debug mode is enabled)
Bit 4 Idle Mode power saving enabled
Bit 3 Sleep Mode power saving enabled
Bit 2 PIC mode enabled (see PICMODE instruction)
Bits 1:0 Mode
00 – CS pin determines interface mode (default)
if CS pin = Low, SPI mode selected
if CS pin = High, I^2C mode selected
01 – I^2C mode selected
1x – SPI mode selected (CS pin used as chip select)

Byte 1: I^2C 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: Auto-Start Function
Mode parameter byte 2 now specifies a user-defined function that can optionally be called when the chip is Reset. Mode parameter byte 2 is only checked at Reset if the `CS` pin is Low. If both the CS pin and SERIN pin are High at Reset, Debug Mode will always be entered. To use auto-start with the I^2C interface, the `CS` pin must be Low at Reset, and the I^2C mode must be selected using mode 01 in mode parameter byte 0.

Bit	7	6	5	4	3	2	1	0
	D	F	Function					

Bit 7 Debug mode
0 - use SERIN to select debug mode
SERIN = Low, Disable debug mode
SERIN = High, Enable debug mode
1 - Disable debug mode
Bit 6 Auto-start function call
0 - No function called
1 - Call the function specified by bits 5:0
Bit 5:0 Function number

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 Instructions

There are several instructions that are designed to work in conjunction with the debug monitor. If the debug monitor is not enabled, these commands are `NOPs`. The instructions are as follows:

BREAK
When the `BREAK` instruction is encountered, execution stops, and the debug monitor is entered. Execution will only resume when a Go command is issued entered with the debug monitor.

TRACEOFF
Turns the debug trace mode off.

TRACEON
Turns the debug trace mode on. All instructions will be traced on the debug terminal until the trace mode is turned off by a `TRACEOFF` instruction 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.

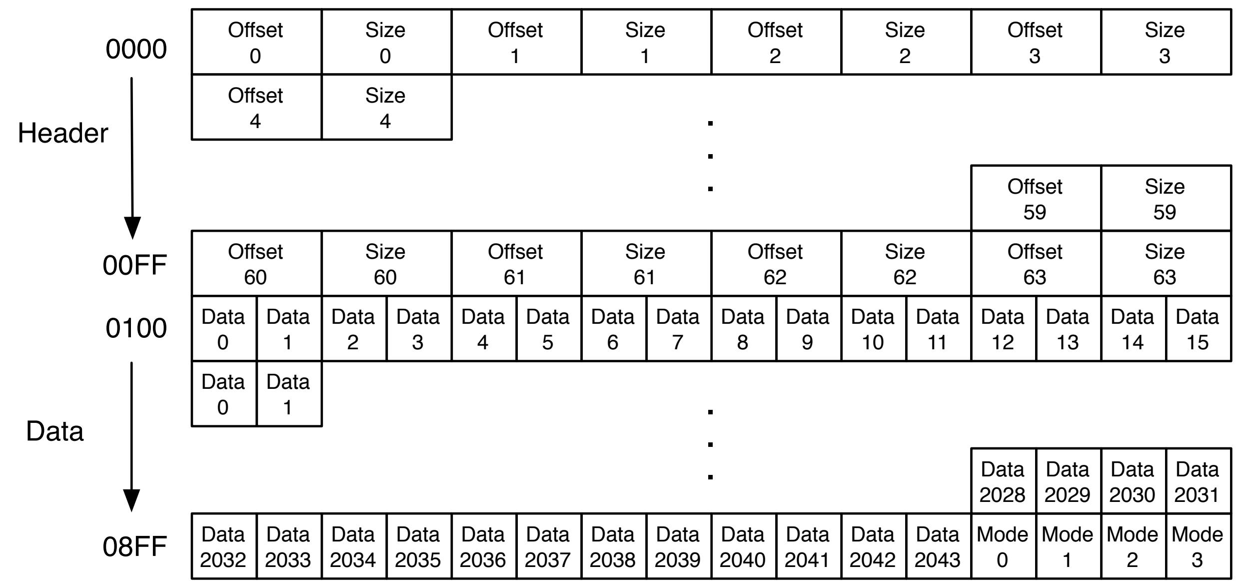
TRACEREG
Displays a trace string with the value of the register to the debug monitor output. Trace registers are always output; they are not affected by the trace mode.

Flash Memory

There are 2304 bytes of Flash memory reserved on the uM-FPU for storing user-defined functions and the mode parameters. Up to 64 user-defined functions can be stored in Flash memory. User-defined 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. `BRA`, `JMP`, `TABLE`, `POLY`) are only valid in user-defined functions. The `FCALL` instruction is used to call the user-defined functions stored in Flash memory. The Busy condition remains set while all of the instructions in the called function execute.

Flash memory for user-defined functions 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 16-bit words (256 bytes) that specify the offset to the data section and the length of the stored function. The data section consists of 2048 bytes and contains the user-defined function code. User-defined functions stored in Flash memory are programmed using the serial debug monitor. The uM-FPU V3 IDE (Integrated Development Environment) provides support for defining and programming user-defined functions. (Refer to uM-FPU V3 IDE documentation.)

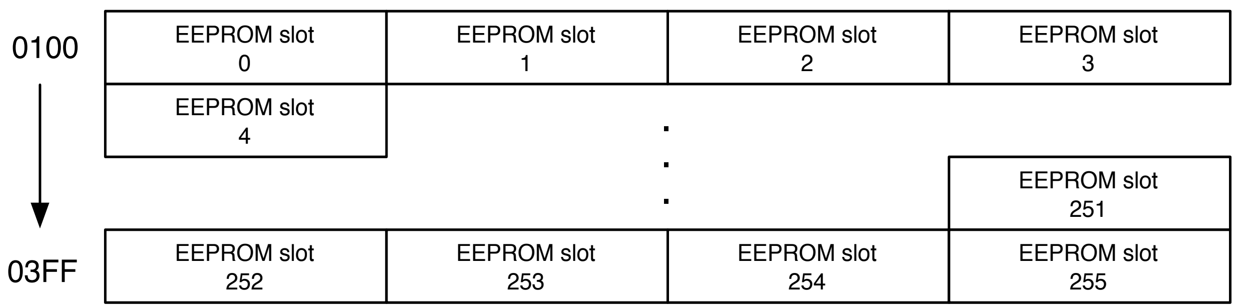
Flash Memory Layout



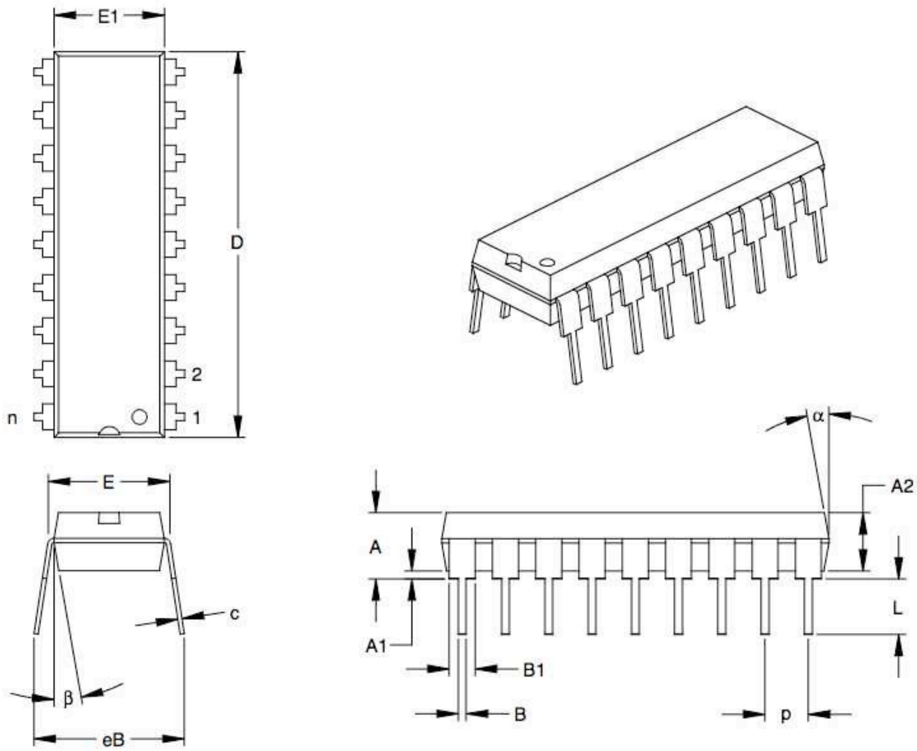
EEPROM Memory

There are 1024 bytes of EEPROM memory reserved on the uM-FPU for storing user-defined functions and data. The `EESAVE`, `EESAVEA`, `EELOAD`, `EELOADA` instructions are used to store and retrieve data. The `EEWRITE` instruction is used to store user-defined functions at run-time. The `ECALL` instruction is used to call the user-defined functions stored in EEPROM memory. The Busy condition remains set while all of the instructions in the called function execute. When storing a user-defined function in EEPROM, the first byte of an EEPROM slot must contain the length of the user-defined function, and the last byte must be a RET instruction. This is used as a validity check for user-defined functions before the code stored in EEPROM is executed. User-defined functions in EEPROM are restricted to a total length of 256 bytes. Care should be taken to keep track of how much space is used by a user-defined functions so that it doesn't overlap any slots used for data storage.

EEPROM Memory Layout



PDIP-18 Through-Hole Package

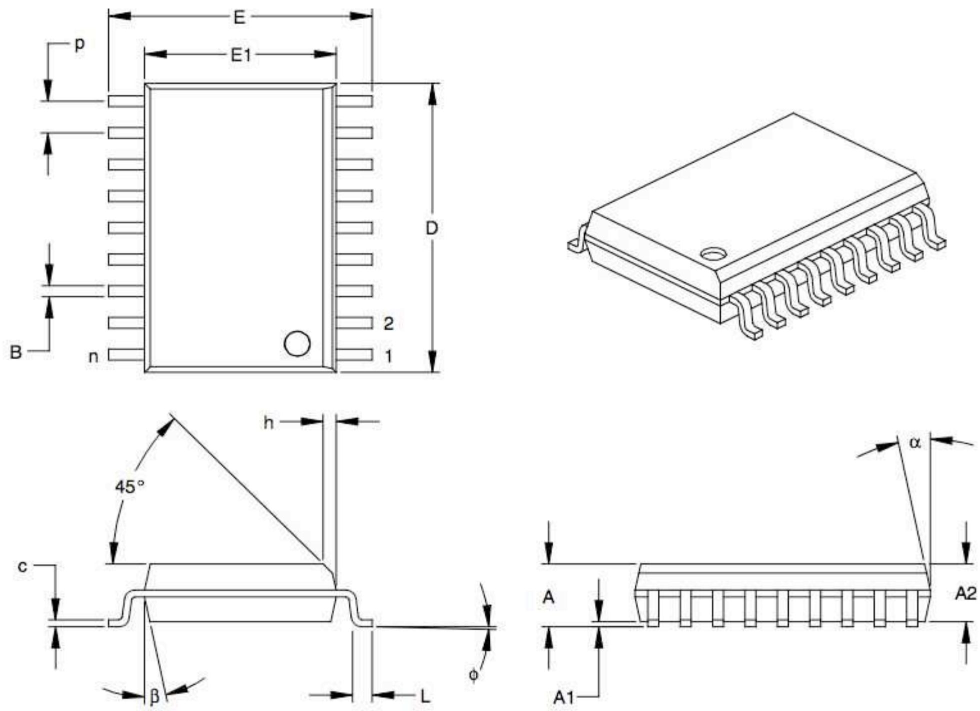


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		18			18	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.140	.155	.170	3.56	3.94	4.32
Molded Package Thickness	A2	.115	.130	.145	2.92	3.30	3.68
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.300	.313	.325	7.62	7.94	8.26
Molded Package Width	E1	.240	.250	.260	6.10	6.35	6.60
Overall Length	D	.890	.898	.905	22.61	22.80	22.99
Tip to Seating Plane	L	.125	.130	.135	3.18	3.30	3.43
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38
Upper Lead Width	B1	.045	.058	.070	1.14	1.46	1.78
Lower Lead Width	B	.014	.018	.022	0.36	0.46	0.56
Overall Row Spacing §	eB	.310	.370	.430	7.87	9.40	10.92
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MS-001
Drawing No. C04-007

SOIC-18 Surface Mount Package

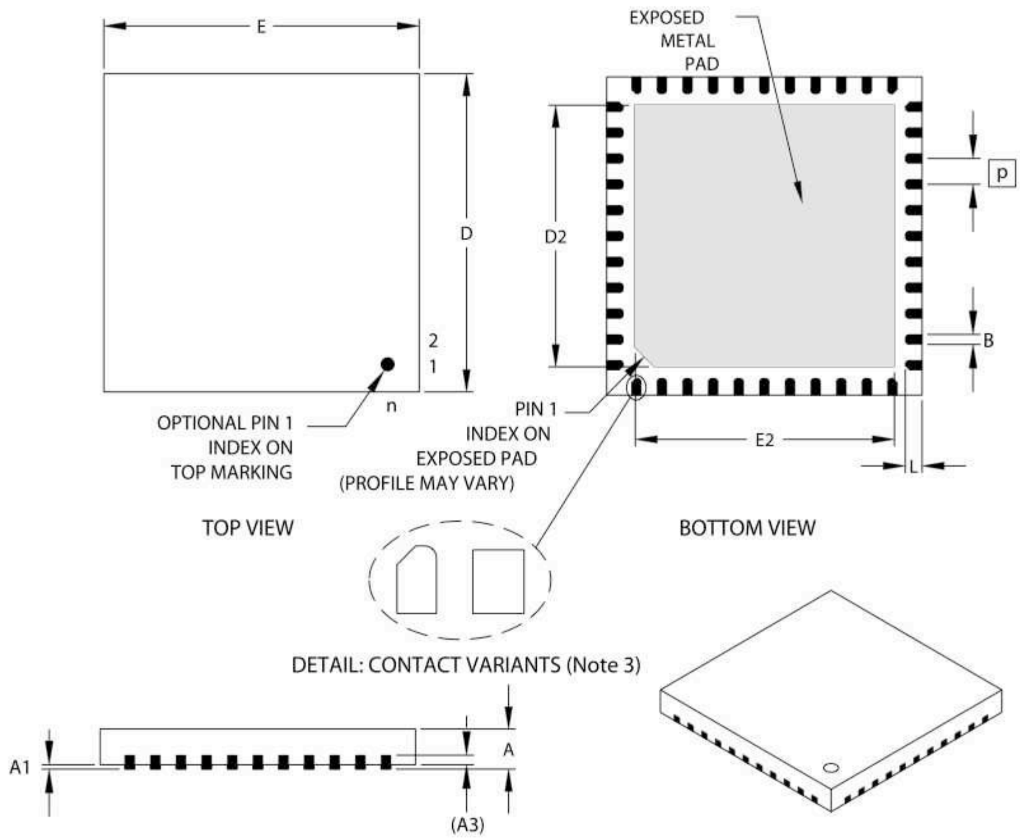


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		18			18	
Pitch	p		.050			1.27	
Overall Height	A	.093	.099	.104	2.36	2.50	2.64
Molded Package Thickness	A2	.088	.091	.094	2.24	2.31	2.39
Standoff §	A1	.004	.008	.012	0.10	0.20	0.30
Overall Width	E	.394	.407	.420	10.01	10.34	10.67
Molded Package Width	E1	.291	.295	.299	7.39	7.49	7.59
Overall Length	D	.446	.454	.462	11.33	11.53	11.73
Chamfer Distance	h	.010	.020	.029	0.25	0.50	0.74
Foot Length	L	.016	.033	.050	0.41	0.84	1.27
Foot Angle	φ	0	4	8	0	4	8
Lead Thickness	c	.009	.011	.012	0.23	0.27	0.30
Lead Width	B	.014	.017	.020	0.36	0.42	0.51
Mold Draft Angle Top	α	0	12	15	0	12	15
Mold Draft Angle Bottom	β	0	12	15	0	12	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MS-013
Drawing No. C04-051

QFN-44 Surface Mount Package



Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Contacts	n		44			44	
Pitch	p		.026 BSC [1]			0.65 BSC [1]	
Overall Height	A	.031	.035	.039	0.80	0.90	1.00
Standoff	A1	.000	.001	.002	0	0.02	0.05
Base Thickness	(A3)		.010 REF [2]			0.25 REF [2]	
Overall Width	E	.309	.315	.321	7.85	8.00	8.15
Exposed Pad Width	E2	.246	.268	.274	6.25	6.80	6.95
Overall Length	D	.309	.315	.321	7.85	8.00	8.15
Exposed Pad Length	D2	.246	.268	.274	6.25	6.80	6.95
Contact Width	B	.008	.013	.013	0.20	0.33	0.35
Contact Length	L	.014	.016	.019	0.35	0.40	0.48

*Controlling Parameter

Notes:

1. BSC: Basic Dimension. Theoretically exact value shown without tolerances. See ASME Y14.5M
2. REF: Reference Dimension, usually without tolerance, for information purposes only. See ASME Y14.5M
3. Contact profiles may vary.

JEDEC equivalent: M0-220

Drawing No. C04-103

Absolute Maximum Ratings

Parameter	Minimum	Typical	Maximum	Units
Storage Temperature	-65	-	+150	° Celsius
Ambient Temperature with Power Applied	-40	-	+85	° Celsius
Supply Voltage on VDD relative to VSS	-0.3	-	+5.5	V
Input Voltage relative to VSS	-0.3	-	VDD+0.3	V
Maximum Current out of VSS pin			300	mA
Maximum Current into VDD pin			250	mA
Maximum Current sourced by any I/O pin			25	mA
Maximum Current sinked by any I/O pin			25	mA
Maximum Current sourced by all I/O pins			200	mA
Maximum Current sinked by all I/O pins			200	mA

DC Characteristics

Parameter	Minimum	Typical	Maximum	Units
I/O Pin Input Low Voltage	VSS	-	0.2 VDD	V
I/O Pin Input High Voltage	0.8 VDD	-	VDD	V
AVDD	greater of VDD - 0.3 or 2.7		lesser of VDD + 0.3 or 5.5	V
AVSS	VSS - 0.3		VSS + 0.3	
Operating MIPS at 4.5 to 5.5 VDD			30	MIPS
Operating MIPS at 3.0 to 3.6 VDD			15	MIPS
Operating MIPS at 2.5 to 3 VDD			7.5	MIPS
Recommended 5V Operating Range (VDD)	4.75	-	5.25	V
Supply Current	-	TBD	-	mA

Further Information

Check the Micromega website at www.micromegacorp.com

Appendix A
uM-FPU V3.1 Instruction Summary

Instruction	Opcode	Arguments	Returns	Description
NOP	00			No Operation
SELECTA	01	nn		Select register A
SELECTX	02	nn		Select register X
CLR	03	nn		reg[nn] = 0
CLRA	04			reg[A] = 0
CLRX	05			reg[X] = 0, X = X + 1
CLR0	06			reg[0] = 0
COPY	07	mm,nn		reg[nn] = reg[mm]
COPYA	08	nn		reg[nn] = reg[A]
COPYX	09	nn		reg[nn] = reg[X], X = X + 1
LOAD	0A	nn		reg[0] = reg[nn]
LOADA	0B			reg[0] = reg[A]
LOADX	0C			reg[0] = reg[X], X = X + 1
ALOADX	0D			reg[A] = reg[X], X = X + 1
XSAVE	0E	nn		reg[X] = reg[nn], X = X + 1
XSAVEA	0F			reg[X] = reg[A], X = X + 1
COPY0	10	nn		reg[nn] = reg[0]
COPYI	11	bb,nn		reg[nn] = long(unsigned byte bb)
SWAP	12	nn,mm		Swap reg[nn] and reg[mm]
SWAPA	13	nn		Swap reg[nn] and reg[A]
LEFT	14			Left parenthesis
RIGHT	15			Right parenthesis
FWRITE	16	nn,b1,b2,b3,b4		Write 32-bit floating point to reg[nn]
FWRITEA	17	b1,b2,b3,b4		Write 32-bit floating point to reg[A]
FWRITEX	18	b1,b2,b3,b4		Write 32-bit floating point to reg[X]
FWRITE0	19	b1,b2,b3,b4		Write 32-bit floating point to reg[0]
FREAD	1A	nn	b1,b2,b3,b4	Read 32-bit floating point from reg[nn]
FREADA	1B		b1,b2,b3,b4	Read 32-bit floating point from reg[A]
FREADX	1C		b1,b2,b3,b4	Read 32-bit floating point from reg[X]
FREAD0	1D		b1,b2,b3,b4	Read 32-bit floating point from reg[0]
ATOF	1E	aa…00		Convert ASCII to floating point
FTOA	1F	bb		Convert floating point to ASCII
FSET	20	nn		reg[A] = reg[nn]
FADD	21	nn		reg[A] = reg[A] + reg[nn]
FSUB	22	nn		reg[A] = reg[A] - reg[nn]
FSUBR	23	nn		reg[A] = reg[nn] - reg[A]
FMUL	24	nn		reg[A] = reg[A] * reg[nn]
FDIV	25	nn		reg[A] = reg[A] / reg[nn]
FDIVR	26	nn		reg[A] = reg[nn] / reg[A]
FPOW	27	nn		reg[A] = reg[A] ** reg[nn]
FCMP	28	nn		Compare reg[A], reg[nn], Set floating point status
FSET0	29			reg[A] = reg[0]
FADD0	2A			reg[A] = reg[A] + reg[0]
FSUB0	2B			reg[A] = reg[A] - reg[0]

FSUBR0	2C			reg[A] = reg[0] - reg[A]
FMUL0	2D			reg[A] = reg[A] * reg[0]
FDIV0	2E			reg[A] = reg[A] / reg[0]
FDIVR0	2F			reg[A] = reg[0] / reg[A]
FPOW0	30			reg[A] = reg[A] ** reg[0]
FCMP0	31			Compare reg[A], reg[0], Set floating point status
FSETI	32	bb		reg[A] = float(bb)
FADDI	33	bb		reg[A] = reg[A] - float(bb)
FSUBI	34	bb		reg[A] = reg[A] - float(bb)
FSUBRI	35	bb		reg[A] = float(bb) - reg[A]
FMULI	36	bb		reg[A] = reg[A] * float(bb)
FDIVI	37	bb		reg[A] = reg[A] / float(bb)
FDIVRI	38	bb		reg[A] = float(bb) / reg[A]
FPOWI	39	bb		reg[A] = reg[A] ** bb
FCMPI	3A	bb		Compare reg[A], float(bb), Set floating point status
FSTATUS	3B	nn		Set floating point status for reg[nn]
FSTATUSA	3C			Set floating point status for reg[A]
FCMP2	3D	nn,mm		Compare reg[nn], reg[mm] Set floating point status
FNEG	3E			reg[A] = -reg[A]
FABS	3F			reg[A] = I reg[A] I
FINV	40			reg[A] = 1 / reg[A]
SQRT	41			reg[A] = sqrt(reg[A])
ROOT	42	nn		reg[A] = root(reg[A], reg[nn])
LOG	43			reg[A] = log(reg[A])
LOG10	44			reg[A] = log10(reg[A])
EXP	45			reg[A] = exp(reg[A])
EXP10	46			reg[A] = exp10(reg[A])
SIN	47			reg[A] = sin(reg[A])
COS	48			reg[A] = cos(reg[A])
TAN	49			reg[A] = tan(reg[A])
ASIN	4A			reg[A] = asin(reg[A])
ACOS	4B			reg[A] = acos(reg[A])
ATAN	4C			reg[A] = atan(reg[A])
ATAN2	4D	nn		reg[A] = atan2(reg[A], reg[nn])
DEGREES	4E			reg[A] = degrees(reg[A])
RADIANS	4F			reg[A] = radians(reg[A])
FMOD	50	nn		reg[A] = reg[A] MOD reg[nn]
FLOOR	51			reg[A] = floor(reg[A])
CEIL	52			reg[A] = ceil(reg[A])
ROUND	53			reg[A] = round(reg[A])
FMIN	54	nn		reg[A] = min(reg[A], reg[nn])
FMAX	55	nn		reg[A] = max(reg[A], reg[nn])
FCNV	56	bb		reg[A] = conversion(bb, reg[A])
FMAC	57	nn,mm		reg[A] = reg[A] + (reg[nn] * reg[mm])
FMSC	58	nn,mm		reg[A] = reg[A] - (reg[nn] * reg[mm])
LOADBYTE	59	bb		reg[0] = float(signed bb)

LOADUBYTE	5A	bb		reg[0] = float(unsigned byte)
LOADWORD	5B	b1,b2		reg[0] = float(signed b1*256 + b2)
LOADUWORD	5C	b1,b2		reg[0] = float(unsigned b1*256 + b2)
LOADE	5D			reg[0] = 2.7182818
LOADPI	5E			reg[0] = 3.1415927
LOADCON	5F	bb		reg[0] = float constant(bb)
FLOAT	60			reg[A] = float(reg[A])
FIX	61			reg[A] = fix(reg[A])
FIXR	62			reg[A] = fix(round(reg[A]))
FRAC	63			reg[A] = fraction(reg[A])
FSPLIT	64			reg[A] = integer(reg[A]), reg[0] = fraction(reg[A])
SELECTMA	65	nn,bb,bb		Select matrix A
SELECTMB	66	nn,bb,bb		Select matrix B
SELECTMC	67	nn,bb,bb		Select matrix C
LOADMA	68	bb,bb		reg[0] = Matrix A[bb, bb]
LOADMB	69	bb,bb		reg[0] = Matrix B[bb, bb]
LOADMC	6A	bb,bb		reg[0] = Matrix C[bb, bb]
SAVEMA	6B	bb,bb		Matrix A[bb, bb] = reg[A]
SAVEMB	6C	bb,bb		Matrix B[bb, bb] = reg[A]
SAVEMC	6D	bb,bb		Matrix C[bb, bb] = reg[A]
MOP	6E	bb		Matrix/Vector operation
FFT	6F	bb		Fast Fourier Transform
WRBLK	70	tc,t1…tn		Write multiple 32-bit values
RDBLK	71	tc	t1…tn	Read multiple 32-bit values
LOADIND	7A	nn		reg[0] = reg[reg[nn]]
SAVEIND	7B	nn		reg[reg[nn]] = reg[A]
INDA	7C	nn		Select register A using value in reg[nn]
INDX	7D	nn		Select register X using value in reg[nn]
FCALL	7E	fn		Call user-defined function in Flash
EECALL	7F	fn		Call user-defined function in EEPROM
RET	80			Return from user-defined function
BRA	81	bb		Unconditional branch
BRA,cc	82	cc,bb		Conditional branch
JMP	83	b1,b2		Unconditional jump
JMP,cc	84	cc,b1,b2		Conditional jump
TABLE	85	tc,t1…tn		Table lookup
FTABLE	86	cc,tc,t1…tn		Floating point reverse table lookup
LTABLE	87	cc,tc,t1…tn		Long integer reverse table lookup
POLY	88	tc,t1…tn		reg[A] = nth order polynomial
GOTO	89	nn		Computed GOTO
RET,cc	8A	cc		Conditional return from user-defined function
LWRITE	90	nn,b1,b2,b3,b4		Write 32-bit long integer to reg[nn]
LWRITEA	91	b1,b2,b3,b4		Write 32-bit long integer to reg[A]
LWRITEX	92	b1,b2,b3,b4		Write 32-bit long integer to reg[X], X = X + 1
LWRITE0	93	b1,b2,b3,b4		Write 32-bit long integer to reg[0]
LREAD	94	nn	b1,b2,b3,b4	Read 32-bit long integer from reg[nn]

LREADA	95		b1,b2,b3,b4	Read 32-bit long value from reg[A]
LREADX	96		b1,b2,b3,b4	Read 32-bit long integer from reg[X], X = X + 1
LREAD0	97		b1,b2,b3,b4	Read 32-bit long integer from reg[0]
LREADBYTE	98		bb	Read lower 8 bits of reg[A]
LREADWORD	99		b1,b2	Read lower 16 bits reg[A]
ATOL	9A	aa…00		Convert ASCII to long integer
LTOA	9B	bb		Convert long integer to ASCII
LSET	9C	nn		reg[A] = reg[nn]
LADD	9D	nn		reg[A] = reg[A] + reg[nn]
LSUB	9E	nn		reg[A] = reg[A] - reg[nn]
LMUL	9F	nn		reg[A] = reg[A] * reg[nn]
LDIV	A0	nn		reg[A] = reg[A] / reg[nn] reg[0] = remainder
LCMP	A1	nn		Signed compare reg[A] and reg[nn], Set long integer status
LUDIV	A2	nn		reg[A] = reg[A] / reg[nn] reg[0] = remainder
LUCMP	A3	nn		Unsigned compare reg[A] and reg[nn], Set long integer status
LTST	A4	nn		Test reg[A] AND reg[nn], Set long integer status
LSET0	A5			reg[A] = reg[0]
LADD0	A6			reg[A] = reg[A] + reg[0]
LSUB0	A7			reg[A] = reg[A] - reg[0]
LMUL0	A8			reg[A] = reg[A] * reg[0]
LDIV0	A9			reg[A] = reg[A] / reg[0] reg[0] = remainder
LCMP0	AA			Signed compare reg[A] and reg[0], set long integer status
LUDIV0	AB			reg[A] = reg[A] / reg[0] reg[0] = remainder
LUCMP0	AC			Unsigned compare reg[A] and reg[0], Set long integer status
LTST0	AD			Test reg[A] AND reg[0], Set long integer status
LSETI	AE	bb		reg[A] = long(bb)
LADDI	AF	bb		reg[A] = reg[A] + long(bb)
LSUBI	B0	bb		reg[A] = reg[A] - long(bb)
LMULI	B1	bb		reg[A] = reg[A] * long(bb)
LDIVI	B2	bb		reg[A] = reg[A] / long(bb) reg[0] = remainder
LCMPI	B3	bb		Signed compare reg[A] - long(bb), Set long integer status
LUDIVI	B4	bb		reg[A] = reg[A] / unsigned long(bb) reg[0] = remainder
LUCMPI	B5	bb		Unsigned compare reg[A] and long(bb), Set long integer status
LTSTI	B6	bb		Test reg[A] AND long(bb), Set long integer status

LSTATUS	B7	nn		Set long integer status for reg[nn]
LSTATUSA	B8			Set long integer status for reg[A]
LCMP2	B9	nn,mm		Signed long compare reg[nn], reg[mm] Set long integer status
LUCMP2	BA	nn,mm		Unsigned long compare reg[nn], reg[mm] Set long integer status
LNEG	BB			reg[A] = -reg[A]
LABS	BC			reg[A] = I reg[A] I
LINC	BD	nn		reg[nn] = reg[nn] + 1, set status
LDEC	BE	nn		reg[nn] = reg[nn] - 1, set status
LNOT	BF			reg[A] = NOT reg[A]
LAND	C0	nn		reg[A] = reg[A] AND reg[nn]
LOR	C1	nn		reg[A] = reg[A] OR reg[nn]
LXOR	C2	nn		reg[A] = reg[A] XOR reg[nn]
LSHIFT	C3	nn		reg[A] = reg[A] shift reg[nn]
LMIN	C4	nn		reg[A] = min(reg[A], reg[nn])
LMAX	C5	nn		reg[A] = max(reg[A], reg[nn])
LONGBYTE	C6	bb		reg[0] = long(signed byte bb)
LONGUBYTE	C7	bb		reg[0] = long(unsigned byte bb)
LONGWORD	C8	b1,b2		reg[0] = long(signed b1*256 + b2)
LONGUWORD	C9	b1,b2		reg[0] = long(unsigned b1*256 + b2)
SETSTATUS	CD	ss		Set status byte
SEROUT	CE	bb bb,bd bb,aa…00		Serial output
SERIN	CF	bb		Serial input
SETOUT	D0	bb		Set OUT1 and OUT2 output pins
ADCMODE	D1	bb		Set A/D trigger mode
ADCTRIG	D2			A/D manual trigger
ADCSCALE	D3	ch		ADCscale[ch] = reg[0]
ADCLONG	D4	ch		reg[0] = ADCvalue[ch]
ADCLOAD	D5	ch		reg[0] = float(ADCvalue[ch]) * ADCscale[ch]
ADCWAIT	D6			wait for next A/D sample
TIMESET	D7			time = reg[0]
TIMELONG	D8			reg[0] = time (long integer)
TICKLONG	D9			reg[0] = ticks (long integer)
EESAVE	DA	nn,ee		EEPROM[ee] = reg[nn]
EESAVEA	DB	ee		EEPROM[ee] = reg[A]
EELOAD	DC	nn,ee		reg[nn] = EEPROM[ee]
EELOADA	DD	ee		reg[A] = EEPROM[ee]
EEWRITE	DE	ee,bc,b1…bn		Store bytes starting at EEPROM[ee]
EXTSET	E0			external input count = reg[0]
EXTLONG	E1			reg[0] = external input counter
EXTWAIT	E2			wait for next external input
STRSET	E3	aa…00		Copy string to string buffer
STRSEL	E4	bb,bb		Set selection point
STRINS	E5	aa…00		Insert string at selection point
STRCMP	E6	aa…00		Compare string with string selection

STRFIND	E7	aa…00		Find string
STRFCHR	E8	aa…00		Set field separators
STRFIELD	E9	bb		Find field
STRTOF	EA			Convert string selection to floating point
STRTOL	EB			Convert string selection to long integer
READSEL	EC		aa…00	Read string selection
STRBYTE	ED	bb		Insert byte at selection point
STRINC	EE			Increment string selection point
STRDEC	EF			Decrement string selection point
SYNC	F0		5C	Get synchronization byte
READSTATUS	F1		ss	Read status byte
READSTR	F2		aa…00	Read string from string buffer
VERSION	F3			Copy version string to string buffer
IEEEMODE	F4			Set IEEE mode (default)
PICMODE	F5			Set PIC mode
CHECKSUM	F6			Calculate checksum for uM-FPU code
BREAK	F7			Debug breakpoint
TRACEOFF	F8			Turn debug trace off
TRACEON	F9			Turn debug trace on
TRACESTR	FA	aa…00		Send string to debug trace buffer
TRACEREG	FB	nn		Send register value to trace buffer
READVAR	FC	bb		Read internal register value
RESET	FF			Reset (9 consecutive FF bytes cause a reset, otherwise it is a NOP)

Notes:

Opcode	Instruction opcode in hexadecimal
Arguments	Additional data required by instruction
Returns	Data returned by instruction
nn	register number (0-127)
mm	register number (0-127)
fn	function number (0-63)
bb	8-bit value
b1,b2	16-bit value (b1 is MSB)
b1,b2,b3,b4	32-bit value (b1 is MSB)
b1…bn	string of 8-bit bytes
ss	Status byte
bd	baud rate and debug mode
cc	Condition code
ee	EEPROM address slot (0-255)
ch	A/D channel number
bc	Byte count
tc	32-bit value count
t1…tn	String of 32-bit values
aa…00	Zero terminated ASCII string

Appendix B
uM-FPU V3.1 Instruction Timing

The instruction times shown in the following table are calculated with a clock speed of 29.48 MHz and are measured from the rising edge of the last bit of the last byte of the instruction (SIN pin) to the Ready state being asserted (falling edge on SOUT). The instruction times do not include the transfer time for sending the instructions to the uM-FPU, which depends on the type of interface (e.g. SPI or I^2C), and the speed of the interface.

The uM-FPU V3.1 chip contains a 256 byte instruction buffer that can be used to minimize the transfer time. Instructions can be queued up in the instruction buffer while previous instructions are executing, allowing the transfer time to overlap the instruction execution time.

User-defined functions can also be stored in Flash memory on the uM-FPU V3.1 chip, which is another option for eliminating the transfer time.

If debug tracing is enabled, the Ready state is delayed once the trace buffer is full. Trace data is output through the SEROUT pin at 57,600 baud. On average, each byte of data in an instruction generates approximately three trace characters, which requires about 521 microseconds to transmit. Once the trace buffer is full, instruction execution is delayed until space is available. When using a fast interface, trace delays can be a dominant part of the overall instruction execution time.

Instruction	Opcode	Arguments	Returns	Execution Time (usec)	Notes
NOP	00			6	
SELECTA	01	nn		4	
SELECTX	02	nn		4	
CLR	03	nn		5	
CLRA	04			7	
CLRX	05			7	
CLR0	06			7	
COPY	07	mm,nn		5	
COPYA	08	nn		5	
COPYX	09	nn		5	
LOAD	0A	nn		5	
LOADA	0B			7	
LOADX	0C			7	
ALOADX	0D			7	
XSAVE	0E	nn		5	
XSAVEA	0F			7	
COPY0	10	nn		5	
COPYI	11	bb,nn		5	
SWAP	12	nn,mm		6	
SWAPA	13	nn		6	
LEFT	14			7	
RIGHT	15			7	
FWRITE	16	nn,b1,b2,b3,b4		5	
FWRITEA	17	b1,b2,b3,b4		5	
FWRITEX	18	b1,b2,b3,b4		5	
FWRITE0	19	b1,b2,b3,b4		5	
FREAD	1A	nn	b1,b2,b3,b4		(note 1)
FREADA	1B		b1,b2,b3,b4		(note 1)

FREADX	1C		b1,b2,b3,b4		(note 1)
FREAD0	1D		b1,b2,b3,b4		(note 1)
ATOF	1E	aa…00		26-90	(note 5)
FTOA	1F	bb		8-250	(note 6)
FSET	20	nn		5	
FADD	21	nn		9-14	(note 2)
FSUB	22	nn		10-15	(note 2)
FSUBR	23	nn		10-15	(note 2)
FMUL	24	nn		9	
FDIV	25	nn		17-18	(note 2)
FDIVR	26	nn		17-18	(note 2)
FPOW	27	nn		5-272	(note 2)
FCMP	28	nn		7	
FSET0	29			5	
FADD0	2A			11-16	(note 2)
FSUB0	2B			12-17	(note 2)
FSUBR0	2C			12-17	(note 2)
FMUL0	2D			11	
FDIV0	2E			19-20	(note 2)
FDIVR0	2F			19-20	(note 2)
FPOW0	30			8-274	(note 2)
FCMP0	31			8	
FSETI	32	bb		10-12	
FADDI	33	bb		15-18	(note 2)
FSUBI	34	bb		15-19	(note 2)
FSUBRI	35	bb		15-19	(note 2)
FMULI	36	bb		14-15	(note 2)
FDIVI	37	bb		23-25	(note 2)
FDIVRI	38	bb		23-25	(note 2)
FPOWI	39	bb		5-47	(note 2)
FCMPI	3A	bb		13	
FSTATUS	3B	nn		5	
FSTATUSA	3C			6	
FCMP2	3D	nn,mm		7	
FNEG	3E			7	
FABS	3F			7	
FINV	40			20-21	(note 2)
SQRT	41			23-24	(note 2)
ROOT	42	nn		25-286	
LOG	43			108-110	(note 2)
LOG10	44			112-144	(note 2)
EXP	45			98-110	(note 4)
EXP10	46			98-144	(note 4)
SIN	47			90-100	(note 2)
COS	48			108-110	(note 2)
TAN	49			103	(note 2)
ASIN	4A			72-101	(note 11)
ACOS	4B			77-96	(note 11)
ATAN	4C			62-101	(note 11)

ATAN2	4D	nn		114-127	(note 11)
DEGREES	4E			10-11	(note 2)
RADIANS	4F			10-11	(note 2)
FMOD	50	nn		7-11	(note 2)
FLOOR	51			8-10	(note 2)
CEIL	52			10-11	(note 2)
ROUND	53			17-25	(note 2)
FMIN	54	nn		6-7	(note 2)
FMAX	55	nn		6-7	(note 2)
FCNV	56	bb		9-23	(note 2)
FMAC	57	nn,mm		16	
FMSC	58	nn,mm		16	
LOADBYTE	59	bb		10	
LOADUBYTE	5A	bb		10	
LOADWORD	5B	b1,b2		10	
LOADUWORD	5C	b1,b2		10	
LOADE	5D			7	
LOADPI	5E			7	
LOADCON	5F	bb		5	
FLOAT	60			10-12	(note 3)
FIX	61			7-10	(note 2)
FIXR	62			18-26	(note 2)
FRAC	63			20	
FSPLIT	64			21	
SELECTMA	65	nn,bb,bb		4	
SELECTMB	66	nn,bb,bb		4	
SELECTMC	67	nn,bb,bb		4	
LOADMA	68	bb,bb		5	
LOADMB	69	bb,bb		5	
LOADMC	6A	bb,bb		5	
SAVEMA	6B	bb,bb		5	
SAVEMB	6C	bb,bb		5	
SAVEMC	6D	bb,bb		5	
MOP	6E	bb			(note 17)
FFT	6F	bb			(note 15)
WRBLK	70	tc,t1…tn			(note 16)
RDBLK	71	tc	t1…tn		(note 16)
LOADIND	7A	nn		5	
SAVEIND	7B	nn		5	
INDA	7C	nn		5	
INDX	7D	nn		5	
FCALL	7E	fn		5	(note 7)
EECALL	7F	fn		13	(note 7)
RET	80			5	(note 8)
BRA	81	bb		6	(note 8)
BRA,cc	82	cc,bb		2-4	(note 8)
JMP	83	b1,b2		7	(note 8)
JMP,cc	84	cc,b1,b2		5	(note 8)
TABLE	85	tc,t1…tn		11	(note 8)

FTABLE	86	cc,tc,t1…tn		25	(note 8)
LTABLE	87	cc,tc,t1…tn		23	(note 8)
POLY	88	tc,t1…tn			(note 8, 9)
GOTO	89	nn		7	(note 8)
RET,cc	8A	cc		5	(note 8)
LWRITE	90	nn,b1,b2,b3,b4		5	
LWRITEA	91	b1,b2,b3,b4		5	
LWRITEX	92	b1,b2,b3,b4		5	
LWRITE0	93	b1,b2,b3,b4		5	
LREAD	94	nn	b1,b2,b3,b4		(note 1)
LREADA	95		b1,b2,b3,b4		(note 1)
LREADX	96		b1,b2,b3,b4		(note 1)
LREAD0	97		b1,b2,b3,b4		(note 1)
LREADBYTE	98		bb		(note 1)
LREADWORD	99		b1,b2		(note 1)
ATOL	9A	aa…00		10-30	
LTOA	9B	bb		20-165	(note 6)
LSET	9C	nn		5	
LADD	9D	nn		5	(note 3)
LSUB	9E	nn		5	(note 3)
LMUL	9F	nn		5	(note 3)
LDIV	A0	nn		22	(note 3)
LCMP	A1	nn		5	
LUDIV	A2	nn		21	(note 3)
LUCMP	A3	nn		5	
LTST	A4	nn		5	
LSET0	A5			7	
LADD0	A6			7	
LSUB0	A7			7	
LMUL0	A8			7	
LDIV0	A9			23	
LCMP0	AA			6	
LUDIV0	AB			22	
LUCMP0	AC			6	
LTST0	AD			6	
LSETI	AE	bb		5	
LADDI	AF	bb		5	
LSUBI	B0	bb		5	
LMULI	B1	bb		5	
LDIVI	B2	bb		21	
LCMPI	B3	bb		5	
LUDIVI	B4	bb		21	
LUCMPI	B5	bb		5	
LTSTI	B6	bb		4	
LSTATUS	B7	nn		4	
LSTATUSA	B8			6	
LCMP2	B9	nn,mm		5	
LUCMP2	BA	nn,mm		5	
LNEG	BB			6	

LABS	BC			6	
LINC	BD	nn		5	
LDEC	BE	nn		5	
LNOT	BF			7	
LAND	C0	nn		5	
LOR	C1	nn		5	
LXOR	C2	nn		5	
LSHIFT	C3	nn		5-11	
LMIN	C4	nn		4-5	(note 3)
LMAX	C5	nn		4-5	(note 3)
LONGBYTE	C6	bb		5	
LONGUBYTE	C7	bb		5	
LONGWORD	C8	b1,b2		5	
LONGUWORD	C9	b1,b2		5	
SETSTATUS	CD	ss		4	
SEROUT	CE	bb bb,bd bb,aa…00			(note 14)
SERIN	CF	bb			(note 14)
SETOUT	D0	bb		5	
ADCMODE	D1	bb		6-7	
ADCTRIG	D2			9	
ADCSCALE	D3	ch		6	
ADCLONG	D4	ch		8	
ADCLOAD	D5	ch		17	
ADCWAIT	D6				(note 11)
TIMESET	D7			9	
TIMELONG	D8			10	
TICKLONG	D9			10	
EESAVE	DA	nn,ee		5590	
EESAVEA	DB	ee		5590	
EELOAD	DC	nn,ee		5	
EELOADA	DD	ee		5	
EEWRITE	DE	ee,bc,b1…bn		1120/byte	
EXTSET	E0			9	
EXTLONG	E1			10	
EXTWAIT	E2				(note 11)
STRSET	E3	aa…00		5	
STRSEL	E4	bb,bb		6	
STRINS	E5	aa…00		5	
STRCMP	E6	aa…00		4-10	
STRFIND	E7	aa…00		7	
STRFCHR	E8	aa…00		5	
STRFIELD	E9	bb		10	
STRTOF	EA			26-90	
STRTOL	EB			10-50	
READSEL	EC		aa…00		(note 1)
STRBYTE	ED	bb			
STRINC	EE				

STRDEC	EF				
SYNC	F0		5C		(note 1)
READSTATUS	F1		ss		(note 1)
READSTR	F2		aa…00		(note 1)
VERSION	F3			9	
IEEEMODE	F4			5	
PICMODE	F5			5	
CHECKSUM	F6			3888	
BREAK	F7				(note 12)
TRACEOFF	F8			20	
TRACEON	F9			22	
TRACESTR	FA	aa…00		8	
TRACEREG	FB	nn		28	
READVAR	FC	bb		5	
RESET	FF				(note 13)

Notes:

1. The minimum Read Setup Delay must occur after all opcodes that return data. See the SPI or I^2C instruction timing diagrams for details.
2. Floating point values 1000.0 and 0.001 used for timing.
3. Long integer values 100 and 100000 used for timing.
4. Floating point values 30.0 and 0.001 used for timing.
5. Strings 1.2, 1.23, 1.234, … 1.234567 used for timing.
6. The timing depends on the register value and format specified.
7. The timing depends on the user defined function specified.
8. Instruction only valid in Flash memory.
9. Approximately (20 + 15 * order of the polynomial) microseconds.
10. Floating point values 0.25 and 0.75 used for timing.
11. Busy state is held indefinitely until condition is met.
12. Busy state is held indefinitely until user continues execution from debugger.
13. After 9 consecutive FF bytes the chip is reset, otherwise it is a NOP.
14. Depends baud rate, number of characters and operation.
15. The FFT instruction can do up to 64 point FFTs on-chip. The calculation times for these are as follows:
    2 point: 43 usec
    4 point: 175 usec
    8 point: 538 usec
    16 point: 1462 usec
    32 point: 3667 usec
    64 point: 8703 usec
    If the data is on the microprocessor, then read/write data transfer times must be added. For larger FFTs, the FFT instruction is a multi-stage calculation.
16. Depends on the transfer speed of the microcontroller.
17. Depends on size of matrix and type of operation.