Micromega Corporation

uM-FPU V3 Datasheet

32-bit Floating Point Coprocessor
*** PRELIMINARY - beta 3 ***

Introduction

The uM-FPU V3 floating point coprocessor that can be easily interfaced with a variety of 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 I^2C or SPI compatible interface.

New Features in uM-FPU V3

- 10 to 20 times faster than uMFPU V2 for all floating point operations
- up to 70 times faster for advanced instructions
- supports 2.7V, 3.3V and 5V supply voltage
- I^2C compatible interface up to 400 kHz
- SPI compatible interface up to 15 MHz
- -40°C to +85°C operating temperature range
- many additional opcodes and features including matrix operations, string handling, A/D converter
- easy migration from uMFPU V2

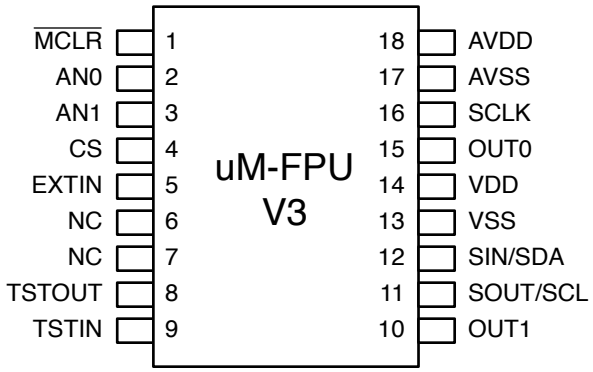
uM-FPU Core Features

- packages available: 18-lead PDIP, 18-lead SOIC, 44-lead QFN
- supports both I^2C and SPI compatible interfaces
- 256 byte instruction buffer
- 128 general purpose 32-bit registers for storing floating point or long integer values
- 8 temporary 32-bit registers to support parentheses in calculations
- Floating Point Operations
    - Set, Add, Subtract, Multiply, Divide, Power
    - Sqrt, Log, Log10, Exp, Exp10, Root
    - Sin, Cos, Tan, Asin, Acos, Atan, Atan2
    - Floor, Ceil, Round, Min, Max, Fraction, Mod
    - Negate, Abs, Inverse
    - Multiply and Accumulate, Multiply and Subtract from Accumulator
    - 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
- Matrix Operations
    - Scalar Add, Subtract, Multiply, Divide, Power
    - Element-wise add, Subtract, Multiply, Divide, Power

    - Matrix multiply
    - Count, Sum, Average, Minimum, Maximum
    - Matrix copy
- 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
    - Reverse table lookup (float and long integer)
    - N^{th} order polynomials

Pin Diagram and Pin Descriptions

PDIP-18, SOIC-18

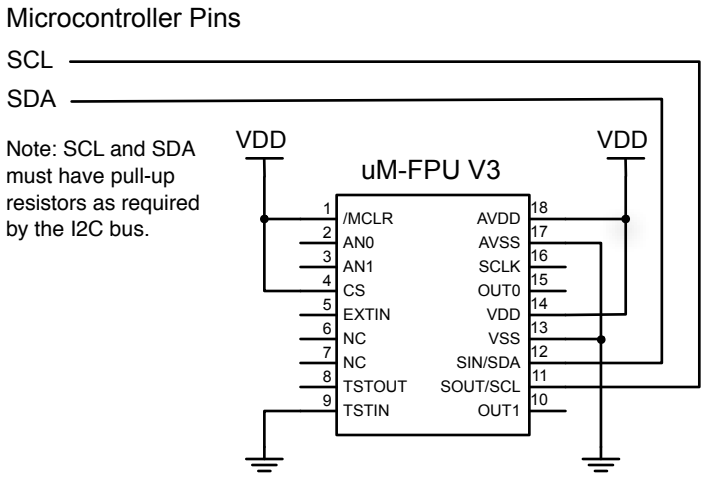


Pin	Name	Type	Description
1	/MCLR	Input	Master Clear (Reset)
2	AN0	Input	Analog Input 0
3	AN1	Input	Analog Input 1
4	CS	Input	Chip Select / Interface Select
5	EXTIN	Input	External Input
6	NC		No Connect
7	NC		No Connect
8	TSTOUT	Output	Test Output Debug Monitor - Tx
9	TSTIN	Input	Test Input Debug Monitor - Rx
10	OUT1	Output	Test Point 1
11	SOUT SCL	Output Output	SPI Output, Busy/Ready Status I^2C Clock
12	SIN SDA	Input In/Out	SPI Input I^2C Data
13	VSS	Power	Ground
14	VDD	Power	Supply Voltage
15	OUT0	Output	Test Point 0
16	SCLK	Input	SPI Clock
17	AVSS	Power	Analog Ground
18	AVDD	Power	Analog 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 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.

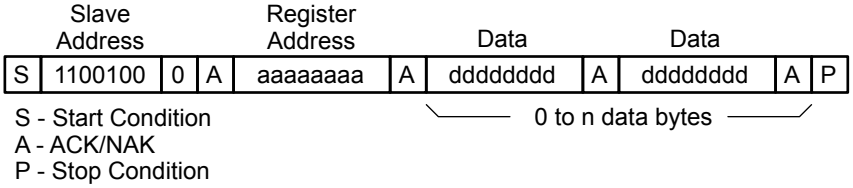
I²C Bus Speed

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

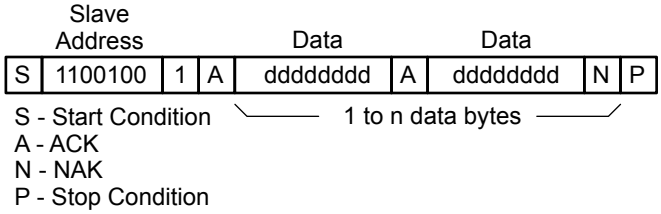
I²C Data Transfers

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



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	TBD	TBD	usec
Read Delay – debug enabled	TBD	TBD	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.

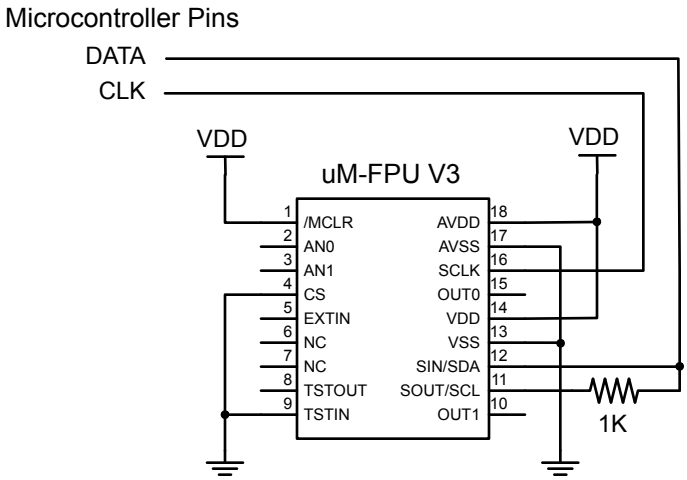
Read Delay

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).

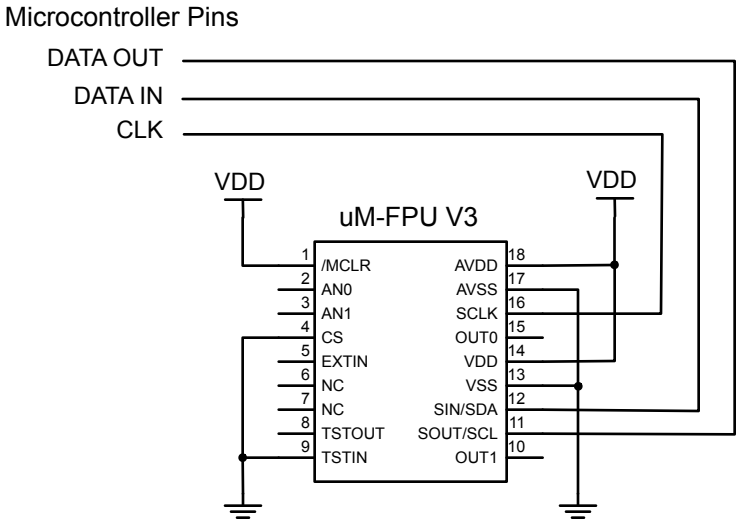
Connecting the uM-FPU using the SPI compatible interface

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



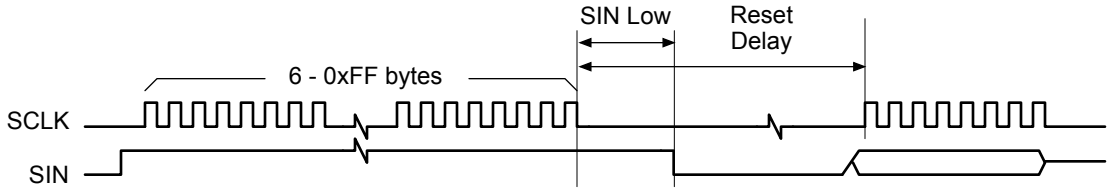
3-wire SPI Connection



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, nine consecutive 0xFF bytes must be sent. A delay of 10 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 7FFFFFFF.

Reset Timing Diagram



Item	Min	Typical	Max	Unit
Reset - 0xFF bytes	9	10		bytes
Reset - SIN Low			2	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 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

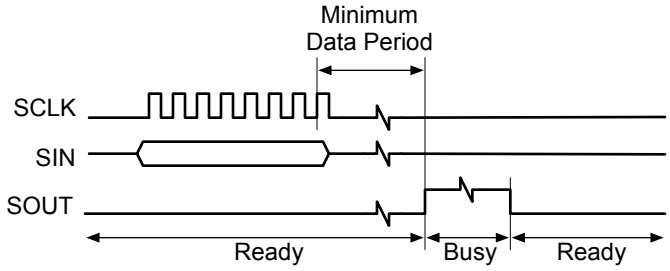
There is a minimum delay required from the end of a read instruction opcode until the first data byte is ready to be read. If debug tracing is active, this delay is longer (see table). With many microcontrollers the call overhead for the interface routines is long enough that no additional delay is required. 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 Busy/Ready Status

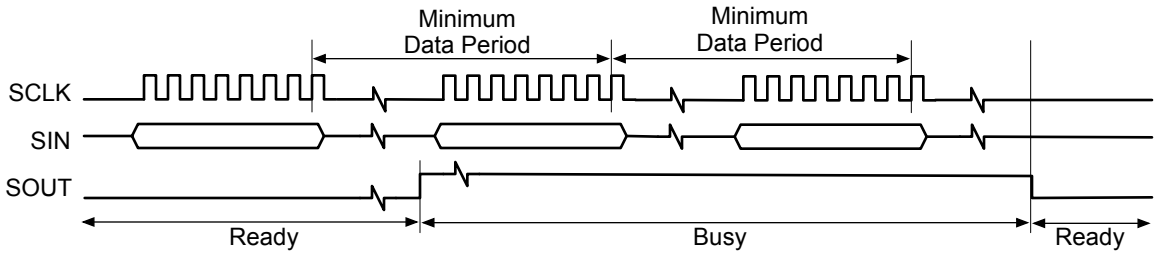
The busy/ready status must always be checked to confirm the Ready status prior to any read operation.
The Busy status is asserted as soon as a valid opcode is received. The Ready status is asserted when both the instruction buffer and trace buffer are empty. If the uM-FPU is Ready the SOUT pin is held 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.

SPI Instruction Timing Diagrams

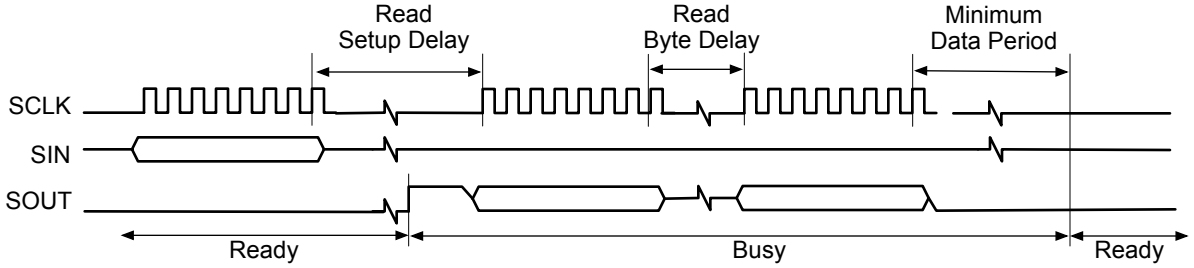
Single Byte Opcode



Multiple Byte Opcode



Opcode followed by return value



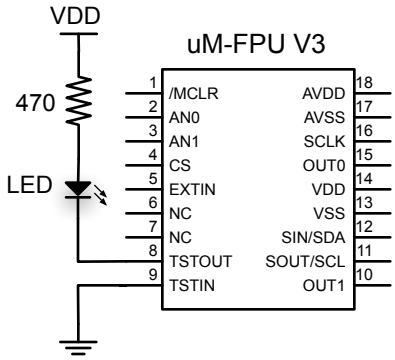
Item	Min	Max	Unit
SCLK Output Low	30		nsec
SCLK Output High	30		nsec
SCLK Frequency - single byte		15	MHz
SCLK Frequency - continuous		5	MHz
Minimum Data Period	1.6		usec
Read Setup Delay	15		usec
Read Byte Delay	1		usec
Falling Edge of CS to Rising Edge of SCLK	120		nsec
Rising Edge of CS to Bus Released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

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.

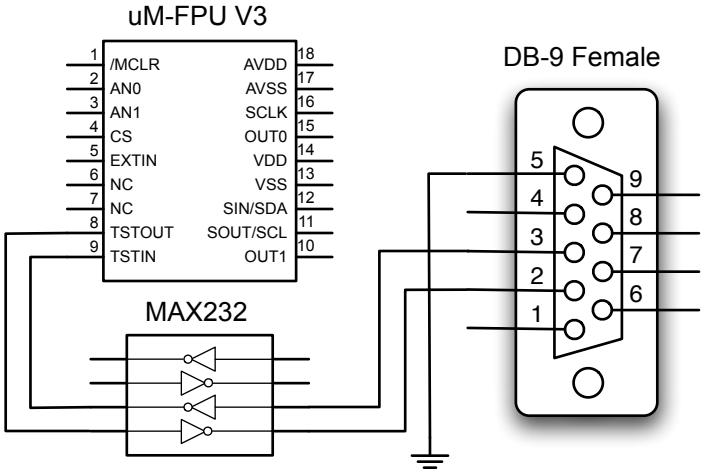
Activity Monitor

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.



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.



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
E	EEPROM	display EEPROM memory
F	Flash	display Flash stored function memory
G	Go	continue execution
MA	Matrix A	display matrix A
MB	Matrix B	display matrix B
MC	Matrix C	display matrix C
R	Register	display registers
T	Trace	toggle trace mode on/off
X	Change	displays all register that have changed
V	Version	display version information
/	Comment	add comment to debug trace
$C	Clock	select clock source
$S	Checksum	display checksum value
$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
 2003                           (i.e. FSET R3)
{BREAK}

>
 5B0028                         (i.e. LOADWORD 0x0028)
{BREAK}
>

 2400                           (i.e. FMUL R0)
{BREAK}

>
0102                            (i.e. SELECTA R2)
{BREAK}
```

EEPROM – display EEPROM memory

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

```
>E
:107C0000000000000000000000000000000000000074
:107C1000000000000000000000000000000000000064
:107C2000000000009900000000000000000000000BB
:107C3000000000000000000000000000000000000044
:107C400001020304050607080900A0B0C00000000E6
:107C500007360A33057F1680033301800000000D9
:107C6000000000000000000000000000000000000014
:107C7000000000000000000000000000000000000004
:107C8000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF04
:107C9000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4
       .
.
.
:107FD000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB1
:107FE000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA1
:107FF000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF91
```

Flash – display Flash stored function memory

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

```
>$F
:103C0000200C230725413648483F588B7B107F1041
:103C10008310874498079A0F9E64B70FBB080000CE
:103C2000000000000000000000000000000000000FF
:103C3000000000000000000000000000000000000FF
:103C4000000000000000000000000000000000000FF
:103C5000000000000000000000000000000000000FF
:103C6000000000000000000000000000000000000FF
:103C7000000000000000000000000000000000000FF
:103C800005FEF05006FEF05007FEF050073041F0CB
       .
.
.
:103F1000000000000000000000000000000000000FF
:103F2000000000000000000000000000000000000FF
:103F3000000000000000000000000000000000000FF
```

Go – continue execution

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

```
>G
```

Matrix A – display matrix A
Matrix B – display matrix B
Matrix C – display matrix C

The Matrix commands are used to display the size and current contents of Matrix A, B or C.

```
>MA
{MA 3,3
R16:40000000 R17:40800000 R18:40C00000
R19:41000000 R20:41200000 R21:41400000
R22:41600000 R23:41800000 R24:41900000}
```

Registers – display registers

The Register command displays the currently selected A register and X register and the current contents of all uM-FPU registers.

```
>R
{A=R1, X=R0
R0:40490FDB R1:3F800000 R2:0000000A R3:7FFFFFFF
R4:7FFFFFFF R5:7FFFFFFF R6:7FFFFFFF R7:7FFFFFFF
R8:7FFFFFFF R9:7FFFFFFF R10:7FFFFFFF R11:7FFFFFFF
R12:7FFFFFFF R13:7FFFFFFF R14:7FFFFFFF R15:7FFFFFFF
R16:7FFFFFFF R17:7FFFFFFF R18:7FFFFFFF R19:7FFFFFFF
R20:7FFFFFFF R21:7FFFFFFF R22:7FFFFFFF R23:7FFFFFFF
R24:7FFFFFFF R25:7FFFFFFF R26:7FFFFFFF R27:7FFFFFFF
R28:7FFFFFFF R29:7FFFFFFF R30:7FFFFFFF R31:7FFFFFFF
R32:7FFFFFFF R33:7FFFFFFF R34:7FFFFFFF R35:7FFFFFFF
R36:7FFFFFFF R37:7FFFFFFF R38:7FFFFFFF R39:7FFFFFFF
R40:7FFFFFFF R41:7FFFFFFF R42:7FFFFFFF R43:7FFFFFFF
R44:7FFFFFFF R45:7FFFFFFF R46:7FFFFFFF R47:7FFFFFFF
R48:7FFFFFFF R49:7FFFFFFF R50:7FFFFFFF R51:7FFFFFFF
R52:7FFFFFFF R53:7FFFFFFF R54:7FFFFFFF R55:7FFFFFFF
R56:7FFFFFFF R57:7FFFFFFF R58:7FFFFFFF R59:7FFFFFFF
R60:7FFFFFFF R61:7FFFFFFF R62:7FFFFFFF R63:7FFFFFFF
T1:7FFFFFFF T2:7FFFFFFF T3:7FFFFFFF T4:7FFFFFFF
T5:7FFFFFFF T6:7FFFFFFF T7:7FFFFFFF T8:7FFFFFFF}
```

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. 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 V3.0, SPI 29.48 MHz

>V
uM-FPU V3.0, I2C C8 29.48 MHz
```

Change – display changed registers

The Change command displays the currently selected A register and X register and the current contents of all uM-FPU registers that have changed since the last Change command.

```
>X
{A=R1, X=R0
R0:40490FDB R1:3F800000 R2:0000000A}

>X
{A=R1, X=R0}
```

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 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 chip will default to an internal clock 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
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 chip, excluding the stored function area. This can be used to confirm that the chip is valid.

```
>$S:27D5E8
```

Mode – set mode parameters

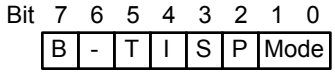
The Mode command is used to set the four interface mode parameter bytes that are stored in Flash memory. The

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 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)
01 – I^2C mode selected (CS pin ignored)
10 – SPI mode selected (CS pin is active as chip select)
11 – SPI mode selected (CS pin is active 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: 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.

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.

READVAR

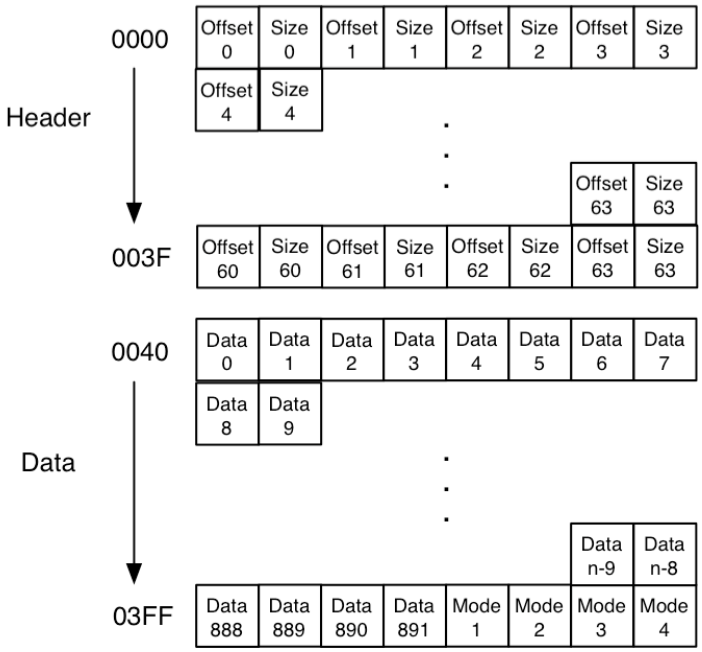
Returns the value of one of the internal registers. See the Instruction Reference for details.

Stored Functions

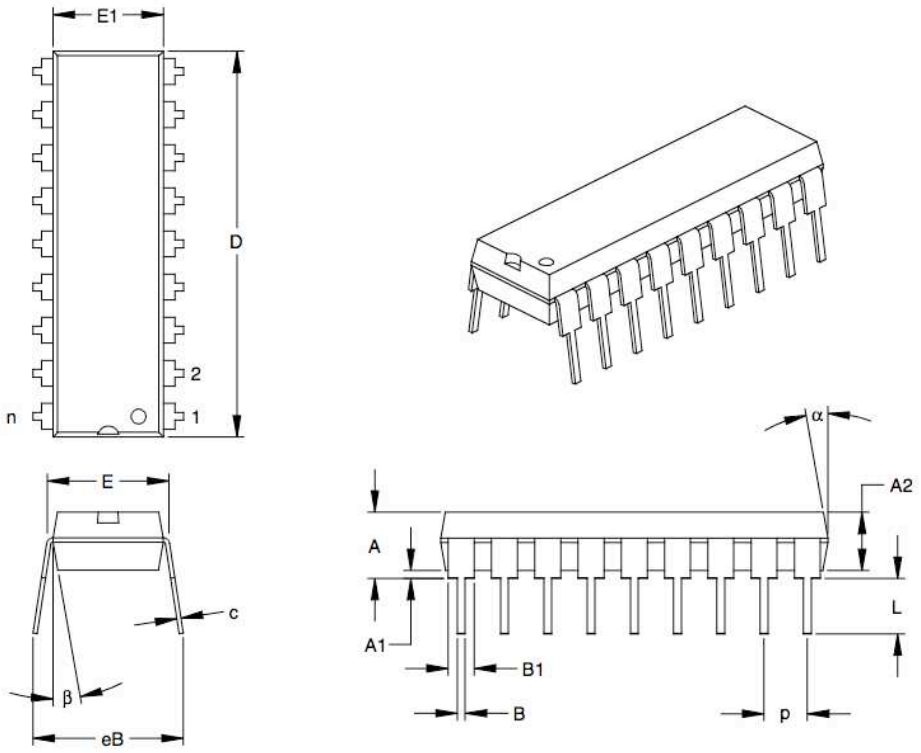
There are 1024 bytes of flash memory reserved on the uM-FPU for storing user functions and the mode parameters. Up to 64 user functions can be defined and saved by the user. Stored user functions have the advantage of conserving space on the microcontroller and greatly reducing the communications overhead 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.)



PDIP-18 Through-Hole Package

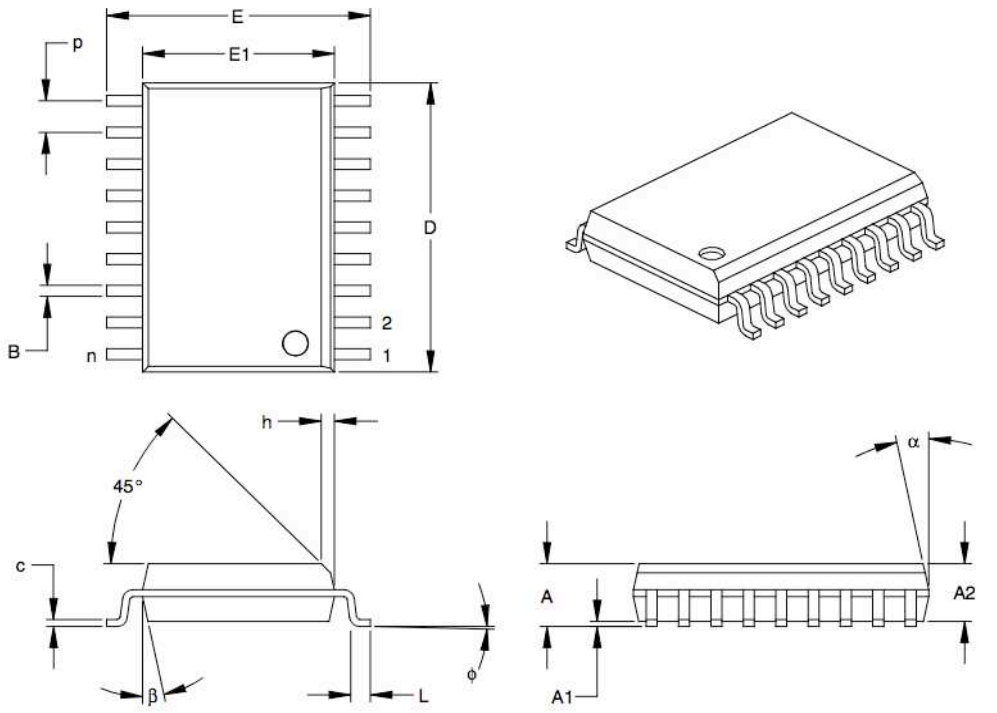


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		18			18	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.140	.155	.170	3.56	3.94	4.32
Molded Package Thickness	A2	.115	.130	.145	2.92	3.30	3.68
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.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

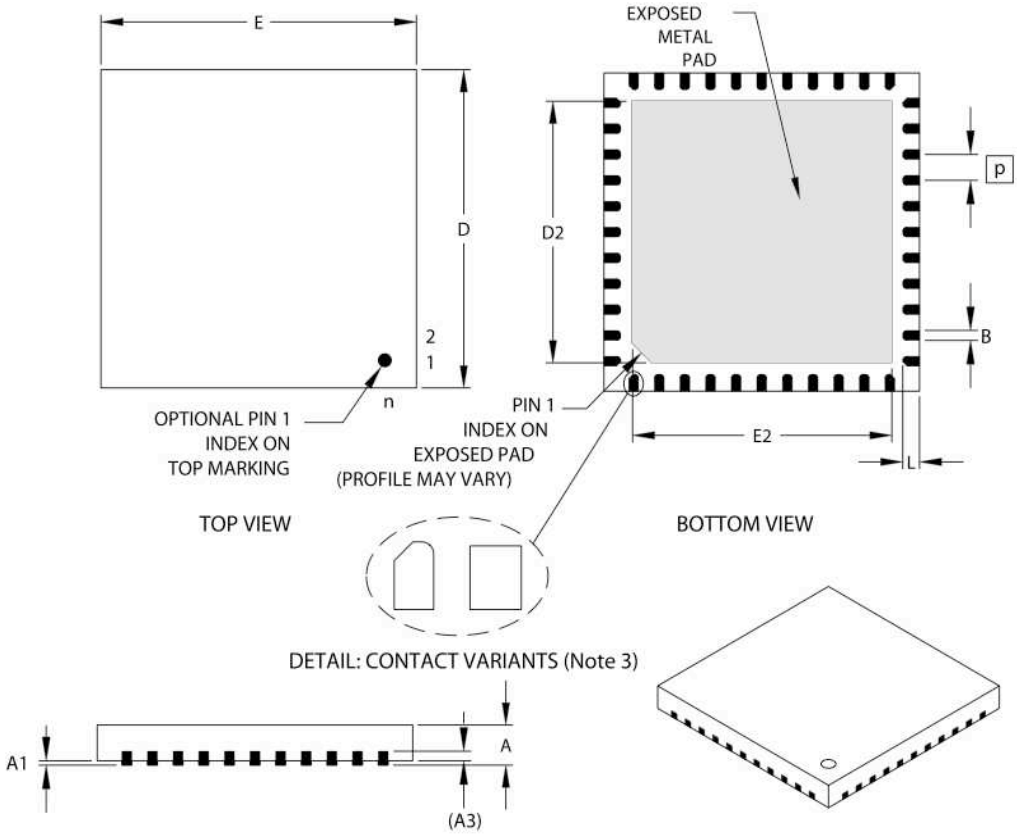


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		18			18	
Pitch	p		.050			1.27	
Overall Height	A	.093	.099	.104	2.36	2.50	2.64
Molded Package Thickness	A2	.088	.091	.094	2.24	2.31	2.39
Standoff §	A1	.004	.008	.012	0.10	0.20	0.30
Overall Width	E	.394	.407	.420	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
Recommended VDD Operating Range	4.75	-	5.25	V
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
Supply Current	-	TBD	-	mA

Further Information

Check the Micromega website at www.micromegacorp.com

Appendix A
uM-FPU V3 Instruction Summary

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[nn] = reg[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[A] and reg[nn]
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	1C		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] = l reg[A] l
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,b1,b2		Select matrix A
SELECTMB	66	nn,b1,b2		Select matrix B
SELECTMC	67	nn,b1,b2		Select matrix C
LOADMA	68	b1,b2		reg[0] = Matrix A[bb, bb]
LOADMB	69	b1,b2		reg[0] = Matrix B[bb, bb]
LOADMC	6A	b1,b2		reg[0] = Matrix C[bb, bb]
SAVEMA	6B	b1,b2		Matrix A[bb, bb] = reg[A]
SAVEMB	6C	b1,b2		Matrix B[bb, bb] = reg[A]
SAVEMC	6D	b1,b2		Matrix C[bb, bb] = reg[A]
MOP	6E	bb		Matrix/Vector operation
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	bb		Call user-defined function in Flash
EECALL	7F	bb		Call user-defined function in EEPROM
RET	80			Return from user-defined function
BRA	81	bb		Unconditional branch
BRA	82	cc, bb		Conditional branch
JMP	83	b1, b2		Unconditional jump
JMP	84	cc, b1, b2		Conditional jump
TABLE	85	tc,t0…tn		Table lookup
FTABLE	86	cc,tc,t0…tn		Floating point reverse table lookup
LTABLE	87	cc,tc,t0…tn		Long integer reverse table lookup
POLY	88	tc,t0…tn		reg[A] = nth order polynomial
GOTO	89	nn		Computed GOTO
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] = | reg[A] |
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)
LONGCON	CA	bb		reg[0] = long constant(nn)
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] = B
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	mm,nn		EEPROM[nn] = reg[mm]
EESAVEA	DB	nn		EEPROM[nn] = reg[A]
EELOAD	DC	mm,nn		reg[mm] = EEPROM[nn]
EELOADA	DD	nn		reg[A] = EEPROM[nn]
EEWRITE	DE	nn,bc,b1…bn		Store bytes in EEPROM
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 buffer
STRFIND	E7	aa…00		Find string and set selection point
STRFCHR	E8	aa…00		Set field separators
STRFIELD	E9	bb		Find field and set 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	nn		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 (0-63)
`mm`	register number (0-63)
`fn`	function number (0-63)
`bb`	8-bit value
`wwww`	16-bit value
`b1...b4`	32-bit value
`b1...bn`	string of 8-bit bytes
`ss`	Status byte
`cc`	Condition code
`ch`	A/D channel number
`bc`	Byte count
`t1...tn`	String of 32-bit table values
`aa...00`	Zero terminated ASCII string

Appendix B
uM-FPU V3 Preliminary Instruction Timing

The instruction times shown in the following table are calculated 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 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 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 TSTOUT 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	Instruction Time	
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…b4		5	
FWRITEA	17	b1…b4		5	
FWRITEX	18	b1…b4		5	
FWRITE0	19	b1…b4		5	
FREAD	1A	nn	b1…b4		(note 1)
FREADA	1B		b1…b4		(note 1)
FREADX	1C		b1…b4		(note 1)
FREAD0	1D		b1…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	nn		7	
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	wwww		10	
LOADUWORD	5C	wwww		10	
LOADE	5D			7	
LOADPI	5E			7	
LOADCON	5F	bb		5	
FLOAT	60			10-12	(note 3)
FIX	61			7-10	(note 2)
FIXR	61			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,b		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			
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	cc,b1,b2		7	(note 8)
JMP,cc	84			5	(note 8)
TABLE	85	tc,t0…tn		11	(note 8)
FTABLE	86	cc, tc,t0…tn		25	(note 8)
LTABLE	87	cc, tc,t0…tn		23	(note 8)
POLY	88	tc,t0…tn			(note 8, 9)
GOTO	83	nn		7	(note 8)
LWRITE	90	nn,b1…b4		5	
LWRITEA	91	b1…b4		5	
LWRITEX	92	b1…b4		5	

LREAD	94	nn	b1…b4		(note 1)
LREADA	95		b1…b4		(note 1)
LREADX	96		b1…b4		(note 1)
LREAD0	97		b1…b4		(note 1)
LREADBYTE	98		bb		(note 1)
LREADWORD	99		wwww		(note 1)
ATOL	9A	aa…00		5	
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	
LONGCON	CA	bb		5	
SETOUT	D0	bb		6	
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	mm,nn		5590	
EESAVEA	DB	nn		5590	
EELOAD	DC	mm,nn		5	
EELOADA	DD	nn		5	
EEWRITE	DE	nn 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	
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			4250	
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.