Micromega Corporation

uM-FPU V3.1.0 Release Notes

This document describes the new features, enhancements and bug fixes in the uM-FPU V3.1.0 chip. For a detailed description of the features, refer to the uM-FPU V3.1 Datasheet and the uM-FPU V3.1 Instruction Reference documents.

New Features

Busy/Ready Status on OUT1 pin

The `OUT1` pin can be configured to output the Busy/Ready status by setting bit 6 in mode parameter byte 0. When this feature is enabled, the `OUT1` signal is High when the uM-FPU V3 chip is Ready, and Low when the chip is Busy. When the `OUT1` pin is used for the Busy/Ready status, the `SETOUT` instruction ignores any changes to the `OUT1` pin.

SEROUT instruction

The `TSTOUT` pin has been renamed to `SEROUT` and can now be used to send serial data when the debug monitor is not enabled. The `SEROUT` instruction is used to set the mode and baud rate of the `SEROUT` and `SERIN` pins, and to send serial data to the `SEROUT` pin.

SERIN instruction

The `TSTIN` pin has been renamed to `SERIN` and can now be used to receive serial data when the debug monitor is not enabled. The `SERIN` instruction supports both character input and NMEA sentence parsing. Using the NMEA input mode, a GPS receiver can be connected directly to the uM-FPU V3.1 chip through the `SERIN` pin. The incoming data can be scanned for valid NMEA sentences, and then processed directly by the uM-FPU V3.1 chip.

WRBLK instruction

The `WRBLK` instruction has been added to support transferring a block of 32-bit values from the microcontroller to the uM-FPU V3 chip. The instruction optionally supports byte reversal so data can be transferred efficiently using the microprocessor's native storage format.

RDBLK instruction

The `RDBLK` instruction has been added to support transferring a block of 32-bit values from the uM-FPU V3 chip to the microcontroller. The instruction optionally supports byte reversal so data can be transferred efficiently using the microprocessor's native storage format.

MOP instruction (matrix operations)

The `MOP` instruction has been extended with the following operations:

31	Matrix Determinant: reg[0] = determinant of MA (2x2 and 3x3 matrices only)
32	Matrix Inverse: MA = inverse of MB (2x2 and 3x3 matrices only)
33	Indexed Load Registers to Matrix A
34	Indexed Load Registers to Matrix B

35 Indexed Load Registers to Matrix C
36 Indexed Load Matrix B to Matrix A
37 Indexed Load Matrix C to Matrix A
38 Indexed Save Matrix A to Registers
39 Indexed Save Matrix A to Matrix B
40 Indexed Save Matrix A to Matrix C

The Indexed Load Registers operations take a list of register numbers and sequentially copy the indexed register values to the matrix specified. The Indexed Load Matrix operations take a list of matrix indexes and sequentially copy the indexed matrix values to Matrix A. The Indexed Save operations take a list of register numbers or matrix indices and sequentially copy the values from matrix A to registers, matrix B, or matrix C. These operations can be used to quickly load matrices and save results, or to extract and save matrix subsets.

STRBYTE instruction

The `STRBYTE` instruction stores the lower 8 bits of register 0 to the string buffer at the current selection point.

STRINC instruction

The `STRINC` instruction increments the string selection point.

STRDEC instruction

The `STRDEC` instruction decrements the string selection point.

RET,cc instruction

The `RET,cc` instruction provides a conditional return from user-defined functions.

READVAR instruction

Two new values are returned by the `READVAR` instruction.

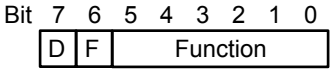
17 8-bit character at string selection point
18 number of bytes in instruction buffer

SETSTATUS instruction

This instruction is mainly intended for testing purposes. It sets the internal status byte to the 8-bit value specified.

Auto-Start Function Call

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 Debug mode
0 - use SERIN to select debug mode (High - enable debug mode, Low - disable 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

Enhancements / Changes

- VERSION instruction has been changed to return a 16-bit version number.
- STRCMP instruction has been changed to compare the current selection, rather than the whole string.
- All instructions that return a long integer result also set the long status value.
- CHECKSUM instruction now includes the Flash memory used by user defined functions.
- The activity monitor output is no longer supported by the SEROUT pin (formerly TSTOUT).
- If the OUT1 pin is enabled to output the Busy/Ready status, it can also be used as an activity monitor output.
- STRINS, FTOA, LTOA and STRBYTE instructions now update the string selection to an insertion point immediately following the current insertion. This allows multiple insertions to append to one another.
- STRSEL and STRFIELD instructions have been changed so that if bit 7 is set in the parameter byte(s) following the instruction, the value is loaded from the register specified by bits 6:0. User-defined functions can use this to make string selections using values stored in registers.
- STRFIND instruction now searches the string selection rather than the full string
- MA, MB, MC matrix commands have been removed from the debug monitor.
- the last status value and current matrix selections have been added to the R and X commands in the debug monitor.
- LEFT instruction now loads the temporary register with the previous register A value.
- LONGCON instruction has been removed.
- INDA, INDX, SAVEIND, LOADIND instruction changed to only use lower 8 bits of indirect register

Bug Fixes

- the A/D conversion speed was corrected for consistent conversions at all clock speeds.
- TRACESTR instruction fixed so that the trace string is skipped if trace is disabled.
- I^2C free space command returns correct value.

Additions to Instruction Reference

CHECKSUM **Calculate checksum for uM-FPU code** *(modified V3.1)*

Opcode: `F6`

Description: A checksum is calculated for the uM-FPU code and user-defined memory and stored in register 0. This can be used as a diagnostic test for confirming the state of a uM-FPU chip, and for confirming that a particular set of user-defined functions is loaded.

LEFT **Left Parenthesis** *(modified V3.1)*

Opcode: `14`

Description: The `LEFT` instruction saves the current register A selection, allocates the next temporary register, sets the value of the temporary register to the current register A value, then selects the temporary register as register A. The `RIGHT` instruction is used to restore previous values. When used together, these instruction are like parentheses in an equation, and can be used to allocate temporary registers, and change the order of a calculation. Parentheses can be nested up to eight levels.

Special cases: • If the maximum number is exceeded, the value of register A is set to NaN ($7FFFFFFF).

MOP **Matrix Operation** *(modified V3.1)*

Opcode: `6E bb` where: `bb` is the operation code
`6E bb ic, i1…in` `ic` is the index count
`i1…in` are the index values

Description: The operation code nn selects one of the following operations:
[new operations]

31	Matrix Determinant: reg[0] = determinant of MA (2x2 and 3x3 matrices only)
32	Matrix Inverse: MA = inverse of MB (2x2 and 3x3 matrices only)
33	Indexed Load Registers to Matrix A : `MOP,33,ic,i1...in`
34	Indexed Load Registers to Matrix B : `MOP,34,ic,i1...in`
35	Indexed Load Registers to Matrix C : `MOP,35,ic,i1...in`
36	Indexed Load Matrix B to Matrix A: `MOP,36,ic,i1...in`
37	Indexed Load Matrix C to Matrix A: `MOP,37,ic,i1...in`
38	Indexed Save Matrix A to Register: `MOP,38,ic,i1...in`
39	Indexed Save Matrix A to Matrix B: `MOP,39,ic,i1...in`
40	Indexed Save Matrix A to Matrix C: `MOP,40,ic,i1...in`

The Indexed Load Registers operations take a list of register numbers and sequentially copy the indexed register values to the matrix specified. The Indexed Load Matrix operations take a list of matrix indexes and sequentially copy the indexed matrix values to Matrix A. The Indexed Save operations take a list of register numbers or matrix indices and sequentially copy the values from matrix A to registers, matrix B, or matrix C. These operations can be used to quickly load matrices and save results, or to extract and save matrix subsets.

Special cases: • Indexed Load Register operations: register 0 is cleared to zero before the indexed values are copied, to provide an easy way to load zero values to a matrix.
• Indexed Load operations: if an index is negative, the absolute value is used as an index, and

the negative of the indexed value is copied.
• Indexed Load Matrix operations: an index of 0x80 is used to copy the negative of the value at index 0.
• Indexed Save Matrix operations: if an index value is negative, the matrix A value for that index position is not stored.

RDBLK **Read multiple 32-bit point values** *(new V3.1)*

Opcode: `71 tc` where: `tc` is the number of 32-bit values to read

Description: Return tc 32-bit values from reg[X], X = X+1
This instruction is used to read multiple 32-bit values from the uM-FPU registers. The byte immediately following the opcode is the transfer count, and bits 6:0 specify the number of 32-bit values that follow (a value of zero specifies a transfer count of 128). If bit 7 of the transfer count is set, the bytes are reversed for each 32-bit value that follows. This allows for efficient data transfers when the native storage format of the microcontroller is the reverse of the uM-FPU format. The X register specifies the register to read from, and it is incremented after each 32-bit value is read.

Special cases: • the X register will not increment past the maximum register value of 127
• if PICMODE is enabled, the 32-bit values are assumed to be floating point values

---

READVAR **Read internal variable** *(modified V3.1)*

Opcode: `FC bb` where: `bb` is index of internal register

Description: reg[0] = internal register value
Sets register 0 to the current value of one of the internal registers (based on index value passed).
[new operations]

17	8-bit character at string selection point
18	number of bytes in instruction buffer

RET **Conditional return from user-defined function** *(new V3.1)*

Opcode: `8A cc` where: `cc` is the test condition

Description: This instruction is only valid in a user-defined function in Flash memory or EEPROM memory. If the test condition is true, it causes a return from the current function, and execution will continue with the instruction following the last function call. If the test condition is false, execution continues with the next instruction.

SERIN **Serial input** *(new V3.1)*

Opcode: `CF bb` where: `bb` specifies the type of operation

Description: This instruction is used to read serial data from the SERIN pin. The instruction is ignored if Debug Mode is enabled. The baud rate for serial input is the same as the baud rate for serial output, and is set with the `SEROUT,0` instruction. The operation to be performed is specified by the byte immediately following the opcode:

0	Disable serial input
1	Enable character mode serial input
2	Get character mode serial input status
3	Get serial input character

4	Enable NMEA serial input
5	Get NMEA input status
6	Transfer NMEA sentence to string buffer

`SERIN,0`
Disable serial input. This can be used to save interrupt processing time if serial input is not used continuously.

`SERIN,1`
Enable character mode serial input. Serial input is enabled, and incoming characters are stored in a 160 byte buffer. The serial input status can be checked with the `SERIN,2` instruction and input characters can be read using the `SERIN,3` instruction.

`SERIN,2`
Get character mode serial input status. The status byte is set to zero (Z) if the input buffer is empty, or non-zero (NZ) if the input buffer is not empty.

`SERIN,3`
Get serial input character. The serial input character is returned in register 0. If this instruction is the last instruction in the instruction buffer, it will wait for the next available input character. It there are other instructions in the instruction buffer, or another instruction is sent before the `SERIN,3` instruction has completed, it will terminate and return a zero value.

`SERIN,4`
Enable NMEA serial input. Serial input is enabled, and the serial input data is scanned for NMEA sentences which are then stored in a 200 byte buffer. Additional NMEA sentences can be buffered while the current sentence is being processed. The sentence prefix character ($), trailing checksum characters (if specified), and the terminator (CR,LF) are not stored in the buffer. NMEA sentences are transferred to the string buffer for processing using the `SERIN,6` instruction, and the NMEA input status can be checked with the `SERIN,5` instruction.

`SERIN,5`
Get the NMEA input status. The status byte is set to zero (Z) if the buffer is empty, or non-zero (NZ) if at least one NMEA sentence is available in the buffer.

`SERIN,6`
Transfer NMEA sentence to string buffer. This instruction transfers the next NMEA sentence to the string buffer, and selects the first field of the string so that a `STRCMP` instruction can be used to check the sentence type. The status byte is set to greater-than (GT) if the sentence is valid, or less-than (LT) if there is an error. If an overrun error occurred, bit 4 of the status byte will also be set. If a checksum error occurred, bit 5 of the status byte will also be set. If this instruction is the last instruction in the instruction buffer, it will wait for the next available NMEA sentence. It there are other instructions in the instruction buffer, or another instruction is sent before the `SERIN,6` instruction has completed, it will terminate and return an empty sentence.

SEROUT **Serial Output** *(new V3.1)*

Opcode: `CE bb` where: `bb` specifies the type of operation
`CE bb bd` `bd` specifies the I/O mode and baud rate
`CE bb aa…00` `aa…00` is a zero-terminated string

Description: This instruction is used to set the serial input/output mode and baud rate, and to send serial data to the SEROUT pin. The operation to be performed is specified by the byte immediately following the opcode:

0	Set serial I/O mode and baud rate
1	Send text string to serial output
2	Send string buffer to serial output
3	Send string selection to serial output
4	Send lower 8 bits of register 0 to serial output
5	Send text string and zero terminator to serial output

`SEROUT,0,bb`

This instruction sets the baud rate for serial input/output, and enables or disables Debug Mode. The mode is specified by the byte immediately following the operation code:

0	57,600 baud, Debug Mode enabled
1	300 baud, Debug Mode disabled
2	600 baud, Debug Mode disabled
3	1200 baud, Debug Mode disabled
4	2400 baud, Debug Mode disabled
5	4800 baud, Debug Mode disabled
6	9600 baud, Debug Mode disabled
7	19200 baud, Debug Mode disabled
8	38400 baud, Debug Mode disabled
9	57600 baud, Debug Mode disabled
10	115200 baud, Debug Mode disabled

For mode 0, a `{DEBUG ON}` message is sent to the serial output and the baud rate is changed. For modes 1 to 10, if the debug mode is enabled, a `{DEBUG OFF}` message is sent to the serial output before the baud rate is changed.

`SEROUT,1,aa..00`

The text string specified by the instruction (not including the zero-terminator) is sent to the serial output. The instruction is ignored if Debug Mode is enabled.

`SEROUT,2`

The contents of the string buffer are sent to the serial output. The instruction is ignored if Debug Mode is enabled.

`SEROUT,3`

The current string selection is sent to the serial port. The instruction is ignored if Debug Mode is enabled.

`SEROUT,4`

The lower 8 bits of register 0 are sent to the serial port as an 8-bit character. The instruction is ignored if Debug Mode is enabled.

`SEROUT,5,aa..00`

The text string specified by the instruction (including the zero-terminator) is sent to the serial output. The instruction is ignored if Debug Mode is enabled.

SETSTATUS **Set status byte** *(new V3.1)*

Opcode: `CD bb` where: `ss` is status value

Description: The internal status byte is set to the 8-bit value specified.

STRBYTE **Insert byte at string selection** *(new V3.1)*

Opcode: `ED`

Description: The lower 8 bits of register 0 are stored as an 8-bit character in the string buffer at the current selection point. The selection point is updated to point immediately after the stored byte, so multiple bytes can be appended.

STRDEC **Decrement string selection point** *(new V3.1)*

Opcode: `EF`

Description: The string selection point is incremented and the selection length is set to zero.

Special cases: • the selection point will not decrement past the beginning of the string

---

STRINC **Increment string selection point** *(new V3.1)*

Opcode: `EE`

Description: The string selection point is incremented and the selection length is set to zero.

Special cases: • the selection point will not increment past the end of the string

---

STRFIELD **Find field in string** *(modified V3.1)*

Opcode: `E9 bb` where: `bb` is the field number

Description: The selection point is set to the specified field. Fields are numbered from 1 to n, and are separated by the characters specified by the last `STRFCHR` instruction. If no `STRFCHR` instruction has been executed, the default field separator is a comma. If bit 7 of `bb` is set, then bits 6:0 of `bb` specify a register number, and the lower 8 bits of the register specify the field number.

Special cases: • if bb = 0, selection point is set to the start of the string buffer
• if bb > number of fields, selection point is set to the end of the string buffer

---

STRFIND **Find string in the string buffer** *(modified V3.1)*

Opcode: `E7 aa...00` where: `aa...00` is a zero-terminated string

Description: Search the string selection for the first occurrence of the specified string. If the string is found, the selection point is set to the matching substring. If the string is not found, the selection point is set to the end of the string selection.

STRSEL **Set string selection point** *(modified V3.1)*

Opcode: `E4 nn mm` where: `nn` is the start of the selection
`mm` is the length of the selection

Description: Set the start of the string selection to character `nn` and the length of the selection to `mm` characters. Characters are numbered from 0 to n. If bit 7 of `nn` is set, then bits 6:0 of `nn` specify a register number, and the lower 8 bits of the register specify the start of the selection. If bit 7 of `mm` is set, then bits 6:0 of `mm` specify a register number, and the lower 8 bits of the register specify the length of the selection.

Special cases:
- if nn > string length, start of selection is set to end of string
- if nn+mm > string length, selection is adjusted for the end of string

---

WRBLK **Write multiple 32-bit values** *(new V3.1)*

Opcode: `70 tc t0..tn` where: `tc` is the number of 32-bit values to write
`t0...tn` are 32-bit values

Description: reg[X] = t, X = X+1, for t = t0 to tn

This instruction is used to write multiple 32-bit values to the uM-FPU registers. The byte immediately following the opcode is the transfer count, and bits 6:0 specify the number of 32-bit values that follow (a value of zero specifies a transfer count of 128). If bit 7 of the transfer count is set, the bytes are reversed for each 32-bit value that follows. This allows for efficient data transfers when the native storage format of the microcontroller is the reverse of the uM-FPU format. The X register specifies the register to write to, and it is incremented after each 32-bit value is written.

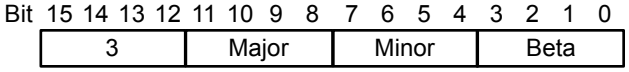
Special cases:
- the X register will not increment past the maximum register value of 127
- if PICMODE is enabled, the 32-bit values are assumed to be floating point values

---

VERSION **Copy the version string to the string buffer** *(modified V3.1)*

Opcode: `F3`

Description: The uM-FPU V3 version string is copied to the string buffer at the current selection point, and the version code is copied to register 0. The version code is represented as follows:



Bits 15-12 allows 3 for uM-FPU V3
Bits 11-8 Major Version
Bits 7-4 Minor Version
Bits 3-0 Beta Version

As an example, for the uM-FPU V3.1.0 general release:
version string: uM-FPU V3.1
version code: 0x3100