Micromega Corporation

Release Notes

uM-FPU64 Firmware Release 403

Changes for Release 403

uM-FPU64 Firmware Release 403 is a maintenance release to fix some known problems.

- fixed bug that was causing certain IDE commands to be ignored after a breakpoint.
- debug monitor commands can now be upper or lower case.

Changes for Release 402

uM-FPU64 Firmware Release 402 is a major release with many new features. The new features and changes are described below.

Hardware Breakpoints

Two general purpose hardware breakpoints, and a step breakpoint have been implemented. A new Break command has been added to the debug monitor for setting hardware breakpoints. The *uM-FPU64 IDE r405* software provides support for source level debugging using the new hardware breakpoint capability.

Serial Input Simulation

The SERIN instruction has been modified to allow the IDE to simulate serial input using the debug monitor connection. The IDE provides support for terminal emulation mode, character input, NMEA sentence input, and input from a file. See the *uM-FPU64 IDE User Manual* for more information regarding `SERIN` and `SEROUT` support.

Serial Output Data Logging and Terminal Emulation

The SEROUT instruction has been modified to allow the IDE to display or log serial output using the debug monitor connection. The IDE provides support for terminal emulation, character display, logging data to a file, and graphing serial data. The SEROUT instruction has four new actions to facilitate data logging.

`WRITE_LONG`	write register value as integer string
`WRITE_FLOAT`	write register value as floating point string
`WRITE_COMMA`	write comma
`WRITE_CRLF`	write carriage return and linefeed

See the *uM-FPU64 IDE User Manual* for more information regarding `SERIN` and `SEROUT` support.

Additional Flash Memory

The Flash memory available for storing user-defined functions has been increased to 4096 bytes.

Support for 32-bit and 64-bit Function Arguments and Returns

Changes were made to the `SELECTA`, `LEFT`, `SETARGS`, `FCALL`, `RET` and `RET,CC` instructions to provide additional support for handling 32-bit and 64-bit function arguments. The IDE has been enhanced to allow function to have a mix of 32-bit and 64-bit arguments, and to support 32-bit and 64-bit function returns.

Arrays stored in RAM

Arrays stored in RAM can be as large as the available RAM allows. The `ADDIND` instruction is generally used for array access. Since the `ADDIND` instruction uses a byte constant, there are some restrictions on maximum array size. They are as follows:

Arrays with one dimension: (e.g. `array[1000]`)
The array must fit in available available memory.

Arrays with two dimensions: (e.g. `array[500, 2]`)
The array must fit in available available memory.
The second dimension must be less than or equal to 256.

Arrays with three dimensions: (e.g. `array[10, 10, 10]`)
The array must fit in available available memory.
The second and third dimensions multiplied together must be less than or equal to 256.

Instruction Changes

`ADDIND`

- register 0 is used for pointer calculation
- the register value and constant are now multiplied by the byte size of the pointer type
- reg[0] (bits 23:0) = (reg[0] (bits23:0) + (reg[*register*] * *unsignedByte* * byte size)
- if *register* and *unsignedByte* are both zero, the pointer is decremented

`COPYIND`

- corrected problem affecting pointers with auto increment selected
- added support for Flash pointers
- fixed problem with auto-increment.

`DELAY`

- if a delay value of 0 is specified, the 32-bit delay value is loaded from register 0.

`DEVIO,COUNTER`

- added EDGE_MSEC1, EDGE_USEC1, EDGE_MSEC2, EDGE_USEC2 actions to provide the time in milliseconds or microseconds that an edge occurs

- fixed problem that occurred when debounce was set to zero

EVENT
- added `TEST` action

FCALL
- the value of registerA, or the value stored by the first `SETARGS` (if any) is stored by `FCALL`, and restored by the `RET` or `RET,CC` instruction when the function returns.

FFT
- instructions now work with matrices stored in registers or RAM

FTOA
- improved the rounding code for strings with no format and trailing nines in fraction.

LEFT
- if the temporary stack level exceeds the maximum, the temporary stack level is reset to zero.

LOADIND
- added support for Flash pointers
- fixed problem with auto-increment.

LOADMA, **LOADMB**, **LOADMC**
- if bit 7 of *row* or *column* byte is set, the value is loaded from the register specified by bits 6:0.

MOP
- instructions now work with matrices stored in registers or RAM

READVAR
- items 2, 5 and 8 return an indirect pointer to the matrix

RET, **RET,CC**
- restores the value of register A stored by the `FCALL` instruction.

RTC
- added action to convert a date and time value to a string
- added action to convert a date and time string to a value

SAVEIND
- added support for Flash pointers
- fixed problem with auto-increment.

SAVEMA, **SAVEMB**, **SAVEMC**
- if bit 7 of *row* or *column* byte is set, the value is loaded from the register specified by bits 6:0.

SELECTA
- this instruction no longer clears the temporary stack level to zero.

SELECTMA, **SELECTMB**, **SELECTMC**
- if bit 7 of the *rows* or *columns* byte is set, the value is loaded from the register specified by bits 6:0.
- indirect pointers can be used to select arrays stored in RAM

SERIN

- if debug monitor is enabled, a serial input request is sent by the debug monitor and a breakpoint occurs.
- fixed bug causing the ASYNC device to always be selected.

SEROUT

- added `WRITE_FLOAT` action: Sends floating point string to serial output.
- added `WRITE_LONG` action: Sends long integer string to serial output.
- added `WRITE_COMMA`: Sends comma to serial output.
- added `WRITE_CR`: Sends carriage return to serial output.
- if debug monitor is enabled, serial output is sent to debug output.
- if debug monitor is enabled, three additional data devices are supported.

SETARGS

- the current value of register A is saved by the first `SETARGS` instruction.
- if register A is 32-bit, registers 1-9 are used to store the arguments.
- if register A is 64-bit, registers 129-137 are used to store the arguments.
- each additional `SETARGS`, executed before the next `FCALL`, toggles between storing arguments to registers 1 to 9 (32-bit), or 129 to 137 (64-bit).

SETIND

- register 0 is set to the value of the indirect pointer
- bit 5 of the pointer data type now specifies a Flash pointer
- instruction format for Flash pointers: `SETIND type,function,offset`

Debug Monitor Changes

Changes to the debug monitor provide additional capabilities used by the *uM-FPU64 IDE* software to implementation new features.

- added semicolon as multiple command separator for debug monitor
- added `H` command to debug monitor: Go or Single-Step
- removed implicit Go command from new break instruction (`;G` used as required)
- added `WS` command to debug monitor: write string to string buffer
- added `WC` command to debug monitor: write character to register 0
- added `WT` command to debug monitor: write status
- extended `B` command to support hardware breakpoints