



**Micromega Corporation**

# Release Notes

## uM-FPU64 IDE

### Release 409

## Changes for IDE Release 409

Note: uM-FPU64 IDE Release 408 was not made available as a separate general release, all changes in IDE release 408 are part of IDE release 409.

uM-FPU64 IDE Release 409 adds several new features and fixes some known problems.

## Firmware Upgrade

To use uM-FPU64 IDE r409 software, the uM-FPU64 chip must be running firmware release 407 or higher. Firmware files are supplied with the IDE and installed in the *Firmware* folder of the IDE installation directory. The firmware can be upgraded using the *Tools>Firmware Update...* menu item. Select the appropriate firmware file as follows:

28-pin chip: *uMFPU64 64K28 Firmware V407.dat*

44-pin chip: *uMFPU64 64K44 Firmware V407.dat*

## Changes to IDE Interface

- if *PICMODE* is set by the target used in the last compile, *PICMODE* values will be displayed for *FWRITE* and *FREAD* instructions in the debug trace
- changed Flash programming to support new Flash format
- new Flash memory display window
- changed *Functions>Read Stored Functions* to *Functions>Read Flash*
- changed *Functions>Show Flash Memory...* to *Functions>Show Flash Window*
- changed *Functions>Clear Functions* to *Functions>Clear Flash*
- changed *Functions>Program Functions* to *Functions>Program Flash*
- Functions window: changed *Read Stored Functions* button to *Read Functions*
- Functions window: changed *Program Functions* button to *Program Flash*
- Functions window: added a splitter bar for resizing the *New Function* code display and the *Stored Function* code display
- Set Parameters dialog: updated *Restore Default Settings* button to ensure settings are restored to the original uM-FPU64 default settings
- added splitter between hex display and formatted display in *RAM Display* window
- added *Reset* and *Read Registers* buttons to *Interactive Compiler* window
- changed *Functions>Program Flash* to *Functions>Program Flash Memory*

- changed *Functions>Clear Flash* to *Functions>Clear Flash Memory*
- moved *Functions>Show Flash Memory* to *Window>Show Flash Memory*
- the window position for all windows is saved in preferences file
- windows snap to the left and right edges of the main window when moved to within 10 pixels of the main window edge
- changed *SEROUT* window to allow up to eight columns in Table and Graph mode

## Changes to Compiler

- *register.bit* notation can now be use for setting and testing bits in registers
- added `READ_HIGH` to `DEVIO, SPI`
- `PICMODE` added to `TARGET_OPTIONS` in target description file
- if `PICMODE` is set by the target, `FWRITE` constants will use `PICMODE` format
- `READ_HIGH` option added to `DEVIO, SPI` configuration byte
- return statement optimized if value already in register 0 or 128
- return statements with an expression will now do type conversions
- fixed code generation for certain register comparisons (e.g. `reg < -reg`)
- fixed code generation when memory array value was set to zero
- fixed check for *Nested function only allowed as first parameter* error
- fixed check for *Nested functions not allowed inside functions* error
- fixed *expected <unknown>, but found long data type* error
- fixed code generated for `#DOUBLE` and `#FLOAT64` directives
- fixed problem with code generation for certain memory pointers
- fixed code generated for `flookup( )` and `llookup( )` functions
- fixed code generated for comparing a 64-bit float with a constant
- fixed problem with 64-bit function calls in comparisons
- added `devio(LCD, INTERFACE, type)` for LCD I<sup>2</sup>C interface
- added `devio(LCD, BACKLIGHT_ON)` for LCD I<sup>2</sup>C interface
- added `devio(LCD, BACKLIGHT_OFF)` for LCD I<sup>2</sup>C interface
- optimized code generated for float register = long constant
- optimized code generated for long register = 0
- fixed the code generated for long constant expressions using `&, l, <<, >>`

# Register Bit Notation

The compiler supports bit manipulation in registers using a *register.bit* notation. Where *register* is the name of any FPU register, and *bit* is a constant from 0 to 31 for 32-bit registers, and from 0 to 63 for 64-bit register. The register.bit notation can be used to set bits, and test bits. The notation is used as follows:

Set bit to 0:

```
register.bit = 0
```

Set bit to 1:

```
register.bit = 1
```

Toggle the bit value:

```
register.bit = ~register.bit
```

Test for bit = 0:

```
if register.bit = 0 then ...  
if ~register.bit then ...  
if register.bit <> 1 then ...
```

Test for bit = 1:

```
if register.bit = 1 then ...  
if register.bit then ...  
if register.bit <> 0 then ...
```

```
fileMode      equ    L10  
READ_FILE     con    4  
APPEND        con    5  
  
fileMode.READFILE = 1      ; set bit 4 in register fileMode to 1  
  
if fileMode.READFILE then  ; execute code if bit 4 in fileMode is 1  
    read(0)  
endif
```