Micromega Corporation

Release Notes

uM-FPU64 IDE Release 405

Changes for Release 405

uM-FPU64 IDE Release 405 is a maintenance release to fix some known problems.

- fixed code generation for SELECTMA when row is a register.
- fixed display of background string selection.

Firmware Upgrade

To use uM-FPU64 IDE r405 software, the uM-FPU64 chip must be running firmware release 403 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 V403.dat*
44-pin chip:	*uMFPU64 64K44 Firmware V403.dat*

Changes for Release 404

uM-FPU64 IDE Release 404 is a major release with many new features. The new features and changes are described below.

- Source Level Debug Support
- Hardware Breakpoints
- SERIN/SEROUT Debug Support
- New RAM Display Window
- Flash Memory Size Increased to 4096 Bytes
- Support for 32-bit and 64-bit Function Arguments and Return Values
- Arrays and Pointers added to Compiler
- Additional Compiler Procedures and Functions
- Improved Code Generation

New Features

- source code and compiled code displayed in debug window
- hardware breakpoints
- single step by compiler statements or assembler instructions
- can step into functions, over functions, and out of functions
- added `SERIN` data input window with support for character mode and NMEA input
- added `SEROUT` data output window with support for:
  - text output
  - terminal emulation
  - table and graph display
- new actions added to `SERIAL` procedure:
  - `WRITE_FLOAT`
  - `WRITE_LONG`
  - `WRITE_COMMA`
  - `WRITE_CRLF`
- Flash memory size increased to 4096 bytes
- support for 32-bit and 64-bit function arguments and return values
- Register display format can be changed for multiple registers by selecting a group of registers (including Select All) in the the register and temporary register window.
- version string is read and displayed when first connected
- added firmware version check
- PICAXE target now uses target description file
- modified code generation for PICAXE to support additional registers
- changed `SELECTMA`, `SELECTMB`, `SELECTMC` to allow register arguments
- changed `LOADMA`, `LOADMB`, `LOADMC` to allow register arguments
- changed `SAVEMA`, `SAVEMB`, `SAVEMC` to allow register arguments
- added pointer support to `SELECTMA`, `SELECTMB`, `SELECTMC`
- added support for register arrays
- added support for pointers and pointer arrays
- added support for register X expressions
- added support for indirect register expressions
- added support for using register arrays to select matrices
- new compiler functions and procedures
  - `COPYIND`
  - `DIGIO`
  - `EVENT`
  - `RTC`
  - `SELECTA`
  - `SELECTX`

    - SETIND
- added register array names to debug register display
- added pointer format to debug register display

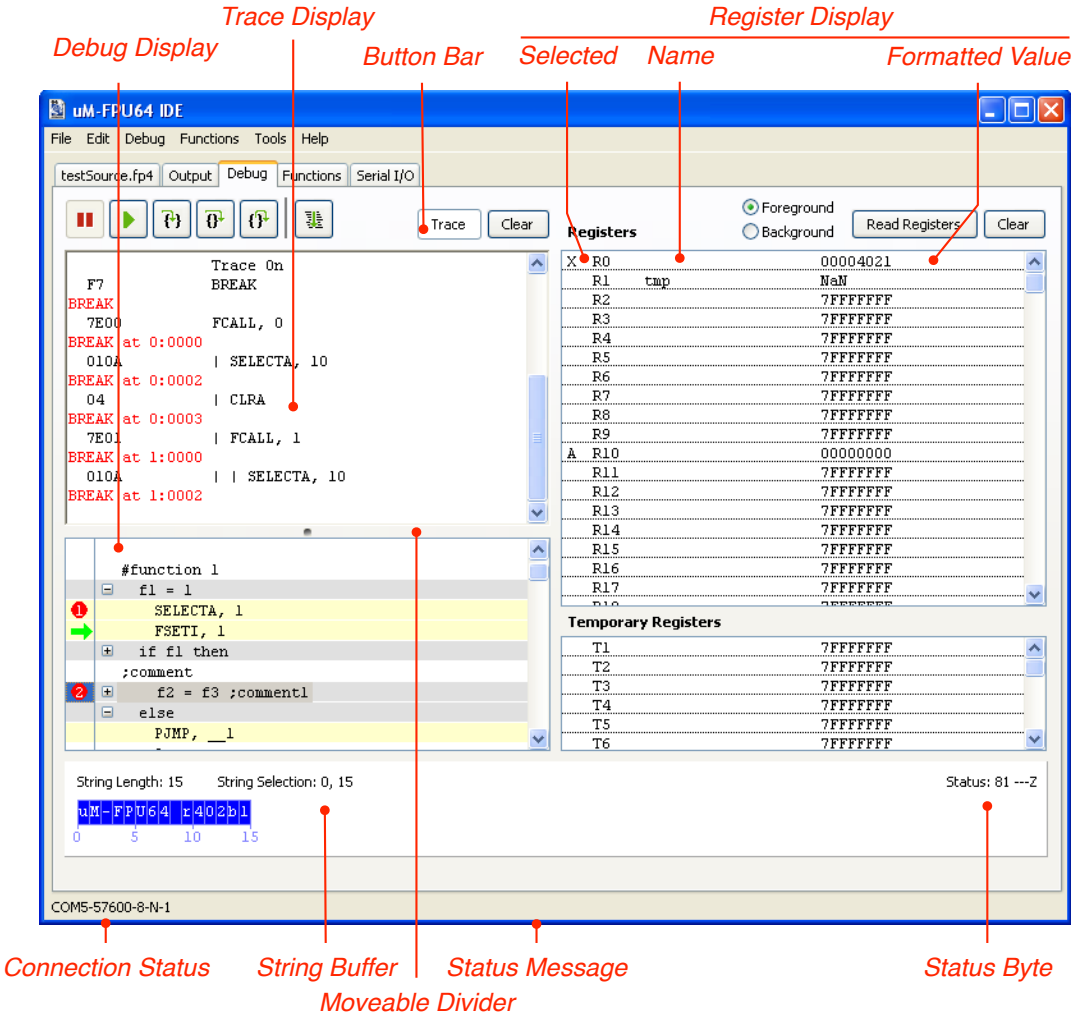
Changes

- fixed several problems with code generation for function arguments and return values if register A is a different size (32-bit/64-bit) then the argument or return values.
- debug buttons changed from text to icons
- fixed the display of unformatted floating point numbers with trailing nines in fraction
- fixed some code generation problems with conditional expressions
- optimized code generation for conditional instructions
- fixed problem with PJMP code generation
- changed DELAY to allow expression
- optimized the debug trace screen updates
- added new symbols for DEVIO,COUNTER
- fixed LTOA procedure to accept negative formats
- fixed code generation for STRSEL and STRINC
- fixed problem with #ASM instructions: FWRITE,reg and LWRITE,reg
- fixed code generation for SAVEMA, SAVEMB, SAVEMC procedures
- fixed floating point numbers to always display at least one digit to left of decimal point
- revised code generation for user defined functions to use new SETARGS/FCALL/RET capabilities provided by uM-FPU64 Release 402 firmware.
- removed LEFT/RIGHT instructions that were automatically added to functions that return values.
- added firmware folder to IDE installation.
- re-implemented firmware loader code to prevent the progress indicator from freezing during the firmware update
- fixed EXTSET, STRBYTE, TIMESET code generation
- fixed code generation for 64-bit constants
- changed the name of some of the SERIAL actions to be consistent with the SERIN and SEROUT instructions
    - WRITE_TEXT changed to WRITE_STR
    - WRITE_TEXTZ changed to WRITE_STRZ
    - WRITE_STRBUF changed to WRITE_SBUF
    - WRITE_STRSEL changed to WRITE_SSEL
    - DISABLE_INPUT changed to DISABLE
- *Show RAM Window* moved from *Tools* menu to *Window* menu
- source line is suppressed in the output when in #ASM mode

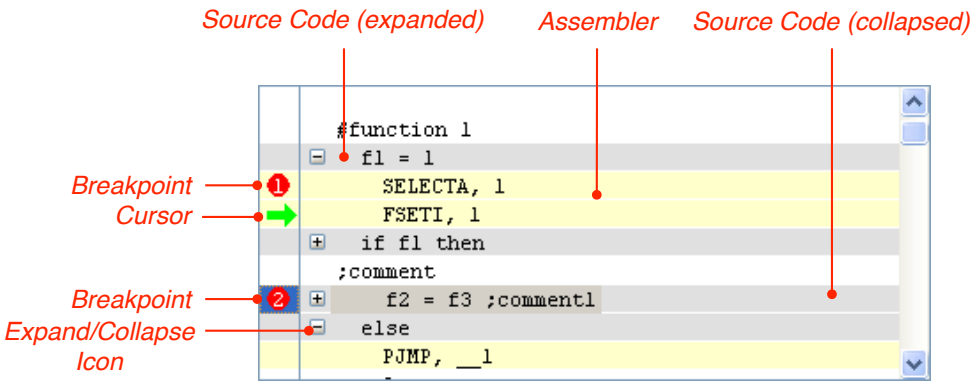
Source Level Debug Support

Source level debugging is only available for user-defined functions. The source file is displayed below the trace display. A movable divider is located between the trace display and debug display. Breakpoints can be set on any executable line shown in the debug display (both source level and assembler). All executable lines have an expand/collapse icon. Source lines can be expanded to display the assembler code generated by the source line. When the debugger is active, a cursor shows the next instruction to be executed. If a source line is expanded, the debugger will step by assembler instruction. If the source line is collapsed, the debugger will step by source line.

Debug Window



Source-level Debug Display

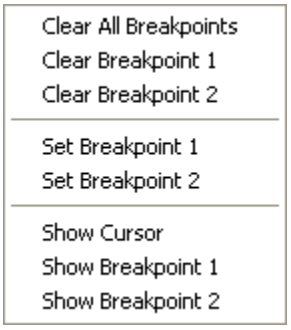


Left column: Breakpoint and cursor display.
Right column: Source code and assembler code display.

White background: Source code (non-executable).
Gray background: Source code (executable).
Yellow background: Assembler code (executable).

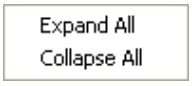
Double-click on left column (executable line):
Sets or clears breakpoint.
If no previous breakpoint, sets the next breakpoint.
If no more breakpoints, displays a placeholder.
If breakpoint or placeholder present, they are cleared.

Right-click on left column:

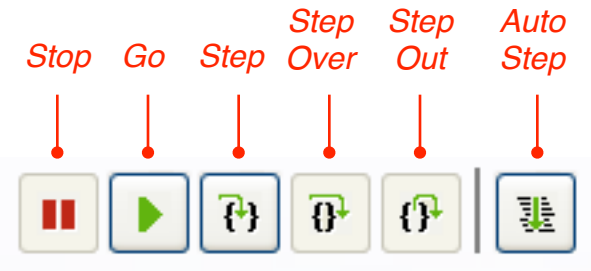


Double-click on right column (executable line):
Expands or collapses the individual line.
Breakpoints are cleared on lines that are collapsed.
Cursor is moved to expanded or collapsed line.

Right-click on right column:



Debug Buttons



Stop
- Stop execution and enter debugger.

Go
- Start or continue execution.

Step
- Step to next executable line.
- If source code is unexpanded, the step is to next executable source line.
- If source line is expanded, the step is to next assembler instruction.

Step Over
- Step to next executable line in the same function (steps over function calls).
- If source code is unexpanded, the step is to next executable source line.
- If source line is expanded, the step is to next assembler instruction.

Step Out
- Steps out of current function.

Auto Step
- Functionality is unchanged from previous version.

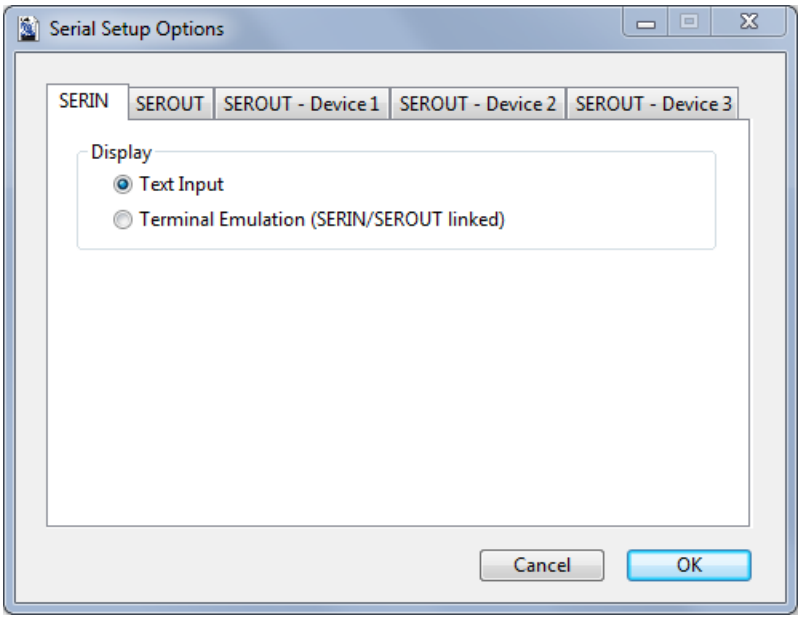
SERIN/SEROUT Support

The IDE uses the SERIN and SEROUT pins for communication with the FPU debug monitor. The uM-FPU64 V402 firmware provides the IDE with the capability to debug a project that uses the SERIN and SEROUT pins, and to receive serial data from multiple serial devices. If the debug monitor is enabled, the FPU communicates with the IDE to get data for the SERIN instruction, and sends data to the IDE from the SEROUT instruction. The SEROUT instruction supports three extra devices that can be used for sending data to the IDE. If the debug monitor is not enabled, output from the additional SEROUT devices is suppressed.

Note: To use the IDE support for the SERIN and SEROUT instructions, the debug monitor on the FPU must be active. All SEROUT,SET_BAUD instructions that disable the debug monitor must be commented out while debugging.

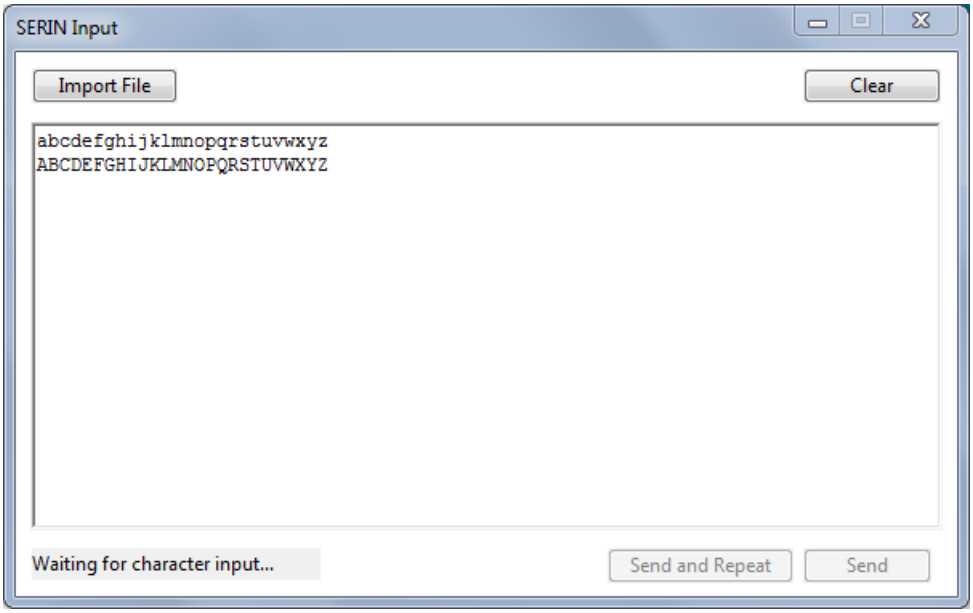
SERIN Window Setup Options

The SERIN window is configured using the *Window>Show Serial Window>Setup Options* menu item. It can be configured for *Text Input* or *Terminal Emulation* mode. In *Terminal Emulation* mode, serial input and output are both handled by the SEROUT window.



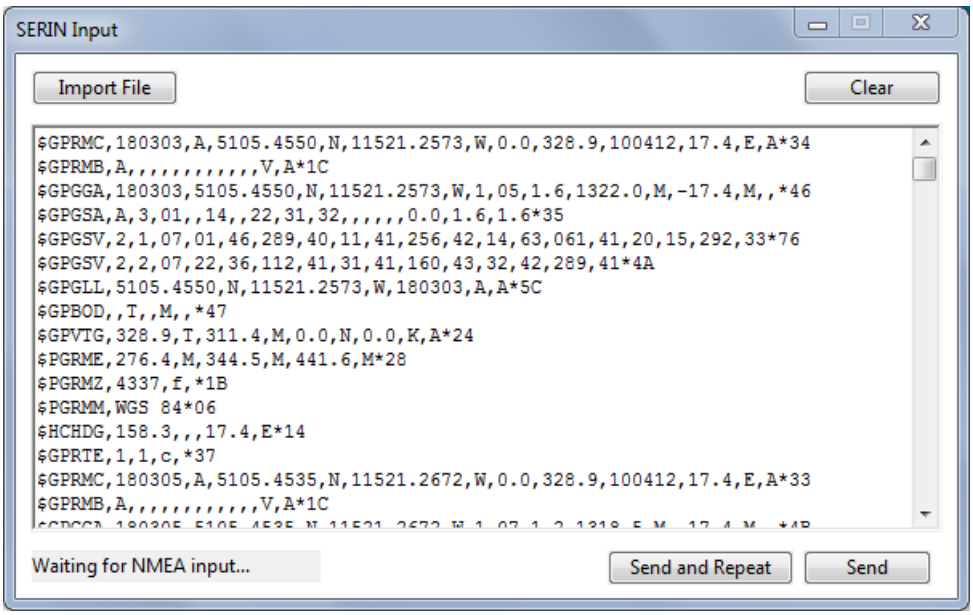
Text Input - Character Mode

When the SERIN,ENABLE_CHAR instruction is executed the IDE enters character mode. When a SERIN,READ_CHAR instruction is executed, the IDE waits for the user to send the next character. The characters to send can be entered manually in the SERIN window or imported from a text file. In *Text Input* mode, the text is not actually sent to the FPU until you select a character or group of characters, and press one of the send buttons. The *Send* button sends the single character at the start of a selection. The *Send and Repeat* button sends each of the selected characters, in sequence, one at a time, as each SERIN,READ_CHAR instruction is executed. The user is not prompted for additional input until the selection has been completely sent. The repeat action can be stopped by making another selection.



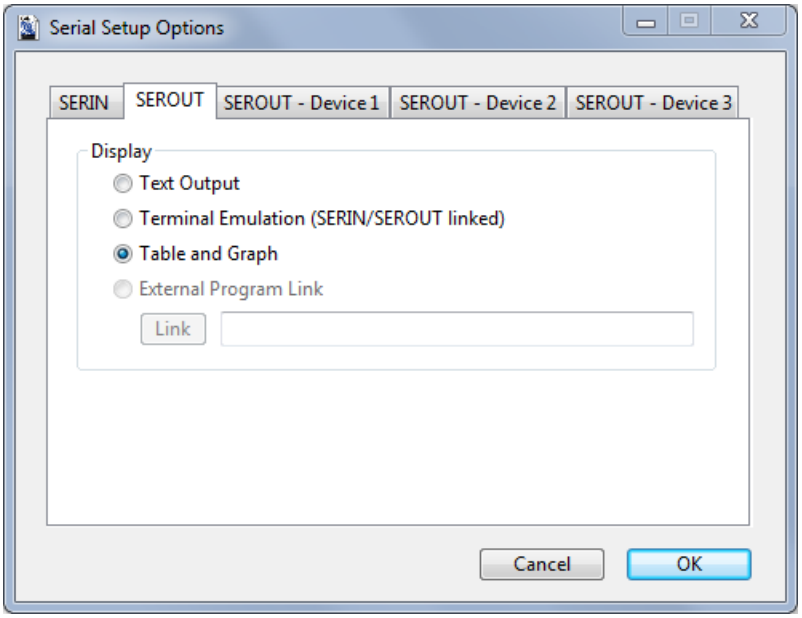
Text Input - NMEA Mode

When the `SERIN,ENABLE_NMEA` instruction is executed the IDE enters NMEA mode. When a `SERIN,READ_NMEA` instruction is executed, the IDE waits for the user to send the next NMEA sentence. The sentences to send could be entered manually in the SERIN window, but they are normally imported from a text file. The sentences are not actually sent to the FPU until you select a sentence or group of sentences, and press one of the send buttons. The *Send* button sends the single sentence at the start of a selection. The *Send and Repeat* button sends each of the selected sentences, in sequence, one at a time, as each `SERIN,READ_NMEA` instruction is executed. The user is not prompted for additional input until the selection has been completely sent. The repeat action can be stopped by making another selection. Only complete sentences are sent to the FPU. If only part of a sentence is selected, the complete sentence will be sent.



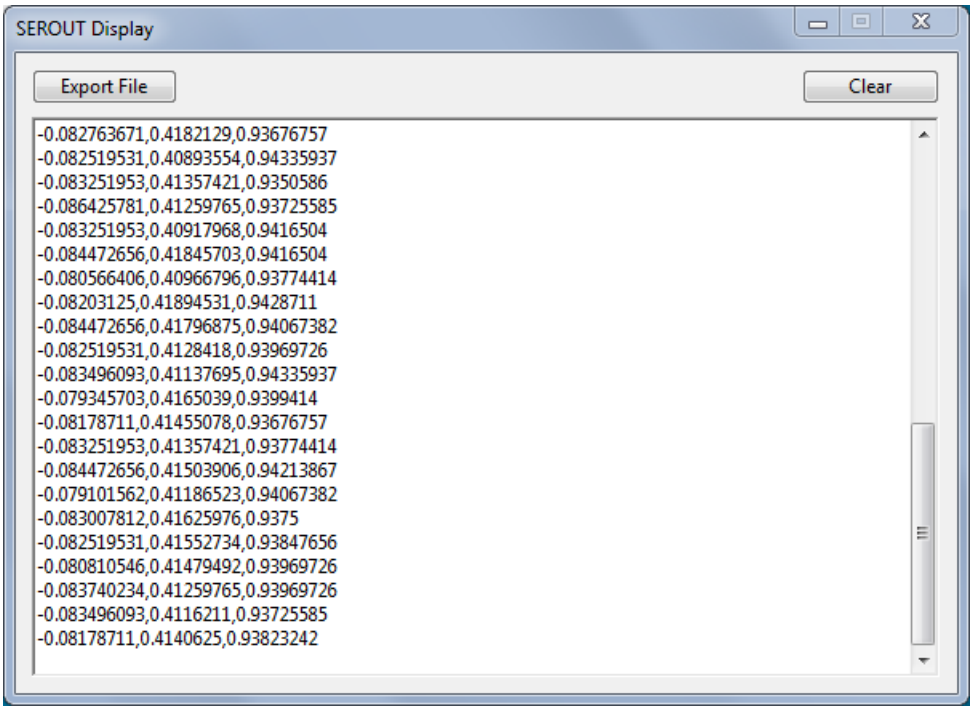
SEROUT Window Setup Options

The SEROUT window is configured using the *Window>Show Serial Window>Setup Options* menu item. It can be configured for *Text Output*, *Terminal Emulation*, or *Table and Graph* mode.



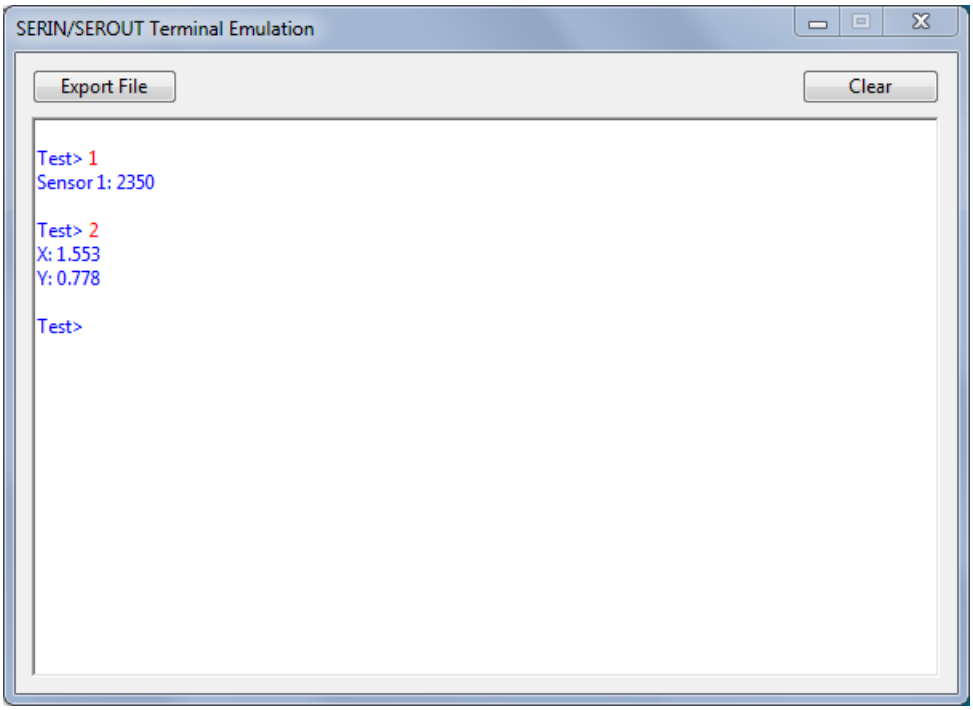
SEROUT Window - Text Output Mode

In *Text Output* mode, data sent by the `SEROUT` instruction is displayed in a text window, in black, with no additional formatting. The text output can be exported to a text file.



SEROUT Window - Terminal Emulation Mode

In *Terminal Emulation* mode, serial input and serial output are both handled by the SEROUT window. Data sent by the `SEROUT` instruction is shown in blue, with no additional formatting. Characters typed by the user are shown in red. They are not displayed until the `SERIN` instruction requests data. A typeahead buffer is provided.

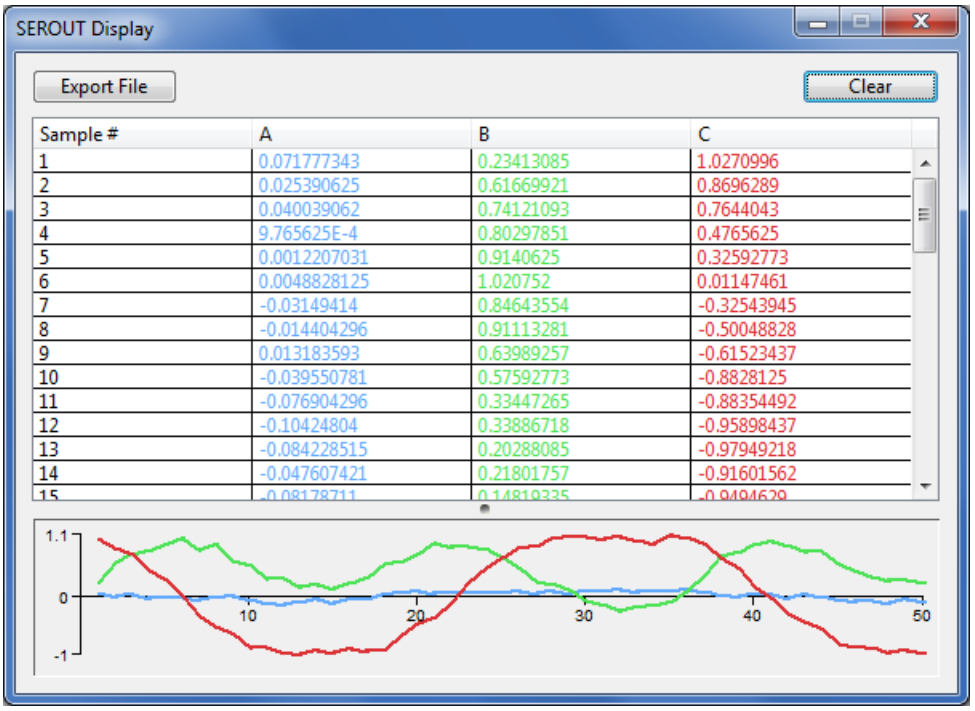


SEROUT Window - Table and Graph Mode

In *Table and Graph* mode, data sent by the `SEROUT` instruction is displayed in a table and graph. The data in each column in displayed in a different color, and each column is graphed using a line of the same color. The X and Y scales for the graph are automatically calculated to display the entire data set.

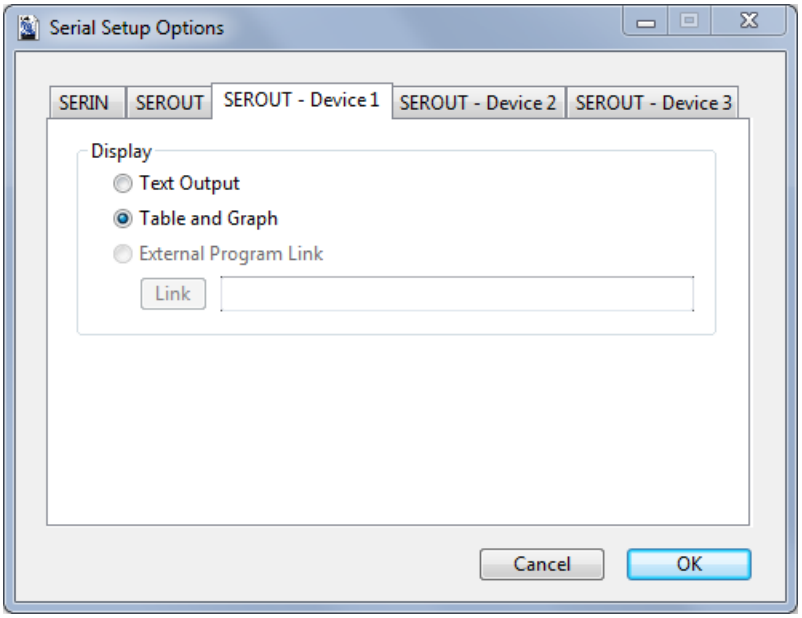
The data received from the `SEROUT` instruction must be comma separated numbers terminated with a carriage return. The new `SEROUT,WRITE_FLOAT`, `SEROUT,WRITE_LONG`, `SEROUT,WRITE_COMMA`, and `SEROUT,WRITE_CR` instructions make it easy to create comma separated values.

The values in the table can be exported to a comma separated value (CSV) file.



SEROUT Window Device 1, Device 2, Device 3 Setup Options

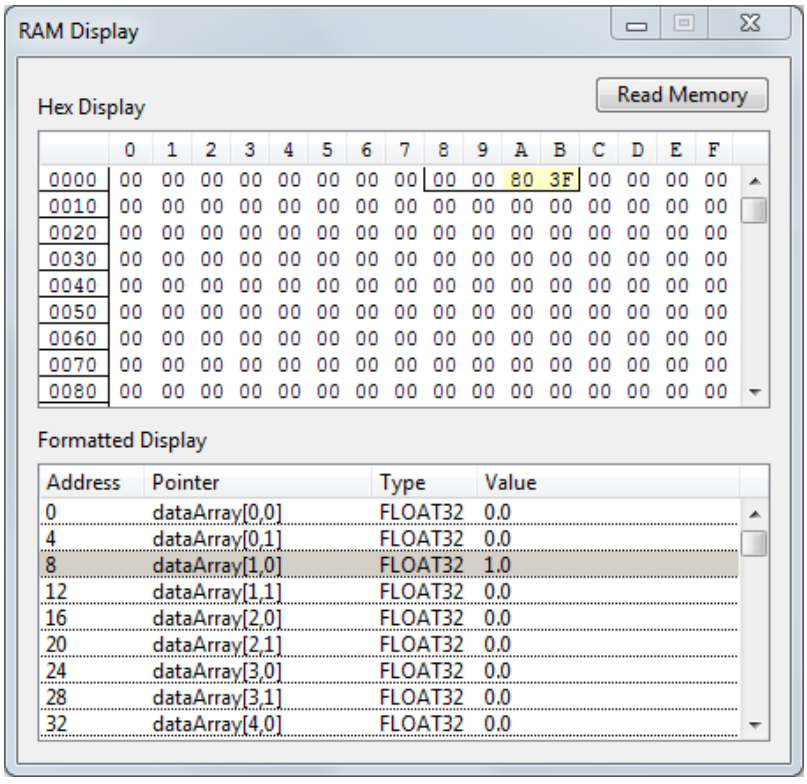
The SEROUT - Device1, SEROUT - Device 2, and SEROUT - Device 3 windows are configured using the *Window>Show Serial Window>Setup Options* menu item. They can be configured for *Text Output* or *Table and Graph* mode. The capabilities of these modes are the same as described for the SEROUT window, with the exception of *Terminal Emulator* mode, which is only available for the SEROUT window.



RAM Display Window

The RAM display window now provides additional information.

- a hex display is shown at the top
- a formatted display is shown at the bottom
- a box is drawn around the selected item in the hex display
- non-zero items are highlighted in both displays
- a pop-up menu is available in the formatted display to change the format of the displayed values
- selecting in one display, selects the corresponding item in the other display
- memory pointers identified in compiler are loaded from the Debug Registers
- registers are loaded automatically on Break, but can also be loaded manually with *Read Registers* button
- the *Read Memory* button must be pressed to read current RAM contents
- if a memory pointer is defined as an array, the array index names are shown
- data items at pointers or pointer arrays are automatically formatted according to the pointer type



Function Arguments and Return Values

User-defined functions can now have 32-bit or 64-bit arguments and return values. There are three new types defined as `FLOAT64`, `LONG64`, and `ULONG64`.

```
#FUNC % add64(FLOAT64, FLOAT64) FLOAT64

#FUNC % conv(FLOAT, LONG) FLOAT64

#FUNCTION add64(FLOAT64, FLOAT64) FLOAT64
   ; arg1 loaded into register 129
   ; arg2 loaded into register 130
   ; return value in register 128
#END

#FUNCTION conv(FLOAT, LONG64) FLOAT64
   ; arg1 loaded into register 1
   ; arg2 loaded into register 130
   ; return value in register 128
#END
```