Converting uM-FPU V3.1 code from ARMbasic V6 to ARMbasic V7

There are a number of changes required to adapt to the new SPIIN/SPIOUT and I2CIN/I2COUT functions provided in ARMbasic V7. The new FPU library code takes advantage of the parameter passing and function returns provided by V7 to provide more readable code. Another advantage of the new libraries is that the function calls are identical in both the SPI and I2C libraries. You can switch from an SPI to an I2C interface by simply changing the library that's included.

The following points summarize some of the steps required to convert uM-FPU V3.1 code from an ARMbasic V6 program to an ARMbasic V7 program.

1) Include FPUspi.bas or FPUi2c.bas library file

The *FPUspi.bas* or *FPUi2c.bas* library file must be included at the start of the program.

For an SPI connection:

```
#include "FPUspi.bas"
```

For an I^{2}C connection:

```
#include "FPUi2c.bas"
```

2) GOSUB is no longer required before a subroutine call

3) Update the Fpu_Reset call

`Fpu_Reset` is now defined as a function that returns the sync character read from the FPU. A sample reset sequence is shown below.

Original ARMbasic V6 Code

```
GOSUB Fpu_Reset                     ' reset the uM-FPU V3 chip
IF status <> SYNC_CHAR THEN         ' check for synchronization
   PRINT "uM-FPU not detected"
   END
ELSE
   GOSUB Print_Version              ' display the uM-FPU version number
   PRINT
ENDIF
```

Converted ARMbasic V7 Code

```
IF Fpu_Reset <> SYNC_CHAR THEN      ' reset FPU and check synchronization
   PRINT "uM-FPU not detected"
   END
ELSE
   Print_Version                    ' display FPU version number
   PRINT
ENDIF
```

4) Replace SPIOUT calls with Fpu_Write calls

In ARMbasic V6, the SPIOUT instruction supported a variable length output list with a variety of data types. In ARMbasic V7, the SPIOUT library routine must be called with a string array that contains the 8-bit data bytes to output. To send instructions and data to the FPU, the SPIOUT function is no longer called directly by the user's program. A series of `Fpu_Write` library functions are now called to send data to the FPU.

Example 1:

Original ARMbasic V6 Code

```
SPIOUT FpuCS, FpuOut, FpuClk, [SELECTA, F1_8, ATOF, "1.8", 0, FSET0]
```

Converted ARMbasic V7 Code

```
Fpu_Write3(SELECTA, F1_8, ATOF)
Fpu_WriteString("1.8")
Fpu_Write(FSET0)
```

Example 2:

Original ARMbasic V6 Code

```
SPIOUT FpuCS, FpuOut, FpuClk,
   [SELECTA, DegC, LOADWORD, rawTemp>>8, rawTemp, FSET0, FDIVI, 2]
```

Converted ARMbasic V7 Code

```
Fpu_Write3(SELECTA, DegC, LOADWORD)
Fpu_WriteWord(rawTemp)
Fpu_Write3(FSET0, FDIVI, 2)
```

5) Print_Float, Print_FloatFormat

These have been replaced with a single `Print_Float` routine which takes the format as a parameter. `Print_Float(0)` is the same as the previous `Print_Float` call and `Print_Float(format)` is the same as the previous `Print_FloatFormat` call.

Original ARMbasic V6 Code

```
format = 51
gosub Print_FloatFormat
```

Converted ARMbasic V7 Code

```
Print_Float(51)
```

6) Print_Long, Print_LongFormat

These have been replaced with a single `Print_Long` routine which takes the format as a parameter. `Print_Long(0)` is the same as the previous `Print_Long` call and `Print_Long(format)` is the same as the previous `Print_LongFormat` call.

Original ARMbasic V6 Code

```
gosub Print_Long
```

Converted ARMbasic V7 Code

```
Print_Long(0)
```

7) Fpu_WriteLong, Fpu_WriteFloat

In V6 these routines selected a register based on the value in the `reg` variable. The value in the `tmp` variable was then sent to the FPU. In V7, the register is no longer selected, it must be selected prior to calling the routine. The value to be sent to the FPU is passed as a parameter.

Example 1:

Original ARMbasic V6 Code

```
reg = -1
tmp = 500000
gosub Fpu_WriteLong
```

Converted ARMbasic V7 Code

```
Fpu_WriteLong(500000)
```

Example 2:

Original ARMbasic V6 Code

```
reg = Total
tmp = 500000
gosub Fpu_WriteLong
```

Converted ARMbasic V7 Code

```
Fpu_Write(SELECTA, Total)
Fpu_WriteLong(500000)
```

8) Fpu_ReadLong, Fpu_ReadFloat

In V6 these routines selected a register based on the value in the `reg` variable. The value was read from the FPU and stored in the `tmp` variable. In V7, the register is no longer selected, it must be selected prior to calling the routine. The value read from the FPU is returned from the function.

Example 1:

Original ARMbasic V6 Code

```
reg = -1
gosub Fpu_ReadLong
```

Converted ARMbasic V7 Code

```
tmp = Fpu_ReadLong
```

Example 2:

Original ARMbasic V6 Code

```
reg = Total
gosub Fpu_ReadLong
```

Converted ARMbasic V7 Code

```
Fpu_Write(SELECTA, Total)
tmp = Fpu_ReadLong
```

9) Fpu_Fcall has been removed

This function was removed in V7. It can be replaced with an `Fpu_Write`.

Original ARMbasic V6 Code

```
tmp = getLocation
gosub Fpu_Fcall
```

Converted ARMbasic V7 Code

```
Fpu_Write2(FCALL, getLocation)
```

Sample Code - original ARMbasic V6 program

```
Main:
   print
   print "Demo 1"
   print "------"

   gosub Fpu_Reset
   if status <> SYNC_CHAR then
      print "uM-FPU not detected"
      end
   else
      gosub Print_Version
      print
   endif

   gosub Init_DS1620

   SPIOUT FpuCS, FpuOut, FpuClk, [SELECTA, F1_8, ATOF, "1.8", 0, FSET0]

'-------------------- main loop -----------------------------------------------

   do
      gosub Read_DS1620
      print "Raw Temp:   $"; HEX(rawTemp)

      SPIOUT FpuCS, FpuOut, FpuClk,
         [SELECTA, DegC, LOADWORD, rawTemp>>8, rawTemp, FSET0, FDIVI, 2]

      SPIOUT FpuCS, FpuOut, FpuClk,
         [SELECTA, DegF, FSET, DegC, FMUL, F1_8, FADDI, 32]

      print "Degrees C: ";
      SPIOUT FpuCS, FpuOut, FpuClk, [SELECTA, DegC]
      format = 51
      gosub Print_FloatFormat
      print

      print "Degrees F: ";
      SPIOUT FpuCS, FpuOut, FpuClk, [SELECTA, DegF]
      format = 51
      gosub Print_FloatFormat
      print

      WAIT(2000)
      print
   loop
```

Sample Code - converted ARMbasic V7 program

```
#include "FPUspi.bas"

Main:
   print
   print "Demo 1"
   print "------"

   if Fpu_Reset <> SYNC_CHAR then
      print "uM-FPU not detected"
      end
   else
      Print_Version
      print
   endif

   Init_DS1620

   Fpu_Write3(SELECTA, F1_8, ATOF)
   Fpu_WriteString("1.8")
   Fpu_Write(FSET0)

'-------------------- main loop ----------------------------------------------

   do
      rawTemp = read_DS1620
      print "Raw Temp:   $"; HEX(rawTemp)

      Fpu_Write3(SELECTA, DegC, LOADWORD)
      Fpu_WriteWord(rawTemp)
      Fpu_Write3(FSET0, FDIVI, 2)

      Fpu_Write4(SELECTA, DegF, FSET, DegC)
      Fpu_Write4(FMUL, F1_8, FADDI, 32)

      print "Degrees C: ";
      Fpu_Write2(SELECTA, DegC)
      Print_Float(51)
      print

      print "Degrees F: ";
      Fpu_Write2(SELECTA, DegF)
      Print_Float(51)
      print

      WAIT(2000)
      print
   loop
```

uM-FPU V3 IDE Release 2 beta

The code generated by the current *uM-FPU V3 IDE V1.3* software is for ARMbasic V6. To generate code for ARMbasic V7 requires using the new version *uM-FPU V3 IDE Release 2.0 beta* software. To use this new software, do the following:

1) Unzip the *uMFPU-V3_1-ARMbasicV7* support software

2) Download the *uM-FPU V3 IDE Release 2.0 beta* software at the following link: http://www.micromegacorp.com/ide-v3-beta.html

3) Start the *uM-FPU V3 IDE Release 2.0 beta* software

4) Select the *Tools/Add Target File...* command, and select the file called *ARMbasic V7.txt* file that is included with the *uMFPU-V3_1-ARMbasicV7* support software. This copies the file into the target directory for the *uM-FPU V3 IDE*. The *ARMbasic V7* target will now be available in the *Target* menu on the input page

5) To generate code for ARMbasic V7 select *ARMbasic V7* target in the *Target* menu on the input page prior to compiling the code.