Application Note 30

Converting uM-FPU V2 code to uM-FPU V3

This application note describes the procedure for converting uM-FPU V2 code to uM-FPU V3. There are several changes between V2 and V3, but converting code is very easy. In many ways coding is simpler for the uM-FPU V3 chip since all instructions have a single byte opcodes (so the XOP instruction is no longer required), and many new instructions are designed to make coding easier and more efficient. In most cases the instruction name in V2 is the same in V3, but there are a few exceptions that will be listed below.

For a full description of the uM-FPU V3 chip, please refer to the uM-FPU V3 Datasheet, uM-FPU V3 Instruction Reference, and the reference documentation for each of the supported microcontrollers.

To show an example of converting V2 to V3, we’ll use the following BASIC Stamp code (sample.bs2). Don’t worry if you’re not familiar with the BASIC Stamp, we’ll only be concerned with the uM-FPU code inside the square brackets of the SHIFTOUT instructions. Once you’re familiar with conversion process it’s often easier to make all the changes to a line of code at once, but in this application note we’ll make the changes one step at a time so it’s easy to follow. We’ll do a direct conversion first, then optimize the code to take advantage of some of the new features in V3.
**uM-FPU V2 sample code**

This sample code takes a diameter value in centimeters, converts it to inches and calculates circumference and area.

```plaintext
Reset:
    DEBUG CR, "Conversion Example"
    DEBUG CR, "------------------", CR

GOSUB Fpu_Reset              ' reset the FPU hardware
IF status <> SyncChar THEN
    DEBUG "uM-FPU not detected."
END
ELSE
    GOSUB Print_Version        ' display the uM-FPU version number
    DEBUG CR
ENDIF

' Load constants for later use
SHIFTOUT FpuOut, FpuClk, MSBFIRST,
    [Pi, XOP, LOADPI, FSET]
SHIFTOUT FpuOut, FpuClk, MSBFIRST,
    [F2_0, LOADBYTE, 2, FSET]
SHIFTOUT FpuOut, FpuClk, MSBFIRST,
    [F2_54, ATOF, "2.54", 0, FSET]

Main:
    diameterCm = 25
    DEBUG CR, "Diameter (cm):", DEC diameterCm

    diameterIn = diameterCm / 2.54

    SHIFTOUT FpuOut, FpuClk, MSBFIRST,
        [DiameterIn, LOADBYTE, diameterCm, FSET, FDIV+F2_54]
    DEBUG CR, "Diameter (in.):"
    GOSUB Print_Float

    ' circumference = diameter * pi

    SHIFTOUT FpuOut, FpuClk, MSBFIRST,
        [Circumference, FSET+DiameterIn, FMUL+Pi]
    DEBUG CR, "Circumference (in.):"
    GOSUB Print_Float

    ' area = (diameter / 2)^2 * pi

    SHIFTOUT FpuOut, FpuClk, MSBFIRST,
        [Area, FSET+DiameterIn, FDIV+F2_0, FMUL+Area, FMUL+Pi]
    DEBUG CR, "Area (sq.in.):"
    GOSUB Print_Float

    DEBUG CR, CR, "Done.", CR
    ' end of program
END
```
Steps required to convert code to uM-FPU V3

1) Remove XOP prefix
In V2, some instructions required an XOP prefix before the opcode (e.g. XOP LOADPI). In V3, all instructions have a single byte opcode so XOP is no longer required. Remove all XOP opcodes.

   Change: [Pi, XOP, LOADPI, FSET]
   to:    [Pi, LOADPI, FSET]

2) Replace opcode+register with opcode, register
In V2, the SELECTA, SELECTB, FWRITEA, FWRITEB, FREAD, FSET, FADD, FSUB, FMUL, FDIV, LSET, LADD, LSUB, LMUL, LDIV, LWRITEA, LWRITEB, LREAD, and LUDIV instructions have the register number of the second operand stored in the lower four bits of the opcode. As a result, these instructions are generally written as opcode+register (e.g. FMUL+Pi). In V3, the second operand is not stored in the opcode, but is specified by a byte following the opcode (e.g. FMUL, Pi).

   Change: [DiameterIn, LOADBYTE, diameterCm, FSET, FDIV+F2_54]
   to:    [DiameterIn, LOADBYTE, diameterCm, FSET, FDIV, F2_54]

   Change: [Circumference, FSET+DiameterIn, FMUL+Pi]
   to:    [Circumference, FSET, DiameterIn, FMUL, Pi]

   Change: [Area, FSET, DiameterIn, FDIV+F2_0, FMUL+Area, FMUL+Pi]
   to:    [Area, FSET, DiameterIn, FDIV, F2_0, FMUL, Area, FMUL, Pi]

3) Add SELECTA opcode where register shortcut used
In V2, the SELECTA opcode is 00, so SELECTA+N is the same as N itself. As a result, the SELECTA opcode is often not specified in V2 code, just the register. In V3, the SELECTA opcode is separate from the register value so it must be specified.

   Change: [Pi, LOADPI, FSET]
   to:    [SELECTA, Pi, LOADPI, FSET]

   Change: [F2_0, LOADBYTE, 2, FSET]
   to:    [SELECTA, F2_0, LOADBYTE, 2, FSET]

   Change: [F2_54, ATOF, "2.54", 0, FSET]
   to:    [SELECTA, F2_54, ATOF, "2.54", 0, FSET]

   Change: [DiameterIn, LOADBYTE, diameterCm, FSET, FDIV, F2_54]
   to:    [SELECTA, DiameterIn, LOADBYTE, diameterCm, FSET, FDIV, F2_54]

   Change: [Circumference, FSET, DiameterIn, FMUL, Pi]
   to:    [SELECTA, Circumference, FSET, DiameterIn, FMUL, Pi]

   Change: [Area, FSET, DiameterIn, FDIV, F2_0, FMUL, Area, FMUL, Pi]
   to:    [SELECTA, Area, FSET, DiameterIn, FDIV, F2_0, FMUL, Area, FMUL, Pi]

4) Add Register 0 value or replace opcode
In V2, if the register value is 0, opcode+reg is the same as the opcode by itself. This shortcut is often used in V2
code. In V3, the register is specified as a separate byte following the opcode, but all of the basic operators also have a register 0 form of the instruction that doesn’t require the extra byte (e.g. instead of FADD, 0 use FADD0).

Change: [SELECTA, Pi, LOADPI, FSET]
to: [SELECTA, Pi, LOADPI, FSET0]

Change: [SELECTA, F2_0, LOADBYTE, 2, FSET]
to: [SELECTA, F2_0, LOADBYTE, 2, FSET0]

Change: [SELECTA, F2_54, ATOF, "2.54", 0, FSET]
to: [SELECTA, F2_54, ATOF, "2.54", 0, FSET0]

Change: [SELECTA, DiameterIn, LOADBYTE, diameterCm, FSET, FDIV, F2_54]
to: [SELECTA, DiameterIn, LOADBYTE, diameterCm, FSET0, FDIV, F2_54]

5) Use immediate mode instructions
In V3, all of the basic operators also have an immediate mode of the instruction that can be used for small integer values (-128 to 127). This is very common (e.g. \( x = x + 5 / 10 \)) and can make the code simpler and more efficient.

Remove: SHIFTOUT FpuOut, FpuClk, MSBFIRST,
[F2_54, ATOF, "2.54", 0, FSET]
(constant 2.54 no longer required)

Change: [SELECTA, Area, FSET, DiameterIn, FDIV, F2_0, FMUL, Area, FMUL, Pi]
to: [SELECTA, Area, FSET, DiameterIn, FDIVI, 2, FMUL, Area, FMUL, Pi]

Change: [SELECTA, DiameterIn, LOADBYTE, diameterCm, FSET, FDIV, F2_54]
to: [SELECTA, DiameterIn, FSETI, diameterCm, FCNV, 5]

6) Use FCNV instruction
Many common conversions are provided by the FCNV instruction in V3. In this example centimeters are being converted to inches which is FCNV, 5 (see uM-FPU V3 Instruction Reference for a full list of conversions).

Remove: SHIFTOUT FpuOut, FpuClk, MSBFIRST,
[F2_54, ATOF, "2.54", 0, FSET]
(constant 2.54 no longer required)

Change: [SELECTA, DiameterIn, FSETI, diameterCm, FCNV, 5]
uM-FPU V3 sample code

The following code shows the results of the conversion to uM-FPU V3 code.

```plaintext
Reset:
   DEBUG CR, "Conversion Example"
   DEBUG CR, "------------------", CR

GOSUB Fpu_Reset                      ' reset the FPU hardware
IF status <> SyncChar THEN
   DEBUG "uM-FPU not detected."
   END
ELSE
   GOSUB Print_Version                 ' display the uM-FPU version number
   DEBUG CR
ENDIF

' Load constant for later use
SHIFTOUT FpuOut, FpuClk, MSBFIRST, [SELECTA, Pi, LOADPI, FSET0]

'==============================================================================
'-------------------- main routine --------------------------------------------
'==============================================================================

Main:
diameterCm = 25
   DEBUG CR, "Diameter (cm):       ", DEC diameterCm

   '------------------------------
   ' convert inches to centimeters
   '------------------------------
   SHIFTOUT FpuOut, FpuClk, MSBFIRST,
   [SELECTA, DiameterIn, FSETI, diameterCm, FCNV, 5]
   DEBUG CR, "Diameter (in.):      ", DEC DiameterIn
   GOSUB Print_Float

   '------------------------------
   ' circumference = diameter * pi
   '------------------------------
   SHIFTOUT FpuOut, FpuClk, MSBFIRST,
   [SELECTA, Circumference, FSET, DiameterIn, FMUL, Pi]
   DEBUG CR, "Circumference (in.): ", DEC Circumference
   GOSUB Print_Float

   '------------------------------
   ' area = (diameter / 2)^2 * pi
   '------------------------------
   SHIFTOUT FpuOut, FpuClk, MSBFIRST,
   [SELECTA, Area, FSET, DiameterIn, FDIVI, 2, FMUL, Area, FMUL, Pi]
   DEBUG CR, "Area (sq.in.):       ", DEC Area
   GOSUB Print_Float

   DEBUG CR, CR, "Done.", CR            ' end of program
END
```
Summary of Conversion Steps (by Task)

This summary includes some conversion steps not shown in the previous example.

- Remove XOP prefix (e.g. XOP, ATAN2 to ATAN2)
- Replace opcode+register with opcode, register (e.g. FADD+N to FADD,N)
- Add SELECTA opcode where register shortcut used (e.g. XVAL to SELECTA,XVAL)
- Add Register 0 value or replace opcode (e.g. FSET to FSET0)
- Use immediate mode instructions (e.g. LOADBYTE,5, FDIV to FDIVI, 5)
- LOADONE has been removed. Use immediate opcodes (e.g. FSUB, 1)
- Use new features in uM-FPU V3

Summary of Conversion Steps (by Opcode)

<table>
<thead>
<tr>
<th>uM-FPU V2</th>
<th>uM-FPU V3</th>
<th>Action Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECTB</td>
<td>removed</td>
<td>change instruction to specify the operand</td>
</tr>
<tr>
<td>FWRITEB</td>
<td>removed</td>
<td>replace with FWRITE, or FWRITE0</td>
</tr>
<tr>
<td>FLOAT</td>
<td>stores result in register A</td>
<td>add FSET0 after FLOAT</td>
</tr>
<tr>
<td>FIX</td>
<td>stores result in register A</td>
<td>add LSET0 after FLOAT</td>
</tr>
<tr>
<td>XOP</td>
<td>not required</td>
<td>remove</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>renamed</td>
<td>replace with FCALL, nested calls now allowed</td>
</tr>
<tr>
<td>IF_FSTATUSA</td>
<td>changed</td>
<td>change conditional assembly to use BRA, JMP, GOTO instructions</td>
</tr>
<tr>
<td>IF_FSTATUSB</td>
<td>(as above)</td>
<td>(as above)</td>
</tr>
<tr>
<td>IF_FCOMPARE</td>
<td>(as above)</td>
<td>(as above)</td>
</tr>
<tr>
<td>IF_LSTATUSA</td>
<td>(as above)</td>
<td>(as above)</td>
</tr>
<tr>
<td>IF_LSTATUSB</td>
<td>(as above)</td>
<td>(as above)</td>
</tr>
<tr>
<td>IF_LCOMPARE</td>
<td>(as above)</td>
<td>(as above)</td>
</tr>
<tr>
<td>IF_LUCOMPARE</td>
<td>(as above)</td>
<td>(as above)</td>
</tr>
<tr>
<td>IF_LTST</td>
<td>(as above)</td>
<td>(as above)</td>
</tr>
<tr>
<td>READBYTE</td>
<td>renamed</td>
<td>replace with LREADBYTE</td>
</tr>
<tr>
<td>READLONG</td>
<td>renamed</td>
<td>replace with LREADWORD</td>
</tr>
<tr>
<td>LINCA</td>
<td>changed</td>
<td>replace with LINC</td>
</tr>
<tr>
<td>LINC</td>
<td>changed</td>
<td>replace with LINC</td>
</tr>
<tr>
<td>LDECA</td>
<td>changed</td>
<td>replace with LDEC</td>
</tr>
<tr>
<td>LDECL</td>
<td>changed</td>
<td>replace with LDEC</td>
</tr>
<tr>
<td>LWRITEB</td>
<td>removed</td>
<td>replace with LWRITE or LWRITE0</td>
</tr>
<tr>
<td>FRACTION</td>
<td>renamed</td>
<td>replace with FRAC</td>
</tr>
<tr>
<td>LOADZERO</td>
<td>removed</td>
<td>replace with CLR, CLRA, or CLR0</td>
</tr>
<tr>
<td>LOADONE</td>
<td>removed</td>
<td>replace with immediate value instruction (e.g. FSETI,1) or LOADBYTE,1</td>
</tr>
</tbody>
</table>
New Features in uM-FPU V3 to Consider Using

- 128 32-bit general registers
- 256 32-bit EEPROM register slots
- Register X opcodes for rapidly reading, writing and accessing sequential registers with auto-increment.
- The FSUBR and FDIVR instructions for reverse operations.
- The FMOD instruction for floating point mod
- The FIXR instruction for round and fix (FIX is truncate and fix).
- The MOP instructions for matrix and vector operations.
- The LOADCON and LONGCON instructions for loading common constants
- The FCNV instruction for converting units
- The FMAC and FMSC for multiply and accumulate
- Two Analog to Digital channels
- Elapsed time counter
- External event counter
- String handling instructions
- user-defined functions stored in Flash or EEPROM
- expanded user-defined function space and function size
- enhanced conditional execution in user-defined functions
- forward and reverse table lookup in user-defined functions

Further Information

Check the Micromega website at www.micromegacorp.com for up-to-date information.