1. Interrupt Handlers in High-Level Code

Interrupt handlers must be written in machine code. However, you can write a machine code "wrapper" that will call a high-level IsoMax word to service an interrupt. This application note describes how. You may find it useful to refer to the previous sections *Machine Code Programming* and *Using CPU Interrupts in the IsoPod*.

2. How it Works

The machine code routine below works by saving all the registers used by IsoMax, and then calling the ATO4 routine to run a high-level IsoMax word. The high-level word returns to the machine code, which restores registers and returns from the interrupt.

```
HEX 0041 CONSTANT WP

CODE-SUB INT-SERVICE
DE0B P,           \  LEA  (SP)+
D00B P,           \  MOVE X0,X:(SP)+
D10B P,           \  MOVE Y0,X:(SP)+
D30B P,           \  MOVE Y1,X:(SP)+
D08B P,           \  MOVE A0,X:(SP)+
D60B P,           \  MOVE A1,X:(SP)+
D28B P,           \  MOVE A2,X:(SP)+
D18B P,           \  MOVE B0,X:(SP)+
D70B P,           \  MOVE B1,X:(SP)+
D38B P,           \  MOVE B2,X:(SP)+
D80B P,           \  MOVE R0,X:(SP)+
D90B P,           \  MOVE R1,X:(SP)+
DA0B P,           \  MOVE R2,X:(SP)+
DB0B P,           \  MOVE R3,X:(SP)+
DD0B P,           \  MOVE N,X:(SP)+
DE8B P,           \  MOVE LC,X:(SP)+
DF8B P,           \  MOVE LA,X:(SP)+
F854 P, OBJREF P, \  MOVE X:OBJREF,R0
FA54 P, WP P,     \  MOVE X:WP,R2
D80B P,           \  MOVE R0,X:(SP)+
DA1F P,           \  MOVE R2,X:(SP)    ; Note no increment on last push!
87D0 P, xxxx P,   \  MOVE #$XXXX,R0    ; This is the CFA of the word to execute
E9C8 P, ATO4 P,   \  JSR  ATO4         ; do that Forth word
FA1B P,           \  MOVE X:(SP)-,R2   ; restore the saved wp
F81B P,           \  MOVE X:(SP)-,R0   ; restore the saved objref
FF9B P,           \  MOVE X:(SP)-,LA
DA54 P, WP P,     \  MOVE R2,X:FWP
D854 P, OBJREF P, \  MOVE R0,X:OBJREF
FE9B P,           \  MOVE X:(SP)-,LC
FD1B P,           \  MOVE X:(SP)-,N
FB1B P,           \  MOVE X:(SP)-,R3
FA1B P,           \  MOVE X:(SP)-,R2
F91B P,           \  MOVE X:(SP)-,R1
F81B P,           \  MOVE X:(SP)-,R0
F39B P,           \  MOVE X:(SP)-,B2
F71B P,           \  MOVE X:(SP)-,B1
F19B P,           \  MOVE X:(SP)-,B0
F29B P,           \  MOVE X:(SP)-,A2
```

```
F61B P,           \  MOVE X:(SP)-,A1
F09B P,           \  MOVE X:(SP)-,A0
F31B P,           \  MOVE X:(SP)-,Y1
F11B P,           \  MOVE X:(SP)-,Y0
F01B P,           \  MOVE X:(SP)-,X0
EDD9 P,           \  RTI
END-CODE
```

The only registers that are saved automatically by the processor are PC and SR. *All* other registers that will be used must be saved manually. To allow a high-level routine to execute, we must save R0-R3, X0, Y0, Y1, A, B, N, LC, and LA. Two registers that need not be saved are M01 and OMR, because these registers are never used or changed by IsoMax. We must also save the two variables `WP` and `OBJREF`, which are used by the IsoMax interpreter and object processor.

Since the DSP56F805 processor does not have a "pre-increment" address mode, the first push must be *preceded* by a stack pointer increment, LEA (SP)+, and the last push must *not* increment SP.

The instruction ordering may seem peculiar; this is because a MOVE to an address reigster (Rn) has a one-instruction delay. So we always interleave another unrelated instruction after a MOVE x, Rn. Note also the use of the symbols `ATO4` and `OBJREF` to obtain addresses. The variable `WP` is located at hex address 0041 in current IsoMax kernels, and this is defined as a constant for readability.

The value shown as "xxxx" in the listing above is where you must put the Code Field Address (CFA) of the desired high-level word. You can obtain this address with the phrase

```
' word-name CFA
```