DSP-93 Programming Hints

Frank H. Perkins, Jr. WBSIPM (fperkins@onramp.net)

Introduction

The DSP-93 is surprising user-friendly to program, considering it has a 40 MHz Harvard-architecture DSP processor under the hood. Applications can be successfully developed with only a PC and an oscilloscope. So far, more than a half dozen radio amateurs have developed and published applications for the DSP-93, including NSEG's windows-based oscilloscope and spectrum analyzer displays, W3HCF's super-hot HF modem, a collection of satellite and terrestrial modems by this author, plus Mac versions of the spectrum analyzer and oscilloscope displays by W5RKN.

In this paper, I offer you several hints on programming the DSP-93 that will hopefully get you around a couple of rough spots I have encountered. These hints are intended for someone with a working knowledge of assembly language programming, the 32OC25 instruction set and the Programming Guide for the DSP-93'.

DMOV, MACD, LTD, etc.

DMOVs and instructions with embedded DMOVs only work in infernal data memory. This is often a big surprise to programmers that are familiar with the 32OC26 DSK kit, which only has internal data memory. The DSP-93 can have up to 64K of data memory. However, DMOV, MACD, LTD, etc. will not work properly in the external RAM segment of this memory. Be sure you keep your data delay lines for FIR filters, correlators, scramblers, etc. in the internal data memory segment!

Data to Program Memory Remapping

To support adaptive filters, the 32OC25 can remap data memory at 0200h to 02FFh to program memory at OFFOOh to OFFFFh. Avoid the temptation to put any tables or code in the OFFOOh to OFFFFh segment of program memory, as it will disappear if the CNFP instruction is invoked.

UART Data Transfers

Remember that UART read/writes, which are done with 16 bit IN/OUT instructions to data memory, are only valid in the lower eight bits. Be sure to mask the upper eight bits to zero.

AND, ANDK, OR, ORK, XOR, XORK

ANDK, ORK, and XORK support mask shifting into the high word of the accumulator; AND, OR and XOR do not. AND zeros the high word of the accumulator. ANDK zeros all bits above and below the shifted mask and always zeros the MSB of the high word, regardless of the

shift. OR and XOR do not affect the high word of the accumulator. ORK and XORK do not affect bits above or below the shifted mask and do not affect the MSB of the accumulator, regardless of the shift. The somewhat subtle differences in high word treatment between memory addressable and immediate versions of these logical instructions can be confusing, especially if sign extension is set.

Nonlinear Operations

Nonlinear operations can be bandlimited or not bandlimited. Due to alias residues, nonlinear operations that are not bandlimited can create some real surprises. For example, in my first attempt at an APT demodulator 1 used the ABS operation to do a full-wave rectification of the 2400 Hz amplitude-modulated carrier signal, followed by a low-pass FIR filter. Bad idea. I had a IO-15% “hum” signal in the demodulated output.

After hours of circuit troubleshooting looking for the source of the hum, I started to realize I had created it in the DSP math. I did a simulation of the APT demodulator in QuickBasic and the hum was there! The problem was that taking the absolute value of a sampled analog signal was not a band-limited operation, and had created many, many harmonics. One of harmonics landed just above or below the sampling frequency and was aliased right into the passband of the low-pass filter. When I squared the signal to detect it, the problem went away. Squaring a signal is a band-limited nonlinear process, and only creates a second harmonic. If your sampling rate is at least four times the highest frequency in the incoming signal, alias residues will not be a problem for square-law detection or product detection, which are both band-limited nonlinear processes that create only second harmonic components.

Clipping, half-wave and full-wave rectification are examples of nonlinear operations that are not bandlimited. These operations can be used under certain circumstances, but be sure to do an alias-residue study before using them.

Analog and Digital SW Probes

My basic code debugging tools are analog and digital software probes. I try to dedicate pointer AR7 for the analog signal probe:

```
; Initialize the probe pointer in your initialization routine as follows.

PNTR-INI  LRLK      AR7,07Eh    ; point AR7 to 07Eh (unused location)

; Then place the following code just below an operation you want to check,
; and reassemble. This code will load the target variable, shifted as needed for
; scaling, into the probe buffer.

PROBE     LARP      AR7         ; make AR7 pointer
          LAC       *, W        ; get variable, shift to scale
          SACL                  ; store @ probe
```

```
; Then use this code near the end of the DSP routine to output the probe
; buffer to the D/A converter, where it can be viewed with an oscilloscope.

AI0_OUT   LDPK    00h       ; data mem page 0
          LARP    AR7       ; make AR7 pointer
          LAC     *         ; get probe
          ANDK    OFFFCh    ; mask out AI0 control bits
          SACL    DXR       ; AI0 out
```

For a digital probe, I move this code just below the logic operation I want to check:

```
; load the buffer containing the target bit and mask it off

TST-LGC   LAC     XXX       ; get buffer containing bit to output
          ANDK    YYY       ; mask off bit

          BZ      BIT-LO    ; if bit = 0 goto BIT-LO

BIT-HI    LAC     DO        ; else load DO
          ORK     04000h    ; OR RDI bit to 1
          SACL    DO        ; store DO

          B       LGCTOUT   ; goto LGC-OUT

BIT-LO    LAC     DO        ; else load DO
          ANDK    OBFFFh    ; AND RDI bit to 0
          SACL    DO        ; store DO

LGC-OUT   OUT     DO,O6h    ; output to TNC port RDI bit
```

In this case, the probed data bit appears on the RDI output line of the DSP-93 modem disconnect header. On my unit, this line is very easy to reach with an oscilloscope probe. I often compare two logic signals by putting the second signal on the RCLKI output line.

AI0 Low-Pass Filter Programming

The sampling rate Fs of the 32044 AI0 chip in the DSP-93 is given as:

Fs = 5000 / (Ta * Tb), in kHz

However, the low-pass input and output filters in the AI0 are programmed by the value of Ta only. The cut-off frequency Fc for these filters is approximately:

Fc = 60 / Ta, in kHz

Take the case of an 8.681 kHz sampling frequency. Ta * Tb = 576. Assuming we want a 2.5 kHz cut-off frequency:

$$Ta = 60/2.5 = 24, and Tb = 24$$

Had we chosen Ta = 12 and Tb = 48 to achieve the same sampling rate, Fc would now be 5 kHz, and input signal aliasing and poor output signal reconstruction could be problems. In picking Ta and Tb values, expect to do some juggling between the sampling rate and the cut-off frequency of the low-pass filters. Remember, for linear signal processing you want Fc less than 50% of Fs and for band-limited nonlinear processing (second-harmonic generation only) you want Fc less than 25% of Fs.

References

1. Parsons, D Haselwood, B. Stricklin, *Programming Guide for the DSP-93,* TAPR, http:lfwww. tapr.orgltaprIhtmllldsp93. html, 1995.

2. TMS320C2x User's *Guide,* Texas Instruments, 1993.