Using the Motorola DSP56002 EVM for Audio Processing in a DSP Laboratory

Richard E. Pfile
Purdue School of Engineering and Technology at IUPUI

Abstract

The EET department at Indiana University-Purdue University at Indianapolis developed a Real Time Digital Signal Processing course with a practical focus on the implementation of DSP algorithms on a real time processor. A low cost 24 bit fixed point real time processor is used for the laboratories, The laboratories focus on audio effects which includes such projects as a guitar tuner, FIR and IIR filters, and tone generators.

Motorola recently introduced the DSP56002 EVM; an evaluation module for the DSP56002 chip. The EVM was designed to provide a low cost test platform for customers interested in evaluating the 24 bit 56002 digital signal processor. The EVM'S low cost and high performance make it an ideal teaching platform for educational institutions interested in developing a real time digital signal processing course with a laboratory.

The EVM hardware includes A/D and D/A converters capable of converting up to 48K samples per second. Two channels each of input and output are provided which can be used for stereo audio experiments.

The EVM comes packaged with an assembler and full screen debugger which interfaces to a personal computer through the serial port. The debugger displays all registers and selected memory locations which allows students to step through programs and easily observe the operation of the DSP chip.

The EVM costs $120.00. For an extra $100. a keyboard, good quality amplified speaker, and cassette player were purchased to go with the system. This equipment along with an oscilloscope and function generator was all that was necessary to equip laboratory stations to perform a variety of DSP experiments. Several laboratories using audio effects proved to be extremely popular with students.

Introduction

The EET department at Indiana University-Purdue University at Indianapolis has developed a Digital Signal Processing course with an applied focus on the implementation of DSP algorithms using a real time DSP processor. Because specialized DSP processors have an instruction set optimized for implementation of DSP algorithms, the programs are quite short and programming can be taught in very little time. If an assembly language programming capability is a prerequisite for the course, I have found that it is possible to teach both DSP theory and programming a DSP processor in one four credit hour course. In addition, the development system hardware is inexpensive and easy to use.

Laboratory Hardware

A fixed point processor was selected for the course. Fixed point processors are used in a growing body of applications because they are low cost and have high clock speeds. A 24 bit fixed point processor has a dynamic range of 144 dB (20 $\log_{10}(2^{24})$) which is adequate for all but the most demanding applications.

We chose a Motorola DSP56002, an upgrade of the DSP56000, for the course. This powerful 24 bit fixed point processor is available in versions with clock speeds up to 80 MHz and can perform many parallel instructions. The processor also has built-in peripheral functions such as a synchronous serial interface, serial communications interface, multiple interrupts, and parallel I/O ports which make it flexible and easy to use in applications.

Motorola recently introduced a low cost evaluation module for the 56002 processor, called the 56002 EVM. It retails for $149.00. but is available for $120. with the educational discounts (contact Motorola University Support). The EVM comes complete with an assembler and full screen debugger. It has 32K of memory and can be powered by a 9 volt plug-in wall transformer or dc supply. The EVM has stereo analog inputs and outputs with 1/8 inch stereo plugs making the EVM compatible with a variety of common audio devices such as cassette and CD players, keyboards, earphones and powered speakers. Students enjoy trying out DSP algorithms using CDs or cassettes they bring in from home.

The EVM uses a 16-bit multimedia audio codec for the A/D and D/A functions. The codec is programmable and can be setup for sample rates from 4 KHz to 48 KHz. It also has programmable input and output amplifiers which allow the unit to be compatible with a variety of equipment. With a programmed input gain of one the system has an input range up to 4 volts peak-to-peak; compatible with battery powered cassette and CD players. If needed, up to 22.5 dB of gain can be provided in 1.5 dB steps. By changing the input gain, inputs as small as 20 millivolts p-p or as large as 4 volts p-p can drive the A/D converter to full scale. At the outputs up to 94.4 dB of attenuation is available in 1.5 dB steps. By changing the output attenuation a full scale D/A output swing can be programmed to range from 3 volts to microvolt levels. The programmable output attenuation is particularly handy when trying to match the output levels to headphones of various sensitivities. Students can set the attenuation to match their preferred listing levels.

Development System Software

The 56000 cross assembler supplied with the EVM is high quality and provides all of the functionality of the commercial assembler except the capability of linking object modules. Because the 56000 processor is optimized for DSP algorithms, the programs tend to be rather short and the inability to link modules in the classroom environment is not a hindrance. Otherwise the assembler is complete with macros and a full slate of assembler directives including special directives to generate sine and cosine tables.

The debugging software for the evaluation module has pull-down menus and is easy to use. The debugging screen displays processor registers, disassembled program code, and memory. Programs can be stepped through or run at full speed. Breakpoints can be inserted and memory locations displayed and modified using symbolic names or actual addresses. Data can be displayed in hexadecimal, decimal, binary, or fractional notation.

Introductory Laboratories

Students spend the first laboratory becoming familiar with the EVM software. Students load an executable file, examine and modify memory and registers, step through the code, set program breakpoints, and exercise other capabilities of the debugging software. By seeing the 56002 registers and memory on the screen at one time, students quickly become familiar with the architecture of the chip.

The second laboratory focuses on the fractional number representation used in DSP processors. Fractional numbers are left justified (in general purpose processors, numbers are right justified) and students examine the numbers after move, multiplication and addition instructions have been executed. They also examine the effects of rounding and the treatment of numbers as they are moved to registers of various lengths.[1]

Finally, students write the code to sample a stereo signal using the audio codec on the EVM board and then send it right back out the codec D/A converter. Interfacing to the codec chip is somewhat complex, but Motorola supplies a program to access it. Using an include statement the codec software can be assembled with the student's program and they can be isolated from the many details involved in accessing the codec. The codec is accessed using the Motorola Synchronous Serial Interface (SS1). Depending on the goals and time available in a course, details of the SS1 interface can be covered in class or skipped entirely without creating problems.

Using the Motorola software, the left and right channels of A/D data can be accessed simply by reading memory locations RX_BUFF_BASE and RX_BUFF_BASE+l. Data placed in memory locations TX_BUFF_BASE and TX_BUFF_BASE+1 is sent to the D/A converter. Students also modify the input gain and output attenuation and change the sample rates by selecting various values to make up the TONE_OUTPUT and TONE_INPUT constants. The following code illustrates reading the A/D converter and writing the result out the D/A converter. A stereo cassette player can provide a suitable input signal and stereo headphones can be connected to the output.

```
;*******************************************************
; Data IN and OUT routine
; program skeleton to which later routines are added
;**** ***** ********************************************

; Codec assess program supplied by Motorola with the EVM
include         'codec.asm'
; Program "mist.asm" is documented in this paper
include         'mist.asm'

TONE_OUTPUT EQU HEADPHONE_EN+LINOUT_EN+(4*LEFT_ATTN)+(4*RIGHT_ATTN)
TONE_INPUT  EQU MIC_IN_SELECT+(15*MONITOR_ATTN)
```

```
                org             p:              ;program memory
user_prog

;*******************************************
; Insertion point (A) for later code additions
;*******************************************

                ;Loopto wait for the synchronous serial
                ; interface to transmit (and receive) 4 words.

loop_1
                ;Waitaslongasbit  lset
                jset            #2,x: ssisr,*
                ; Wait as long as bit 1 clear
                jclr            #2,x: ssisr,*

                ; Code to get receive data andload transmit data

                ; Read A/D left channel into A register
                move            x: RX_BUFF_BASE,a
                ; Read A/D right channel into B register
                move            x: RX_BUFF_BASE+ 1,b

                ; Codec setup data for transmit
                move            #ToNE_ouTPuT,yo
                move                    y0,x:TX_BUFF_BASE+2
                ; Codec setup data for receive
                move            #TONE_INPUT,y0
                move                    yo,x:Tx_BuFF_BAsE+3

;*******************************************
; Insertion point (B) for later code additions
;****   ****************************************

                ;WritedatatoD/A   converter
                ; Left channel, data right back out from a
                move            a,x:TX_BUFF_BASE
                ; Right channel, data right back out from b
                move            b,x:TX_BUFF_BASE+l

                ; Endless loop
                jmp             loop_1
                end

                        ; End of program
;*******************************************
```

Sine Wave Generators

For the forth laboratory students generate a sine wave. This simple exercise allows students to become familiar with circular queues; one the main features of DSP processors. On the 56002 processor, a circular queue of any length can be specified by loading the size of the queue minus 1 in a pointer modifier register and setting the pointer register to the beginning of the queue. When the pointer is incremented to the end of the queue it will automatically wrap around back to the beginning of the queue. The queue must be placed in memory such that there are zeros in the k LSBS where 2^k is greater than or equal to the size of the buffer. For example, if a circular buffer is 96 stages, the lower address boundary of the data must have it's seven LSBS equal to zero. The modifier register is loaded with a value of 95.[2]

The code shown below for queue access is very compact and powerful. With a sample rate of 48 KHz the code will loop through the 96 word sine buffer 500 times a second generating a 500 Hz tone. The code down to "endm" generates a sine table in X memory.

```
;****     **********************************************
; Insert this code in the DATA IN and OUT routine
;****     **********************************************

                ; **** Insert this memory declaration after the
                ; include statements
                org             x:
                ; Dsm reserves memory on a boundary suitable for
                ; a circular buffer
cir_buf         dsm             96
                org             x:cir_buf
points  equu            96
pi              equ             3.141592654
freq            equ             2. 0*pi/@cvf(points)

                ; Assembler directive loops to endm generating a
                ; sine table
count   set             o
                ; Repeat all statements to "endm" 96 (points)
                dup             points
                ; Reserve on word of memory
                dc              @sin(@cvf(count) *freq)
count   set             count+ 1
                endm

                ; ****Place this code at insertion point A
                ; Point r0 to circular buffer

                move            #cir_buf,r0
                ; Load modifier register
                move            #95,m0
                ; ****Place this code at insertion point B
                ; Access circular buffer and increment
                ; a pointer with every interrupt
                move        x:(r0)+,a
                ; Move data in right channel
                move        a,b
;****   ******************************************
```

A warning stating a floating point value is outside of the fractional domain will be issued. This warning can be ignored, since the assembler will substitute the maximum possible fractional value for a one.

For the next two laboratories, a timer interrupt is programmed to interrupt at fixed intervals and step through a sine table. These programs are a bit longer and I did not include them in the paper, but I will e-mail them to anyone who is interested.

In the first laboratory an interrupt is set up to step through the circular queue at a set rates to make different frequency tones corresponding to the notes of guitar strings. A timer/counter with an interrupt is used to step through the sine table at the appropriate rate. Using the timer with interrupts is easy to teach and is an important concept for DSP processors. We usually have several students in the class who play guitar and are particularly motivated by this laboratory.

For the next laboratory we measure the harmonics of a complex tone on a small electronic keyboard and duplicate the sound using the DSP. Sine waves at five different frequencies and amplitudes make up the tone. This requires several pointers to a sine table which are incremented by different indexes. Again the 56002 can implement this in just a few instructions using the circular registers with the associated index registers.

Filters

For the next laboratory we implement a finite-duration impulse response (FIR) filter. We use Matlab with the Signal Processing Toolbox to calculate coefficients for the FIR digital filter. Matlab has a module (fir1) which will generate the coefficients for a lowpass filter and put them in a row vector.[3] The number of coefficients and cutoff frequency of the filter must be specified. Matlab will also plot the frequency response of filters various order filters. The filter has a 4000 hertz cutoff frequency but only has eleven coefficients and does not have a sharp cutoff.

After the coefficients are calculated students implement a lowpass filter on the 56002 EVM. Again because the 56002 has specialized DSP instructions the program is very short. The code for a FIR filter is shown below:

```
;**************************************************
;InsertinDATA  IN and OUT routine
;**************************************************

                    ;**** Insert after include statements

                    ; Reserve a circular buffer for input signal
                    org            x:
ntaps  equu                11
states  equ                ntaps

                    ; Reserve coefficients for 11 tap filter
                    org            y:o
coef
                    dc             219839
                    dc             227621
                    dc             309733
                    dc             380389
                    dc             428243
                    dc             445220
                    dc             428243
                    dc             380389
                    dc             309733
                    dc             227621
                    dc             219839

                    ; ****Place this code at insertion point A
                    ; RO points to the A/D data
                    ; R4 points to the coefficients
                    ; Register initialization
                    ; Initialize data pointer
                    move           #states,r0
                    ; Initialize circular buffer size for r0
                    move           #ntaps- 1 ,m0
                    ; Initialize coefficient pointer
                    move           #coef,r4
                    ; Initialize circular buffer size for r4
                    move           #ntaps-1,m4

                    ; FIR filter
                    ; ****Place this code at insertion point B
                    ; Get A/D value
                    move           a,x0
                    ; Move A/D data into buffer
                    ; Move 1st coefficient into y0
                    Clr            a              x0,x:(r0)+      y:(r4)+,y0
                    ; Repeat multiply and accumulate, ntaps -1 times
                    rep         #ntaps-1
                    mac         x0,y0,a       x:(r0)+,x0     y:(r4)+,y0

                    ; Last multiply and accumulate with rounding
                    macr        xO, y0,a      (ro)-
                    ; Put data in right channel too
                    move        a,b
;**** ***** *********************************************
```

In the above program the filter coefficients are located in y memory and the analog signal data is stored in x memory. By locating the data and coefficients in two different memory spaces, parallel access may be executed, increasing execution speed. A function generator can provide a suitable input and an oscilloscope or headphones can be used to monitor the output.

The next laboratory project is a three band equalizer. TTL inputs on the 56002 EVM allow the user to select low pass, band pass, or high pass attenuation. We designed a small circuit board with a dip switch which mounts directly on the EVM. Pull-up resistors are connected to the input side of the dip switches and the other side of the dip switches is connected to ground. A low results when the switch is closed and high when open.

The last two laboratories of the class are an Infinite Impulse Response (IIR) falter and a fast fourier transform (FFT). The code for the IIR filter is quite compact, but the FFT algorithm is somewhat complex involving nested loops to solve the butterflies for the Decimation-in-Time butterflies! Fortunately Motorola provides a good reference with 56002 implementations of a FFT implementations

The file “mist.asm” included in the above programs bridges the codec program supplied by Motorola with the programs in this paper. The code for “mist.asm” is as follows:

```
;*********************************************
; File MISC.ASM
; Put this program in the directory with the ASM files
;****   ******************************************


            org         p:
main
; Initialize SC0,SC1 as GPIO inputs
            bclr        #0,x:PCC
            bclr        #1,x:PCC
            bclr        #0,x:PCDDR
            bclr        #1,x:PCDDR
; Enable SS1 interrupts
            movep       #$fb00,x:CRB
            movep       #$3000,x:IPR
            jmp         user_prog
```

Summary

It is easy to implement real-time processing in a DSP course. Adding a real time laboratory to a DSP course is inexpensive and if students are already familiar with microprocessors it does not take much classroom time to teach DSP processors. A complete list of the programs used for the course is available; contact me by e-mail at pfile @ tech.iupui.edu.

1. Mohamed El-Sharkawy, *Real Time Digital Signal Processing Applications with Motorola's DSP 56000 Family,* Prentice Hall, Englewood Cliffs, New Jersey, 1990.

2. *DSP56000 Digital Signal Processor Family Manual,* Motorola Inc, Phoenix, Arizona, 1992.

3. Thomas P. Krauss, Loren Shure, and John N. Little, *Signal Processing Toolbox for Use with MATLAB,* The Mathworks Inc., Natich, Massachusetts, 1993.

4. Emmanuel C. Ifeachor and Barrie W. Jervis, *Digital Signal Processing-A Practical Approach,* Addison-Wesley Publishing Company, Reading, Massachusetts, 1993.

5. Guy R. L. Sohie and Wei Chen, *Implementation of Fast Fourier Transforms on Motorola's Digital Signal Processors,* Motorola Inc., Phoenix, Arizona, 1993.

RICHARD E. PFILE is an Associate Professor of Electrical Engineering Technology at Indiana University Purdue University Indianapolis. He received his B.S. from the University of Louisville and his M. Eng. from the University of Michigan. He has twelve years of teaching experience and eight years of industrial experience, including three years as a systems engineer.