Getting A Jump Start With the TI TMS320C6713 Digital Signal Processor

James E. Cross

Electrical Engineering Department
Southern University, Baton Rouge, LA. 70813
Email: cross4153@aol.com
Phone: (225) 775-4153

ABSTRACT

Southern University is upgrading its Digital Signal Processing Laboratory with Texas Instruments TMS320C6713 (C6713) Digital Signal Processor Starter Kits (DSKs). The TMS320C6000 Digital Signal Processor (DSP) series is Texas Instrument's most powerful DSP processor. The C6713 is the latest in this series. A number of wide bandwidth analog expansion daughter boards are also being installed as part of this laboratory upgrade. During the 1980's, with the continuous increasing speed of digital computers, it became apparent that digital signal processing would become a viable alternative to analog signal processing. In recognizing this trend, digital signal processing lecture and laboratory courses were introduced into the Electrical Engineering curriculum at Southern University during the early 1990's. Both Motorola and Texas Instruments digital signal processors have been used in the laboratory. The DSP boards have included an analog to digital converter to digitize the signal and a digital to analog converter to change the signal back to the analog form after processing. Digital signal processors are special purpose microprocessors with architecture and instruction set especially designed for real-time signal processing. The TI C6713 DSK is a stand alone development system having all necessary parts to perform experiments. The kit includes a floating-point digital signal processor and a 32-bit stereo codec TLV320AIC23 (AIC23) with variable sampling rates from 8 kHz to 96 kHz for A/D and D/A conversion. The architecture of the C6713 includes 264KB of internal memory, eight functional or execution units composed of six ALUs and two multiplier units, a 32-bit address bus to address 4GB (gigabytes) of memory, and two sets of 32-bit general-purpose registers. Various types of filters being studied in the lecture are designed and tested with this equipment. The TI Code Composer is included for software development using the C language. However, additional support software can be very helpful in getting started with laboratory work. The information presented in this paper will provide a jump start for those who purchase this equipment.

Introduction

A course in Digital Signal Processing has become standard in many Electrical and Computer Engineering curricula. As with other topics, the learning process is greatly enhanced when laboratory experiments accompany material being presented in the lecture. Many universities are therefore adding a real-time DSP lab to accompany the DSP lecture. The DSP lab at Southern University has been in the curriculum for over ten years. The laboratory has been taught using Texas Instruments TMS320C25, C31, and C6211 DSPs, and with the Motorola DSP56000 DSP. The TI C6713 is the latest and most advanced TI DSP. We have upgraded our laboratory with twenty of the C6713 DSKs. TI has a very viable University Relations Program and is aggressively promoting the C6713 DSK. As a result, a number of universities are beginning to design their DSP laboratory around this kit. The objective of this paper is to assist other faculty members in placing this kit into operation with the least amount of difficulty.

A number of TI third party vendors market the C6713 DSK. A Technical Reference Manual and a CD with software is supplied with the kit. The Technical Manual provides a description of the C6713 DSK, describing the operation of the major board components, the physical layout and the connectors. The software CD is the TMS320C6000TH Code Composer Studio. There is a file on the CD in the documents (docs) folder with the title “TMS320C6000 Code Composer Studio Manuals”. It contains many (but not all) TI C6000 DSPs manuals. These manuals, and others, can also be found on the TI Internet website. One document listed under “Application Reports” is report number SPRA474 with the title How to Begin Development Today With the TMS3206211 DSP. Several other documents give information on C6000 DSPs in general. However, none of the documents on the CD has C6713 as part of their title. This is undoubtedly because the C6713 has been available for less than two years. A document, SPRA809, is found on the TI website with the title How to Begin Development Today With the TMS320C6713 Floating-Point DSP. However, this document does not present actual experiments. Instead, it gives general information such as a comparison between the various C6000 DSPs, sources of support, etc. TI recently announced the availability of a free Teaching ROM for the TMS320C6000 DSPs. However, shortly thereafter (early January 2005) TI states that due to its popularity, there is a back-order and it will be delivered as soon as the stock is replenished. As an alternate source for finding teaching material, other persons who have experience using this kit should be sought. A few faculty members have posted their preliminary laboratories on the Internet. A limited number of persons have published books with practical teaching material. One such person is Dr. Rulph Chassaing. Over the last ten years, he has taught DSP workshops for other faculty members, such workshops being sponsored by the National Science Foundation and Texas Instruments. He has published books on the TI TMS320C25, C30, C31 and C6x. His latest book with the title Digital Signal Processing and Applications with the C6713 and C6416 became available during December 2004. This is an excellent resource for getting started with the C6713 DSK. The material presented below is mainly based on the first two chapters of this text.

Equipment and Parts Needed to Get Started

A rather modern computer is needed so as to meet the minimum hardware requirements. These minimum requirements for configuration are given in a Quick Start Installation Guide. The DSK is a desktop board with dimensions of approximately 5 inches by 8 inches. A USB (serial) cable is supplied to connect the host computer to the DSK. The board also ships with a 5 volt external power supply. On board is a 32-bit stereo codec TLV320AIC23 (AIC23) with variable sampling rates from 8 kHz to 96 kHz for A/D and D/A conversion. The codec has two input connections and two output connections for interfacing to the analog world. These are the microphone input, the line input, the line output and the headphone output. The minimum requirements to get started will be a pair of headphones or a set of speakers. This will be sufficient for the first set of experiments as will be discussed in the next section. A signal generator and an oscilloscope will be needed to perform most of the experiments. However, to get started, the output from a CD player (or similar device such as a small keyboard) can be used as the input and a set of speakers can be used for the output. As mentioned above, a CD with the Code Composer Studio (CCS) is shipped with the DSK. The CCS provides an integrated development environment (IDE) for developing, debugging and executing programs. The beginner should write programs using the C language (rather than assembly language), compile, link and execute the programs. This and more can be done using CCS. It also has graphical capabilities for plotting results. An initial set of experiments are provided to make certain that the DSP board is operating properly.

Initial Experiments with the DSK: Checking it Out

Use the CD provided to install the Code Composer Software (CCS) on your PC using the Quick Start Installation Guide. Use the director c:\C6713 when installing the software rather than the default director c:\ti if you wish to be able to use the exact instructions given later for locating and loading files. An icon for Code Composer should be created on the desktop. You can either connect headphones to the headphone output jack or speakers to the line output jack or both headphones and speakers can be connected. It will be assumed that speakers will be used. (If you have speakers connected to your computer, you can simply disconnect them and connect them to the DSK instead). The DSK has 4 addressable LEDs and 4 addressable DIP switches which provide a simple way to communicate with the board. The LEDs are given labels from 0 to 3 and the switches are given labels from 0 to 3. Connect the power supply to the DSK and use the USB cable to connect the DSK to the host computer. As a self-test, when the power is connected, the 4 LEDs will blink 3 times and then stay on. In addition, a 1 kHz signal will sound through the speakers for about a second. The DSK is functioning OK if it passes this test.

Launch Code Composer Studio by double clicking on the CCS icon. Note that the power should be on to the DSK and the USB cable connected when CCS is launched. If it is not connected, a message will appear stating that it was not possible to initialize the DSK. The 4 LEDs will turn off when CCS is started.

Texas Instruments has lots of documentation on its DSPs, some of the documents being more than 600 pages long. It is helpful to know which documentation is needed to accomplish your particular task. If you are not overly impatient, you might take an hour or so to read the first two chapters of Code Composer Studio Getting Started Guide, Manual number SPRU509. It is in the document directory of CCS (and also found at the TI website). However, if you want to get started as quickly as possible, you can take advantage of the experiences of others who have used the DSK. As mentioned in the introduction, the text book Digital Signal Processing and Applications with the C6713 and C6416 by Dr. Rulph Chassaing will provide a quick start. All the files or programs listed in the text, except for some student projects, are included on the CD that accompanies the book. The CD has a C language file for initializing the DSK. This file is not contained in the CCS. Also include are two other files that have been modified from the files in the CCS. The needed support files from the CCS software are documented. A summary of the three examples from Chapter 1 is as follows.

Example 1.1: Sine Generation Using Eight Points with DIP Switch Control (sine8_LED.c)

The first example is a program written in C that will generate a sine wave using 8 points in a lookup table. The process begins by using the CCS to create a “project”. Detailed instructions and explanations are given of the needed support files, their location, how to add the files to the project, how to compile, link, and load the files, and how to start and halt execution. Information is given on the sampler for A/D conversion. The various instructions in the C file are explained. For example, it is explained that the sampling rate is set in the instruction “Unit32 fs = DSK6713_AIC23FREQ_8KHZ”. Here the rate is set at 8 kHz. Changing the 8 to 16 will change the sampling rate to 16 kHz. The instructions to read from the DIP switches and to write to the LEDs are apparent. The program determines if DIP switch 0 is pressed, turning on LED 0 if it is pressed. In addition to turning the LED on, a 1 kHz signal is sent to the speakers and or the headphones. An oscilloscope could be connected across the headphones, or instead of the headphones, to verify the frequency. After executing the program, you can try your programming skills by reading from a different DIP switch and turning on any of the 4 LEDs. The frequency can be changed by changing the sampling rate or changing the number of points in the lookup table. The various compiler and linker options are explained. One has the option of generating an assembling language file, generating inline code, selecting various levels of optimization, etc. Addition features demonstrated are the use of the “Watch Window” and the operation of a “slider” within a General Extension Language (GEL) file to slide through different values of a variable (changing the values of that variable) while the program is being executed.

Example 1.2: Generation of the Sinusoid and Plotting with CCS (sine8_buf.c)

This example uses the same method as in Example 1.1 to create a 1 kHz signal; however, whereas the first example used polling to determining when data was ready, this example uses the interrupt method. A buffer is created to capture and store 256 output data samples in the internal DSK memory. The length of the buffer can easily be changed. Two methods of plotting the data are demonstrated using the CCS. The sine wave is plotted in the time-domain and a fast Fourier transform (FFT) is used to obtain a frequency plot. The frequency plot produces an impulse (or spike) on the frequency axis at 1 kHz.

Example 1.3: Dot Product of Two Arrays (dotp4.c)

The convolution process is used when performing finite impulse response (FIR) filtering. This process involves a series of multiplications and summing. It is referred to as the multiply and accumulate operation, and is the biggest claim to fame for DSPs. Digital signal processors have an instruction that will multiply two numbers and add the results to a previous sum all in one cycle. The TI C6713 can perform two such operations in one cycle (having parallel computing capability). This particular example performs the dot product of two matrices, multiplying a row matrix, x, by a column matrix, y, resulting in a scalar. The values x = [1 2 3 4] and y = [0 2 4 6]' are used (the y matrix having the transpose symbol). The result of this is (1 x 0) + (2 x 2) + (3 x 4) + (4 x 6) = 40. The x and y matrices are defined in a header file called dotp4.h. This header file can be edited to use a different set of values if you so desire. You can also change the problem so there will be more than four values but if this is done, it will be necessary to edit the source file dotp4.c to change the instruction that gives the length of the arrays. The main purpose of this example is to demonstrate the compiler optimization facility and how a portion of code can be benchmarked (such that the time of execution can be determined). Breakpoints are set for the multiply and accumulate portion of the code and a comparison is made between executing times with and without optimization. The results were that 191 cycles were required without compiler optimization whereas only 25 cycles were required with compiler optimization.

Gaining Confidence with Input Output Programming

Nineteen examples are presented in chapter 2 but only two will be presented here. They will be labeled Example 1 and 2 but the second example is actually the eighteenth example in the chapter.

Example 2.1: Loop Program Using Interrupt (loop_intr.c)

The first example in chapter two is called a looping program. It simply reads in data from the A/D converter and writes out data through the D/A converter. This particular example uses the interrupt method to determine when data is ready. The second example in chapter 2, which will not be presented here, is similar except that it uses the polling method to determine when data is ready. These first two examples in the chapter are very important in that one of them will be used as the “seed” program for all the input-out problems. Below are the two lines (along with a comment) that basically show how the program works.

```
x = input_sample();
// The processing algorithm goes here.
output_sample(x);
```

As mentioned above, C6713dskinit.c is a file on the CD supplied with the book by Dr. Chassaing. It is not part of CCS that accompanies the DSP kit. The program loop_intr.c is only 14 lines long. This looping program has been kept short by having the program C6713dskinit.c added during the project build process. The two functions above, input_sample() and output_sample(), along with another function that is used, comm._intr, are defined in the program C6713dskint.c. This makes writing input-output programs very easy. A statement, while(1), makes this a “loop for ever” type program. The program is interrupt driven with an interrupt occurring every sample period. The ability to change the sampling period is given with the same statement noted in Example 1.1.

The comment “The processing algorithm goes here” have been injected. Also, simple names x and y have been given to the variables. A data sample is read in and is given the name x. A real-time 21 tap FIR filter could be implemented using the convolution as y = x*h (where * denotes the convolution). A set of 21 impulse response values, h, could be determined based on the type of filter, and 21 initial pieces of data read in and stored. The algorithm to compute a value for y using the convolution would be programmed and the value of y outputted. A new value of x would then be acquired, the 21 values of x shifted right, with the oldest value in of x shifted out and the most current 21 values retained. The program would loop back to the input statement and the procedure repeated “forever” (until operator intervention is made to halt execution).

Example 2.1: Use of External Memory to Record Voice (record.c)

This program demonstrates the recording of sound, storing it in external memory (on-board the DSK) and then playing it back. As much as 5 minutes of data (for the latest version of the DSK) can be stored and played back. A statement is given defining the section (type) of memory to use for storage. A buffer is declared along with statements designating the number of samples to acquire and store, and designating the sampling frequency. There are basically two parts to the program. The first part reads in data as long as DIP switch #3 is pressed. (LED #3 will be turned on to indicate that data is being acquired). The second part of the program outputs the data when DIP switch #0 is pressed. (LED #0 will be turned on to indicate that data is being sent out). This program was tested using a very inexpensive CD player as the input source for sound. The speakers were disconnected from a computer and used as the output device.

Some Concluding Remarks

The Texas Instruments TMS320C6713 DSP Starter Kit (DSK) is an excellent system for demonstrating real-time digital signal processing principles. TMS320C6x is the designation for all TI DSPs in the TMS320C6000 series. With the C6713 having been introduced less than two years ago, there is a limit amount of documentation that provides step by step instructions for implementing experiments with the TMS320C6713 DSK. Many lines of code are needed to place the DSK into operation. An understanding of this detail code is necessary to master the many applications of this processor. However, one can conduct a basic Digital Signal Processing laboratory with only a limited knowledge of the underlying code. It can be very frustrating if half of a semester is needed to study documentation before a meaningful laboratory can be performed. The text Digital Signal Processing and Applications with the C6713 and C6416 by Dr. Rulph Chassaing is designed to give the instructor and students a jump start in quickly getting a DSP laboratory up and running using the TI TMS320C6713 Digital Signal Processor Kit.

REFERENCES

1. R. Chassaing and D. W. Horning, Digital Signal Processing Laboratory using the TMS320C25, Wiley, New York, 1990.
2. R. Chassaing, Digital Signal Processing with C and the TMS320C30, Wiley, New York, 1992.
3. R. Chassaing, Digital Signal Processing Laboratory Experiments Using C and the TMS320C31 DSK, Wiley, New York, 1999.
4. R. Chassaing, DSP Applications Using C and the TMS320C6x DSK, Wiley, New York, 2002.
5. R. Chassaing, Digital Signal Processing and Applications with the C6713 and C6416 DSK, Wiley, New York, 2005.
6. K. S. Lin (Editor), Digital Signal Processing Applications With the TMS320 Family, Dallas, Texas Instruments Incorporated, 1986.
7. M. El-Sharkawy, Real Time Digital Signal Processing Applications with Motorola's DSP56000 Family, Prentice Hall, Englewood Cliffs, 1990.

7. A. V. Oppenheim and R. W. Shafer, Digital Signal Processing, Prentice Hall, Englewood Cliffs, 1975.

JAMES E. CROSS
Mr. Cross is an Associate Professor of Electrical Engineering at Southern University in Baton Rouge, LA where he has served since 1962. His research areas of interest are Digital Signal Processing and Real-time Computing. He has a wide range of research experiences to include work for IBM, Texas Instruments and the Wright-Patterson Airforce Research Laboratory.