Digital Signal Processing Fundamentals

D. Koenig

Introduction[1]

Recent advances in digital signal processing (DSP) technology make it easier for scientists to develop powerful personal computer-based data acquisition and analysis systems. This application note discusses DSP technology from a hardware, software, and application point of view. The Reduced Instruction Set Computer (RISC) architecture of the Texas Instruments (TI) TMS320C30 chip is used as an example of the power of a DSP processor. For software, programming methods–from low-level machine code to high-level DSP operating systems–are compared. The application note concludes with examples of how DSP is being integrated into test-and-measurement systems ranging from audio and speech analysis to image processing and real-time control.

Defining DSP

Perhaps the most common misconception among prospective users of DSP technology is that DSP is for signal processing only–if a person is not doing Fast Fourier Transforms (FFTs), then he or she has no use for DSP technology. Unfortunately, by limiting their perspective of DSP, these prospective users are missing some of the most powerful numerical processing capabilities available today.

The term *DSP* is somewhat misleading because it is usually associated with the FFTs, digital filters, and spectral analysis theories that are taught as part of the classic DSP curriculum. This is not to say that DSP is unrelated to frequency domain analysis techniques–spectral analysis remains an integral part of DSP technology.

A more appropriate definition for DSP as it applies toward the computer industry can be derived from the name *digital signal processing* itself–DSP is the *processing* of analog *signals* in the *digital* domain. Real-world signals, such as voltages, pressures, and temperatures, are converted to their digital equivalents at discrete time intervals for processing by the CPU of a digital computer. The result is an array of numerical values stored in memory, ready to be processed.

DSP is useful in almost any application that requires the high-speed processing of a large amount of numerical data. The data can be anything from position and velocity information for a closed-loop control system, to two-dimensional video images, to digitized audio and vibration signals.

---

[1]Product names listed are trademarks of their respective manufacturers. Company names listed are trademarks or trade names of their respective companies.

This application note describes DSP from an application point of view to demonstrate the many different and effective uses for digital signal and array processing. The common factor in all of these applications is the need to do extremely high-speed calculations on large amounts of data in real time.

What Is Real Time?

The concept of *real-time* processing is so subjective that it is almost impossible to apply an absolute definition to it. For example, a real-time temperature monitoring system might require an update rate of only 5 times/sec to control the climate of an environment, whereas a real-time digital audio system might require the ability to process data at up to 48 kHz, or 48,000 times/sec. Thus, a system that operates in real time for temperature monitoring would be inadequate for a digital audio application. Likewise, a system that operates in real time for a digital audio application might be inadequate for radio- or microwave-frequency experiments.

What is a good definition of real time? A good definition for this subjective concept is itself subjective–real-time processing depends upon the situation at hand. If an application to be controlled or analyzed requires that time-critical functions be completed within a given time interval, then a system can be called a real-time system if it will completely execute the necessary functions within the given time interval *for all cases.* More specifically, real time implies real time within the constraints of the system of interest.

Plug-in DSP Boards Deliver Real-Time Power

You can attain real-time processing on the PC today with a fundamental change in the concept and architecture of personal computing. System designers are introducing DSP boards that plug into PCs to produce multiple-processor systems that are more powerful in computationally intensive tasks than many single-CPU minicomputers and workstations.

Plug-in DSP boards take advantage of the powerful digital signal processors being developed by companies like TI, AT&T, and Motorola. DSP chips such as the TI TMS320C30, illustrated in Figure 1, can perform 33.33 million floating-point operations per second (MFLOPS) or 16 million instructions per second (MIPS). The number of MFLOPS performed is an indicator of the *processing power* of a chip.

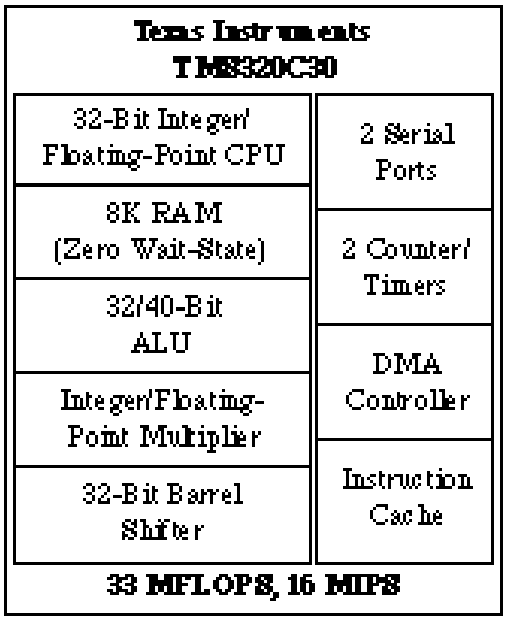

Figure 1. Overview of TI TMS320C30 DSP Chip Functionality

DSP chips are high-speed, dedicated microprocessors that have been optimized to perform arithmetic operations on the huge amounts of data required by spectral analysis and signal processing algorithms. They are similar in architecture to RISC processors, in which a small set of frequently used instructions has been optimized at the expense of less frequently used operations. Additionally, nearly all of these instructions execute in a single CPU clock cycle.

In addition to the high-performance CPU, many DSP chips include I/O functionality, timing circuitry, and high-speed memory onboard the chip itself. Serial ports, counter/timers, direct memory access (DMA) controllers, and instruction caches work together with the CPU to provide the DSP board with maximum performance and throughput.

With the addition of a plug-in DSP board to the PC, the basic architecture of the personal computer is changed from a single-CPU system to a dual-processor workstation. Real-time tasks can run on the optimized hardware of the DSP board while system-level requirements, such as I/O and disk accesses, are handled by the PC microprocessor. With an appropriate interprocessor communication protocol, the two CPUs run independently of one another, and the system (PC and DSP board) achieves real-time performance.

Software for Maximum Flexibility and Performance

Until recently, designers had to program the DSP hardware at its lowest level to fully use its real-time processing power. The idea of using assembly language to program a DSP chip was often too daunting for all but the most experienced software engineers. Thus, a technology barrier existed between the powerful DSP hardware and the ability of the average scientist or engineer to program it. Many system designers who could benefit greatly from DSP technology did not use it because of the seemingly insurmountable obstacle of software development.

Vendors are breaking this technology barrier by offering a variety of software tools for the new generation of DSP plug-in boards. These software products, ranging from low-level debugging software to high-level DSP operating systems, make programming the DSP hardware as easy as possible.

DSP software tools can be divided into five categories–hardware evaluation and development tools, optimizing cross-compilers, off-the-shelf DSP libraries, block diagram development environments, and special DSP operating systems.

At the lowest level are the traditional DSP *hardware evaluation and development tools*, including assemblers, linkers, and debuggers. Maximum performance can be achieved by programs developed in this environment, because such programs take full advantage of the hardware. The obvious disadvantage is the requirement for extensive programming expertise and special knowledge of DSP hardware.

The next category includes *optimizing cross-compilers*, which convert programs written in high-level languages like C into their assembly language equivalents. Code development in high-level languages represents an enormous increase in software productivity, providing results comparable to assembly language in a fraction of the development time. There is no such thing as a perfect cross-compiler, however, and cross-compiled programs are neither as efficient nor as fast as programs developed in assembly language.

Manufacturers of DSP hardware also offer *off-the-shelf DSP libraries* of ready-to-use functions, such as FFTs, filters, and windows, which are common to most DSP applications. Off-the-shelf DSP libraries relieve programmers from having to develop these algorithms and require no special knowledge of the DSP hardware to use them. These libraries represent a finite number of the DSP routines, however, and the user must develop additional routines to tailor the functions for a working application.

Developers can also design their applications using *block diagram development environments*. An example of a block diagram development environment is shown in Figure 2. Most DSP developers start with a conceptual drawing, or block diagram, of the system they are building. Using the block diagram approach, you can construct signal-flow diagrams on the computer and compile them to take advantage of the DSP hardware. Block diagram development is limited only by the amount of DSP functionality provided by the application program and memory constraints of the computer.

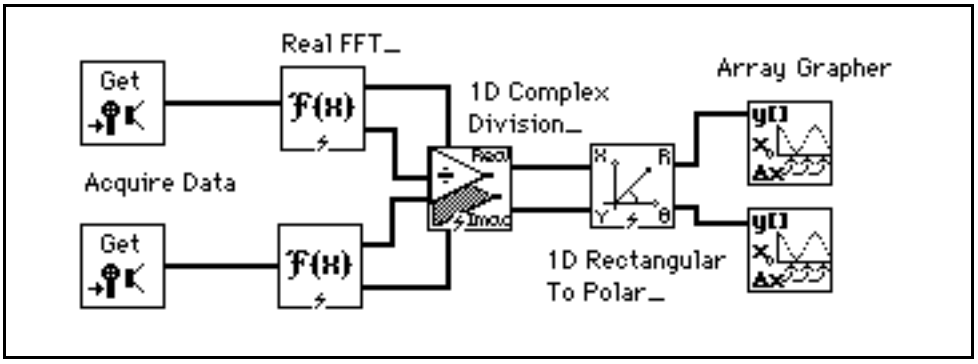

Figure 2. Block Diagram Development Environment

At the highest level, special *DSP operating systems* provide software primitives, such as I/O and memory management, that comprise a virtual DSP computer. Because most DSP chips are designed without regard to the platform on which they will run, operating systems are not usually included with them. The intent of the DSP operating system is to make programming DSP hardware similar to programming general-purpose CPUs. The added functionality of the operating system, however, introduces a substantial increase in software overhead.

Applications

The performance and versatility of plug-in DSP boards bring real-time processing power to a wide variety of application areas. In this section, several examples of DSP technology are presented. In each case, the applications were developed using a Macintosh II computer, a TI TMS320C30-based DSP board, two 16-bit ADCs, and two 16-bit DACs (for cases requiring data acquisition). For software, several options were used, including a cross-compiler for C, a DSP assembler, an off-the-shelf analysis library, and a block diagram development environment.

General-Purpose DSP

DSP chips are optimized to efficiently process large buffers of data, making them ideal for numerically intensive DSP algorithms, such as FFTs, convolutions, and digital filters. These algorithms involve repetitive multiplications and additions on data that is often sampled at rates exceeding 8 ksamples/sec.

DSP chips can execute more than one operation at a time. The most common example is parallel multiplication and accumulation. A multiplication and an addition can be made in a single instruction cycle. Signal processing applications can be designed to take advantage of these multiple operations to significantly increase execution speed.

Application: 1,024-Point Real FFT

An optimized 1,024-point real FFT takes 198 msec to execute on the Macintosh II CPU. The same algorithm optimized for the TMS320C30 executes in 3.48 msec. In terms of real-time processing on a signal being acquired, a 200-msec execution time represents an update rate of 5 times/sec, with an effective throughput of 5 ksamples/sec. With a DSP chip, the update rate is increased by over 50 times to almost 300 times/sec, for a throughput of 300 ksamples/sec. Table 1 compares the processing speed of the Motorola 68020 to the TI TMS320C30.

Table 1. Benchmarks for Typical DSP Functions

DSP Function	Benchmarks (msec)	
	Motorola 68020	TI TMS320C30
Real FFT	197.0	3.48
Power Spectrum	200.9	3.70
Convolution	16,950.0	65.32
Butterworth Bandpass Filter	119.3	5.05
Matrix Multiplication	30.7	0.20
Matrix Inversion	33.2	0.24
Note: All arrays are 1,024 points. All matrices are 10 by 10.		

Real-Time Control

One of the most useful benefits of including a DSP board in real-time control applications is that time-critical tasks can be distributed among the DSP processor and the host CPU for maximum performance. Real-time tasks can be programmed to run on the DSP chip without the threat of interruption, while other noncritical tasks are executed on the main CPU. With the tasks divided in this manner, system-level interrupts, such as keyboard input and disk accesses, are serviced by the main CPU and do not effect the real-time program running on the DSP board.

DSP is a powerful solution for functions that must execute within a specified time interval, regardless of external interruptions. In the past, closed-loop programs running on the main CPU were at the mercy of unexpected system-level interrupts. Distributed programs that use a DSP board for dedicated parallel processing now eliminate that worry.

Application: Closed-Loop Control

One of the best-known implementations of a closed-loop controller is the Proportional-Integral-Derivative (PID) controller. The digital transfer function of a PID controller can be implemented easily in software on a DSP chip. This chip, with fast ADCs and DACs, can be used to create an efficient closed-loop controller. With the instruction cycle of the TMS320C30 being 60 nsec, a typical PID algorithm executes in less than 1 µsec. Assuming arbitrarily fast ADCs and DACs, you can implement a real-time PID controller with a maximum processing rate greater than 1 MHz.

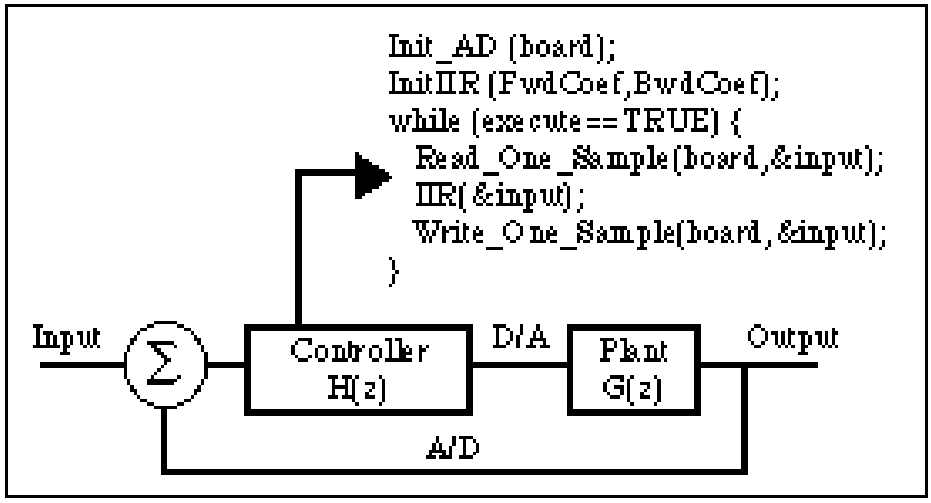

Figure 3. Digital Implementation of a PID Controller
Using a One-Biquad IIR Filter

Graphics/Imaging

Graphics and image processing are two application areas for which the advantages of DSP are not readily apparent. Graphic images are two-dimensional arrays of numbers. DSP hardware is optimized to operate on arrays and is easily programmed to process the huge amounts of data needed for image processing.

Application: 128-by-128 Real Two-Dimensional FFT

As with its one-dimensional counterpart, the two-dimensional FFT is a key function in two-dimensional image processing. A two-dimensional FFT, shown in Figure 4, is implemented by running an optimized one-dimensional FFT on each row of the two-dimensional image, then running the one-dimensional FFT on each of the columns of the resulting real and imaginary matrices. Using the Macintosh CPU, a 128-by-128 two-dimensional FFT executes in 11.47 sec. The same algorithm coded to run on the DSP hardware executes in 0.23 sec.

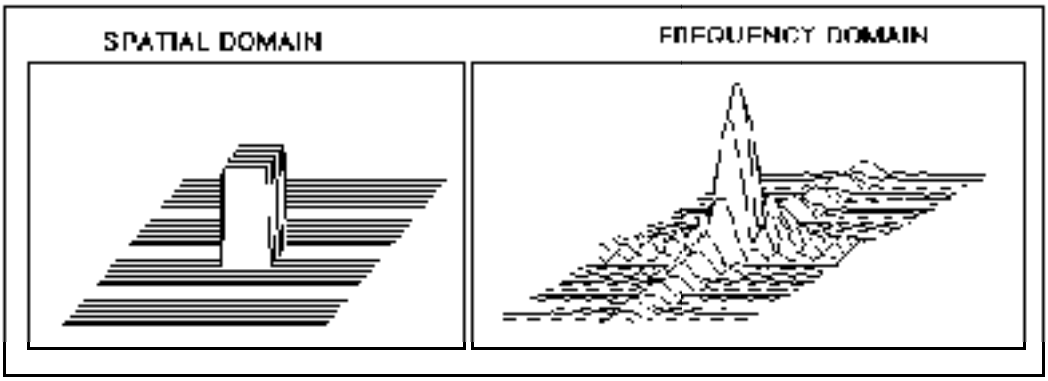

Figure 4. Two-Dimensional FFT

Acoustics and Speech Processing

Acoustics and speech applications operate on data in the audio frequency range (DC to 20 kHz). Acquisition and analysis programs working in this type of environment most commonly use sampling rates between 8 kHz and 48 kHz to fully characterize the audio frequency data.

With input signals near the upper audio frequencies, data must be acquired at up to 48 kHz to satisfy the Nyquist criterion. At this rate, ADCs produce one point every 20.83 µsec, or 1,024 points every 21.3 msec. For real-time processing of acoustic signals, execution of analysis algorithms must satisfy this constraint. Not much work can be done by a general-purpose microprocessor in only 20.83 µsec. However, a DSP chip with a 60-nsec instruction cycle, such as the TI TMS320C30, can execute almost 350 instructions in that amount of time.

Notice that for lower sampling rates, such as 16 kHz or 22.05 kHz, processing time is less stringent and real-time analysis is easier to achieve.

DSP technology has been an integral part of acoustics and speech processing for years. Recent advances in digital audio research, speech recognition, radar, and sonar would not have been possible without the real-time processing capabilities of DSP.

Application: 10-Band Stereo Graphic Equalizer

A 10-band graphic equalizer, shown in Figure 5, is implemented using a TI TMS320C30-based DSP board and an audio-frequency I/O board. Stereo data is sampled at 48 kHz, filtered by 20 bandpass filters (10 per channel), scaled by a user-selectable gain value at each frequency, and recombined for output. All of this processing–two A/D conversions, 20 bandpass filters, 20 instantaneous power calculations, combination of the 20 bandpass outputs, and two D/A conversions–is completed in less than the 20.83 µsec allotted by the 48-kHz I/O sampling rate.

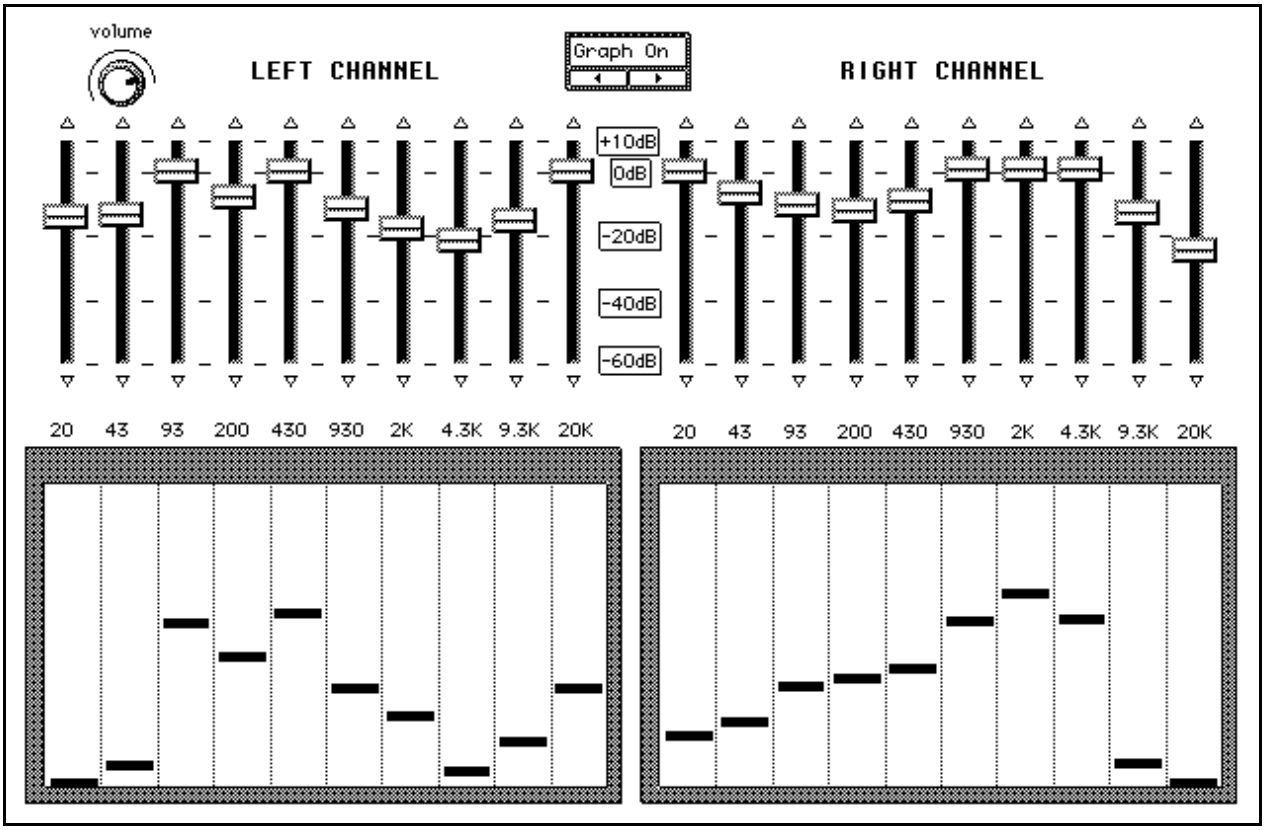

Figure 5. 10-Band Stereo Graphic Equalizer

Instrumentation

Another type of application made possible by DSP technology is *real-time virtual instrumentation*. With the wide selection of plug-in A/D boards, GPIB interfaces, and waveform generators available, computers can be programmed to implement the characteristics of many stand-alone laboratory instruments. Plug-in DSP boards add the power of real-time processing to these virtual laboratory instruments.

In addition to the power that PC-based virtual instruments (VIs) have to offer, virtual instrumentation also has flexibility and cost advantages over traditional laboratory instrumentation. With virtual instrumentation, the functionality of the instrument is implemented in software,

interfacing ADCs and DSP boards into a desired analysis system. Changing the functionality of a VI is as simple as changing the software that defines the flow of data within the instrument. Virtual instrumentation also reduces equipment costs. There is no need to purchase separate instrumentation for each function in the experiment.

Summary

By changing the fundamental architecture of the personal computer into a parallel-processing environment, DSP boards provide a quantum leap in performance over many powerful single-CPU computers. Numerically intensive applications can be implemented on PC-based acquisition and analysis systems, replacing dedicated test-and-measurement equipment. With the powerful software support available for these plug-in DSP boards, system designers can complete in weeks what once took months or even years of research and development.

DSP technology is the choice of scientists and engineers who want a cost-effective solution to their processing needs. With the wide selection of hardware and software systems available, researchers can harness the power of DSP for use in real-time analysis applications. The power of DSP is limited only by the developer's ability to combine these components into real-time processing environments.