DATA TRANSLATION®

Embedded Signal Processing System: Build or Buy?

If your design needs to measure and process signals in real-time, do you establish your own development effort or form a partnership with a skilled vendor?

INTRODUCTION

Engineers confronting the need to add real-time signal processing capabilities to a product will consider the build-to-buy question. Their decision must address the critical issues of time-to-market and more importantly, time-to-profit. Beyond the technical issues, engineers and their managers also must consider:

- The cost benefit and risk analysis of project management
- Design and staffing requirements
- Manufacturing issues, and the need to sustain and support a complex product

As part of the analysis, engineers should carefully search for reliable partners who already have addressed these issues and who offer their expertise at reasonable cost. Armed with this information, it is easier to address the build-or-buy question.

BUSINESS CONSIDERATIONS

For the last twenty years, time-to-market issues have been driving us toward shorter and shorter design cycles. For example, automobiles manufacturers have been transforming their industry to better address the build-or-buy question. In the early 1980's, a typical U.S. manufacturer averaged 60-70 months to design an automobile from concept to production.

Today, design cycles are often as low as 20 months. Automotive manufacturers accomplished this amazing transformation, in part, by outsourcing.

Outsourcing the design and manufacture of components and subsystems to skilled suppliers greatly reduces the design cycle. The result, lower expenses and improved quality. Partnerships with skilled suppliers allow businesses to provide a better product at a lower cost.

When a project calls for a few signal-processing boards, buying them from a supplier makes the most sense. On the other hand, if an application requires OEM quantities, adding a digital signal processor (DSP) chip, analog circuits, and software may seem realistic. But, if the application involves intensive data acquisition and analog I/O tasks, don't get swayed by the low cost of a DSP chip. Adding a DSP to a product involves designing peripheral circuitry, careful attention to issues such as power, grounding, and noise, and the development of custom software. It all takes time and money, so how should you proceed?

ANALOG SUB-SYSTEM

If you are thinking about building a signal-processing board or subsystem, first consider the analog front end -- the point where analog signals enter your system. This part of the design presents critical challenges because it involves processing low-level signals when they're most susceptible to noise and distortion. A "poorly captured" analog signal degrades the performance of an entire system. So, your design team needs skills and experience in analog design. Designing an analog to digital converter (ADC) front end is as much an art as it is a skill. Your group should have experience preserving the integrity of incoming analog signals for the ADCs without letting noise cause problems. Typically, the analog input portion of a data acquisition board or system represents its weakest link.

Your team must choose the best ADC to provide the DSP chip with the digital data at a rate and accuracy to match the application. There are several types of ADCs available, from high-speed flash converters and successive-approximation converters to high-resolution delta-sigma ($\Delta\Sigma$) converters used for vibration and sound (audio) signals. Choosing a converter involves a careful balance of ADC sampling rate, resolution, accuracy, harmonic distortion, and other characteristics while also accounting

for power-supply noise, processor noise, and so on. Every clocked circuit on a board will generate noise. So, your design team must know how to specify and test ADCs for their suitability to your application.

Let's say your signal processor will acquire audio signals. A 24-bit ΔΣ ADC will realistically provide 16 to 20 effective bits and a 100 kHz bandwith -- plenty of performance for most audio-frequency signal-processing needs. Keep in mind that a 20-bit conversion provides a a resolution of 1 part in 1 million (220 or 120 dB), essentially the same as detecting a 1-μV change in a 1-V signal. so, even a small amount of noise can greatly affect measurements.

Because a 100 kHz ΔΣ converter "oversamples" its analog input (typically 256 times faster than the required frequency response) it requires only a simple R-C antialias filter to prevent high-frequency signals from affecting a measurement. But if you plan to use another converter type, always consider the needs for more complicated antialias filters and sample-and-hold circuits.

These circuits, placed between analog inputs and an ADC add noise, degrade signals, and cause phase delays. A ΔΣ converter also slightly delays the output of a digital value due to its inherent design. (See references.)

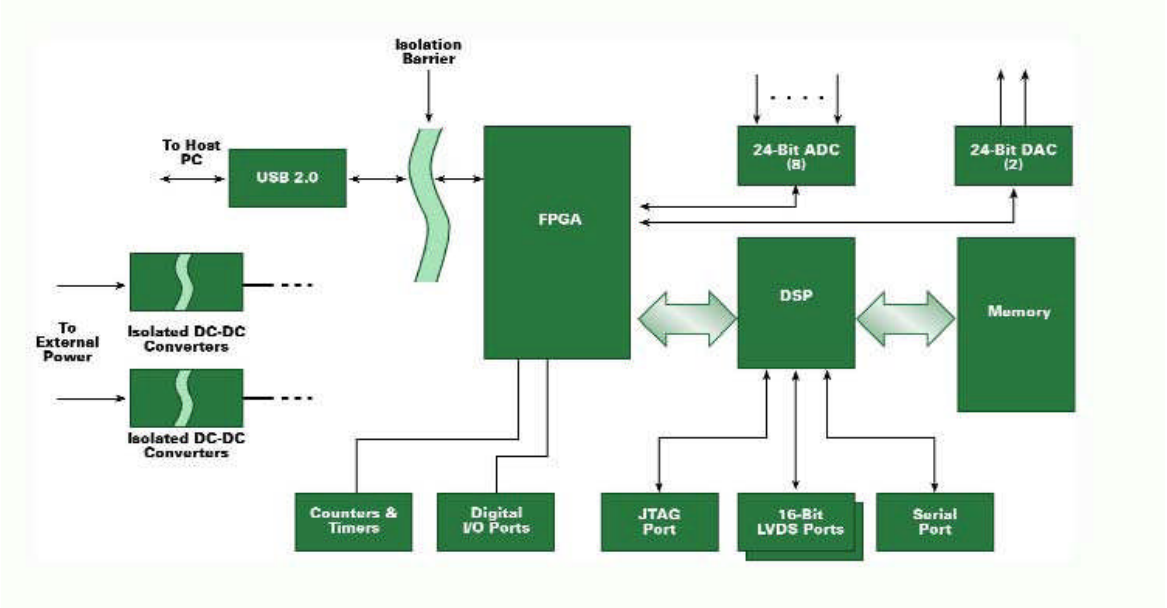
In addition to getting analog signals to ADCs, an analog designer must provide input connections that protect the board and its host computer from transients caused by electrostatic discharge (ESD) events and even from an accidental short circuit with a power line. External electrical problems must be prevented from damaging an expensive signal-processing board. At the same time, those input protection circuits reduce the quality of the analog signal being captured.

DATA MANAGEMENT

Most systems that employ a DSP will process several analog signals at the same time. Thus, designers must determine how to handle the data produced simultaneously by several ADC's. ADC data could flow directly to a buffer memory, which would require support circuits.

Or it could go directly the DSP chip, robbing the chip of processing cycles as it manages data movement. Neither hard-wired approach will adapt readily to changes in ADC types or changes in how ADC data streams, say from parallel to interleaved serial, would demand major circuit and software changes.

An alternate approach involves inserting interface circuits , perhaps a field-programmable gate array (FPGA), in the data flow. An FPGA can simplify the transfer of data to the DSP chip's main memory. The Texas Instruments TMS320C6713 DSP chip, for example, provides a direct-memory- access (DMA) capability that makes it easy to quickly move data from a register to memory, with minimal process involvement. When working with ΔΣ ADCs, the FPGA can be used to separate serial bit streams and reconstruct bytes or words for movement to memory. Using an FPGA, or taking the similar approach to moving data, gives designers the ability to adapt to changes in the types of converters selected and the type of DSP chip selected. Adding the FPGA complicates a design, but without this flexibility designed-in at the start, minor technology changes could require a board redesign. And redesigns are followed by costly debugging and testing.

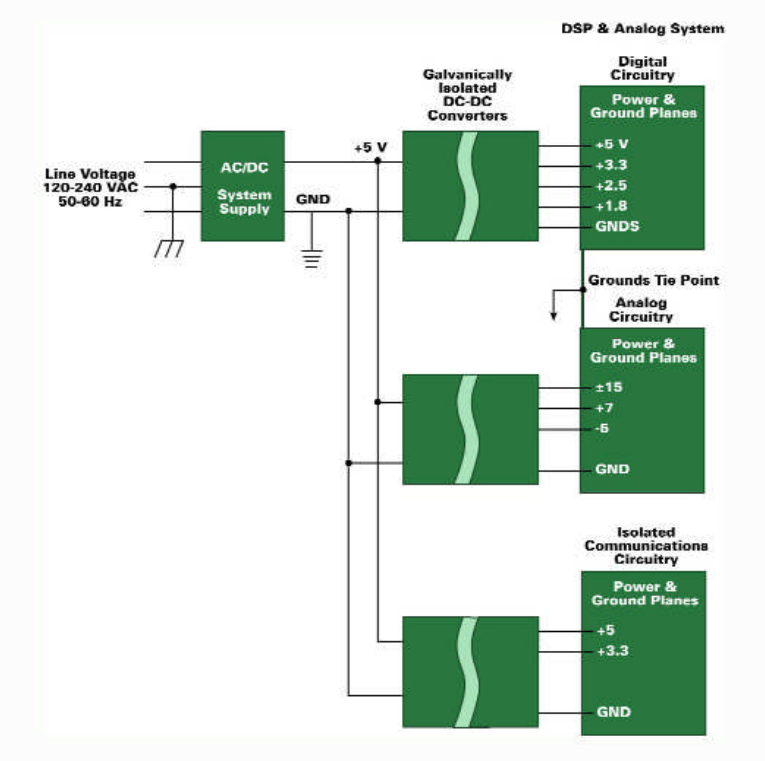


An FPGA manages the transfer of information between a DSP chip and its surrounding circuits. The use of an FPGA adds flexibility to the circuitry so it can quickly adapt to future changes and customer needs.

SYSTEM OVERVIEW

Engineers may tend to focus on the “cool” DSP and ADC circuits and leave power-supply design until the board project enters its last stages. In a DSP-based system, that won’t do. Power supplies require the same attention to detail as the analog front-end. Not only must power-supply circuits provide the sequence of applied voltages on powerup. These circuits -- often switching regulators -- must not radiate noise into sensitive analog circuits. So, a design team needs expertise in power-supply design, as well. Design of power circuits also must provide for the accurate reference voltages needed by the ADCs.

The liberal use of ferrite beads to attenuate high-frequency noise on power lines, and the use of over-sized transformer and inductor cores -- that won’t saturate -- can help reduce noise. So can proper conductor positioning and routing. Designs must carefully separate power and ground signals that feed analog digital circuits. A DSP chip running at 300 MHz, for example, is an excellent noise source. In a poor design, noise will find its way into many analog signals, even creeping into unshielded connectors from nearby PCB traces. Remember, not only does a DSP clock radiate at its fundamental frequency, but the clock produces harmonics, too. And each time a noise producing circuit undergoes a design change, it may affect other parts of the circuit.



The Data Translation DT9841 board uses galvanic isolation to electrically separate sensitive analog circuits from power supplies and communication ports. Careful design of power regulators and analog references ensures high signal integrity.

COMMUNICATIONS

Most signal-processing boards will communicate with other subsystems or host computers through Universal Serial Bus (USB), FireWire (IEEE 1394), Ethernet (IEEE 802), or other busses. Although companies do provide hardware modules and reference designs that ease the task of adding communication capabilities to a design, these modules always require software support. Connecting a communication device to a DSP chip requires special expertise with protocols and driver software. The addition of high-speed communication ports also places yet another noise source in a design. Engineers must thoroughly understand how to attenuate any noise introduced through a communication port, and they many need to isolate such ports from other circuits.

EMI/EMC STANDARDS

Today, an OEM's product must go through rigorous testing to a variety of standards established by the Federal Communications Commission (FCC), and by the International Electrotechnical Commission (IEC). Standards cover radiated emissions, conducted emissions, electromagnetic compatibility (EMC), and electromagnetic interference (EMI). Testing to these exacting standards is a must and it requires using the services of a specialized test lab. Even though a company may perform its own pre-compliance tests, it still must go through a

The DT9841 from Data Translation is a real-time data acquisition USB module with an embedded DSP for high accuracy noise and vibration testing. It features full data acquisition capability: eight 24-bit analog inputs, two 24-bit analog outputs, 24 lines of digital I/O, and three 32-bicounter timers.

OEM VERSION

SLEEK BOX VERSION

Key features of the DT9841 include:

- Delta-Sigma A/D and D/A converters for high-accuracy measurements at up to 800 kS/s (aggregate)
- Real-time control with embedded
- TMS320C6713, 300 MHz DSP for stand-alone or system operation
- Software programmable DSP
- Two packaging configurations: Sleek Box version for easy connections or board-level version for OEM-embedded applications
- Flexible memory configuration: 128 MB SDRAM and 2 MB Flash for autonomous operation
- Scalable design for connecting up to 8 modules for high-channel count applications
- High-speed host USB 2.0 host interface (480 Mbits/seconds)
- 500 V Isolation for maximizing signal integrity

The DT9840 Series Software CD is shipped free with every DT9840 Series module. It contains device drivers for the modules, an API library for developing DSP programs, an API communications library for developing host programs, the DT Dynamic Signal Analyzer application to measure real-time signals without programming, example programs, and a number of utility programs - everything you need to get up and running.

costly series of final tests with a certified test lab, often at considerable cost. So, any DSP-based board you produce for sale or use in a product must go through the process. Major circuit or packaging changes may require a retest and recertification. These days, compliance with these standards requires proper planning and design right from the start. Without attention to EMI and EMC effects, and how to reduce them, the project will likely fail certification tests.

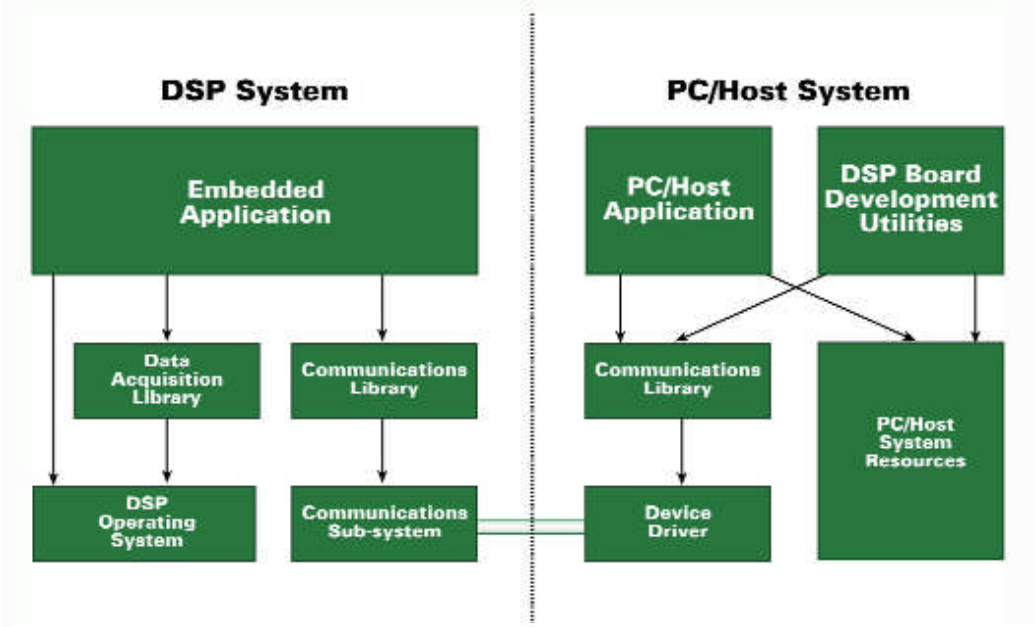
SYSTEM SOFTWARE / FIRMWARE

So far, the discussions above concentrate on hardware issues. Don't short change the budget for software -- the application code that will run on the DSP chip, and the underlying real-time operating system (RTOS) that will manage the DSP chip's resources. If you develop your own board, include time and money to adapt a commercial RTOS or a basic input-output system (BIOS) for your specific implementation. Even if you buy DSP-based boards from an OEM supplier, you may receive only a bare board with a minimum of software. In such a case, you'll have to develop a custom configuration of BIOS and RTOS software.

The next consideration is sending/receiving data to the PC. If you plan to buy a DSP board, ensure the supplier offers a rich set of libraries and code that will let you easily download and test application programs. Your application may also require runtime communications to the other parts of the system or a PC. Suppliers also may offer libraries of code already embedded on their board or within their development tools -- a real time saver.

APPLICATION SOFTWARE

When it comes to building a DSP board, always keep in mind the requirements of the application -- the code that actually acquires and processes data to provide useful results. Developers can write code starting with the assembly language native to a chosen DSP chip, or they can use a higher level language such as C or C++. A third choice, commercial application environments, lets developers choose operations from menus or drag-and-drop lists of functions.



Fully implemented system software architectures simplify application development and reduce support costs over the life of the product.

The choice of programming environment often depends on who will use the application, what support it requires, and what programming resources are available. The proper selection also hinges on whether developers plan to code algorithms and tasks from “scratch” or purchase off the shelf application libraries or complete software packages. In many cases, the need to get a product to market quickly will force a decision. Writing in C or C++, for example, gives developers complete control over processor operations, but that flexibility comes at a cost -- time to understand the DSP chip’s architecture and the time to thoroughly test and debug the final application code. On the other hand, using application environments pulls developers away from the intricacies of the chip’s functions, but this alternative may offer ready-built “core” functions such as filtering, Fourier transforms, and others, that provide the foundation of many signal-processing tasks.

Developers will find that support for application environments outstrips that for programming languages such as C and C++. It’s easier for a supplier’s support people to help a developer understand how to use a pre-built fast Fourier-transform (FFT) algorithm than to help debug home-built FFT code tailored for a specific application.

Also, application environments let developers unfamiliar with the internal operations of signal-processing algorithms, quickly set up and test an application. Using C or C++ would force them to understand the logical flow or signal processing algorithms. Developers may need to get involved at that level to optimize software for a specific task, but many more developers find it easy to use libraries of functions supplied by application environments and by the DSP chip manufacturers.

USER NEEDS

Application environments should let developers quickly set up and test user interfaces, whether this step involves designing a complete GUI front end, establishing a communication protocol with a host computer or embedding an application that doesn't require host PC communications. Developers should not get mired in application program interface (API) calls or obscure routines to produce graphical images or to transmit data.

DEVELOPMENT CYCLE

Always budget time to test and debug an application. Code developed using C or C++ may require debugging and testing tools that monitor individual variables and DSP-chip operations. In many cases application environments automatically perform operations, such as type checking, and memory allocations, so time gets spent on testing how an application works, not on watching how underlying code works. Always consider time to market and time to profit when you evaluate application-development environments.

COST

Always evaluate cost in the overall context of other variables, such as time and staff resources. Although a DSP-chip manufacturer or support group may offer low-cost or even free compilers and development tools, consider the resources needed to use them effectively. A free C++ compiler may sound like a good deal, but its idiosyncrasies may present a steep learning curve. In addition, such a "free" tool usually lacks regular, timely support. An application development environment with a higher initial cost may offer advantages of getting developers started quickly, simplicity of coding,

available libraries of working functions, and a professional, dedicated support staff.

SUPPORT

Even if you develop a good signal-processor board, it will need continuing support, starting on the production line and going on to end users. You must develop test routines that a board manufacturer or your in-house production people can use to test boards. Boards you ship should include self-test and diagnostic software, all of which takes time and energy to prepare. In addition, you must support customers who have questions about your system. Customer's needs change, so you must stay up to date with new DSP technologies, new ADC devices, and new sensor requirements.

CONCLUSION

Adding signal-processing capabilities to a system takes considerable planning, time and energy as well as a commitment to long-term support. You may find it easier to establish a partnership with a company that supplies proven designs for DSP-based systems. Such a partnership removes much of the risk inherent in adopting a new technology.

REFERENCES

"Delta Sigma A/D Conversion Technique Overview," Application Note 10 (AN10), Crystal Semiconductor Corp., Austin, TX January 1997. www.crystal com.

"Demystifying Sigma-Delta ADCs," Application Note 1870, Maxim Integrated Products, Sunnyvale, CA January 2003. www.maxim-ic.com.

"An Introduction to Delta Sigma Converters," Uwe Beis, May 2003 www.beis.de/Elektronik/DeltaSigma/DeltaSigma.html.

"Are ADCs Greek to You?" Tom Lecklider, Evaluation Engineering, May 2003, pp. 12 - 18.