# *Frequently Asked Questions About eXpressDSP*

**Question: What is eXpressDSP™ [1]?**

**Answer:** eXpressDSP Real-Time Software Technology is a premier, open software environment for Texas Instruments (TI™) digital signal processors (DSPs). Its four ingredients are: powerful, integrated development tools; real-time software foundation; standards for application interoperability; and the world's largest DSP third-party network. It is designed to reduce development time well above 50% depending on the application and to give DSP end-users "make vs. buy" choices for DSP algorithms, or simply to make it easier to mix and match algorithms they have developed internally. This will allow programmers to invest their time focusing on the next winning application instead of reinventing the wheel.

**Question: What are the benefits of eXpressDSP?**

**Answer:** It is an industry breakthrough for DSP programmers, enabling them to fully tap the power of these chips and discover new opportunities through a simpler development process that slashes development time and enables choices for reusable, modular software.

**Question: What is the DSP Algorithm Standard?**

**Answer:** The DSP Algorithm Standard is one of the key ingredients of eXpressDSP. It is a set of coding conventions for algorithm writers that ultimately eliminates much of the repetitive and time-consuming system integration associated with algorithms in the past. It achieves this by defining common programming rules and guidelines plus a set of programming interfaces that make the use of algorithms more consistent across a similar set of applications.

**Question: Why implement the DSP Algorithm Standard?**

**Answer:** During the integration process, algorithms are often provided in the form of object code and not source code. As such, it is often difficult or impossible for the system integrator to know what assumptions the algorithm has made about the underlying software foundation in the system in which it is to be deployed. Since simplification of the integration process is key, the eXpressDSP algorithm standard reduces many of these unknowns so that the system integrator can be much more confident about the way an algorithm will operate within his system and shorten his time to solution. The result is that third parties can deliver algorithms that are more easily integrated by the original equipment manufacturers (OEMs), increasing the total algorithm market. Likewise, OEMs enjoy an increase in the quantity and quality of algorithms available. Software developers who previously were consumed by re-engineering the same algorithm for each system application can now turn their attention to inventing new algorithms and applications.

1. eXpressDSP, TI, and Code Composer Studio are trademarks of Texas Instruments Incorporated.

**Question:** **What is the code and performance overhead required to support the DSP Algorithm Standard?**

**Answer:** We have found that cycle count and program and data memory increased less than 1%.

**Question:** **What is TI providing?**

**Answer:** Texas Instruments is providing the eXpressDSP software technology environment, while third parties and OEMs are creating algorithms that conform to the standard and plug-ins that extend the capability of Code Composer Studio™.

**Question:** **What is available today?**

**Answer:** All four ingredients of eXpressDSP are available today. These are delivered via a complete eXpressDSP Algorithm Standard Developers Kit, which includes Rules and Guidelines, "How to" application notes, tools, and a TMS320C6000 demonstration showing algorithm interoperability. For more details, see: http://www.ti.com/sc/expressdsp .

**Question:** **How do customers get expressDSP?**

**Answer:** Log on to the TI eXpressDSP Website at http://www.ti.com/sc/expressdsp.

**Question:** **What DSP platforms are supported by eXpressDSP?**

**Answer:** TMS320C6000 and C5000 platforms are supported by eXpressDSP.

**Question:** **What is unique about eXpressDSP from other software environments?**

**Answer:** eXpressDSP is an initiative that is tightly married to our leadership programmable TMS320 DSP families. It tackles the complex challenges of real-time programming that requires predictable, guaranteed responses to the human environment in a way that let developers of all levels creatively tap the flexibility and power of the DSP.

**Question:** **Do I need Code Composer Studio (CCS) to use the DSP Algorithm Standard?**

**Answer:** The DSP Algorithm Standard is independent of Code Composer Studio; however, the availability of tools such as XDASGen, and the ability to run example code and the Demo makes CCS a very productive environment for using the DSP Algorithm Standard.

**Question:** **How can I measure the overhead of the DSP Algorithm Standard?**

**Answer:** The profiling tool in CCS provides CPU cycle counts for the code between any two profile points. Comparing counts before and after implementing the DSP Algorithm Standard interface for an existing algorithm will allow you to measure the overhead. The application report "Using the eXpressDSP Algorithm Standard in a Static DSP System" (literature number SPRA577) discusses the overhead of the DSP Algorithm Standard, which in most cases, will be well under 1%.

**Question:** **How much effort is needed to convert my algorithm to conform to the DSP Algorithm Standard?**

**Answer:** If the existing algorithm already follows standard programming practices, then the additional work of adding the DSP Algorithm Standard interface takes about a week the first time through, and about a day when you are familiar with the procedure.

**Question:** **Do I have to implement everything in the Standard?**

**Answer:** There are required and recommended parts in the DSP Algorithm Standard. Not all of the interface functions are required.

**Question:** **What are algorithm instance objects?**

**Answer:** The DSP Algorithm Standard requires that algorithms be implemented with an object-like structure and interface. This means that each algorithm operates within a fully characterized, location-independent, encapsulated memory state. The algorithm functions can only be called using its well-defined interface and must be re-entrant.

**Question:** **Does a program need to be object-oriented to be complaint with the DSP Algorithm Standard?**

**Answer:** How you write the algorithm and the language used is not part of the DSP Algorithm Standard specification. A program is compliant with the DSP Algorithm Standard as long as the algorithm module exports a DSP Algorithm Standard interface, and obeys the rules and guidelines in its operation.

**Question:** **How is memory-allocation done for algorithm instance objects?**

**Answer:** All run-time memory allocation must be done by the application code. When the algorithm instance object is created, the application can call the IALG function, `algAlloc()` to obtain the memory requirements and do the allocation. The actual allocation can be done either dynamically at run time, or statically at design time.

**Question:** **How much extra run-time overhead is incurred due to calls to the DSP Algorithm Standard-compliant algorithm functions?**

**Answer:** The only extra overhead is a single function table indirection for each service provider interface function call.

**Question:** **How much extra program and data memory overhead is incurred by using the DSP Algorithm Standard-compliant algorithms?**

**Answer:** The data overhead attributed exclusively to the DSP Algorithm Standard for each used algorithm is the vector table and module ID reference. The vector table contains eight required IALG function pointers, plus pointers to the interface functions. For each algorithm instance object, the only additional overhead is a pointer to the mentioned vector table, the algorithm object handle.