White Paper
SPRA581B

# The TMS320 DSP Algorithm Standard

*Steve Blonstein* *Technical Director*

## Abstract

The TMS320 DSP Algorithm Standard is part of TI's eXpressDSP™ (XDAIS) initiative. The purpose of the standard is to reduce those factors that prohibit an algorithm to be easily integrated into a system without significant reengineering by the system integrator.

Many of the unknowns in such a model relate to resource allocation and consumption on a DSP. Bugs often occur during system integration as a result of the algorithm designer's unfounded assumptions about the system into which the algorithm is to be integrated. The standard, therefore, focuses on a set of general rules and guidelines that should be applied to all algorithm IP. In addition, all algorithms must comply with a generic resource management API, called IALG. Finally, specific rules and guidelines are provided for each family of TI DSPs.

Algorithms provide one or more algorithm standard demonstrations that contain several algorithms which conform to the standard. Additionally, the demonstration will document examples of specific application programming interfaces (APIs), based on the standard, for each algorithm in the demonstration.

### Contents

# Background

Texas Instruments has long understood the importance of good host tools that typically include a C compiler, linker, and debugger. Since improvements in each of these tools translate to an improvement in productivity for the development team, TI invests significant time to continuously improve these tools. Our 1998 acquisition of GO-DSP and the 1999 release of Code Composer Studio™ (CCStudio) further illustrate TI's commitment to push the tool environment forward.

The same cannot be said for the software on the target DSP. In the past, TI did not focus on code reusability on the target. Until recently, DSP applications typically revolved around proprietary technologies instead of standards. The result is the splintered nature of mass market DSP whereby the entire target code implementation is left to the customer who, despite the progress of C compilers, often writes much of the code in assembly to achieve the performance required to justify selecting a DSP.

Today, however, many DSP applications are based on established standards. This has created a market opportunity for TI third parties to develop, market, and distribute commercial off-the-shelf (COTS) algorithms. Since an application often requires the use of several such COTS algorithms, it is feasible to conceive of DSP software frameworks that can provide the infrastructure that enables multiple algorithms to operate on a single platform. Depending upon the provider, the framework may or may not include an operating system kernel, resource management capabilities, or other infrastructure. As the number of these frameworks increase along with the number of algorithms, it is difficult to take COTS algorithms and integrate them into a framework without a significant amount of effort. This integration issue impacts not only system integrators and the framework providers, but also the thousands of customers who look at these systems. TI believes that minimizing integration time will benefit everyone in the development process.

This white paper discusses a new TI standard called the TMS320 DSP Algorithm Standard. The premise for creating the standard was to directly address the issues that cause problems in system integration, and in particular the issues revolving around the use of existing algorithm IP that is supplied from another party.

The algorithm standard is the result of a tremendous support effort from many TI third parties and several key customers. We wish to thank them for their dedicated efforts to make this standard a reality.

# Terminology

This white paper is the result of many meetings and consultations within TI and with third parties and customers. During these meetings, it became obvious that there was confusion over standard terminology. Below is a sampling of definitions for some of the more common terms that we encounter when discussing the new standard.

- **Algorithm**: A module of code that consumes a data stream, processes it, and outputs a resultant stream. Examples include vocoders, modems, audio compression, video decompression, etc.

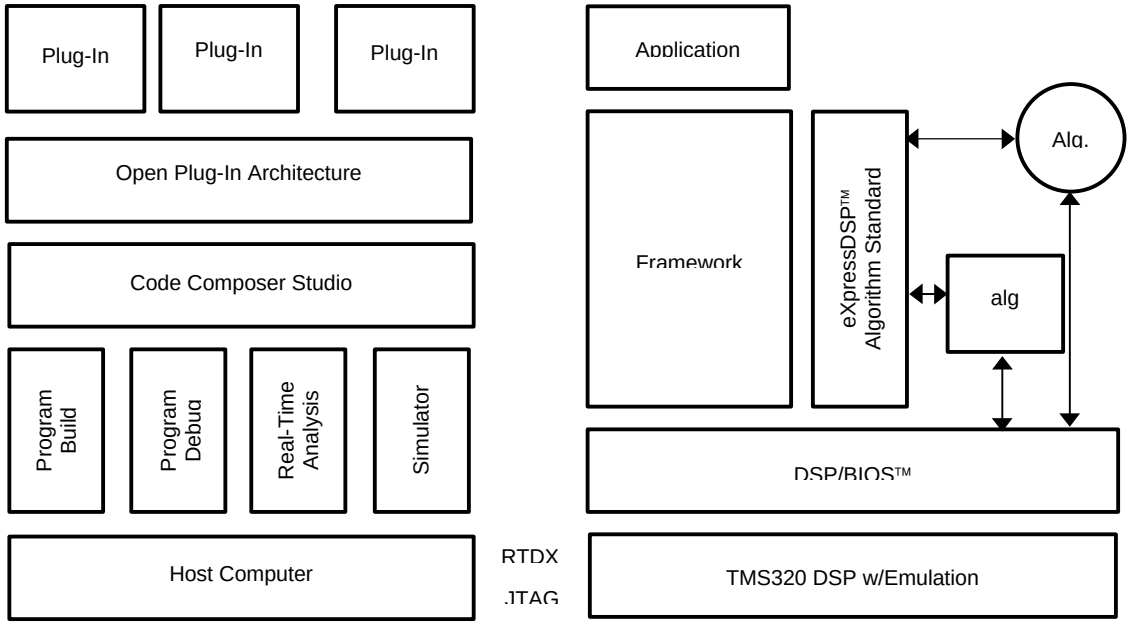
- **Framework**: The "glue" that holds together the application, the algorithms, and the underlying infrastructure or operating system. Depending upon the instantiation of the framework, it may control resources such as memory usage, processor usage, I/O issues, and be responsible for passing data to, from, and between algorithms. Some frameworks contain their own OS, or tasking kernel, tailored to the application. This has led to the concept that such a framework be considered an application-specific operating system (ASOS).

- **Operating System**: The underlying system software that manages system resources, schedules activities, time and timers, handles interrupts, resolves resource conflicts, handles I/O, etc. SPOX™ is a good example of a DSP operating system. Note that SPOX is application-agnostic, i.e., it is designed to be usable in any application environment. As a result, frameworks can leverage SPOX functionality or attempt to perform some of the functions themselves in a fashion optimized to the particular application. Other more sophisticated (and larger) operating systems might include network stacks, file systems, and multiprocessor support.

- **DSP/BIOS™**: Software that underlies the operating system, or possibly displaces it. DSP/BIOS is not supposed to be an operating system per se. While it does offer rudimentary threading, interrupt support, pipes, and signals, the primary purpose of DSP/BIOS is the logging and statistical accumulation features that enable real time analysis of the system. Again, DSP/BIOS is application-agnostic.

- **Application**: The definition depends upon the use of some or all of the other components. If a customer writes all the code from scratch including a kernel, algorithms, and a framework, then the entire software system may be described as the application. However, in an environment where a commercial OS, framework, and algorithms have been deployed, the application programmer may see no further down into the system than the APIs to the controlling framework.

*Figure 1. Diagrammatic view of eXpressDSP™ elements on a TI 320Cxxxx DSP*




# Description of the eXpressDSP Elements

The left-hand side of the diagram shows the new CCStudio environment. It is not the intent of this white paper to discuss this area of technology. More detail on this can be found at www.ti.com/sc/ccstudio.

The right hand side of the diagram represents the target DSP in the eXpressDSP environment. It assumes the use of the DSP/BIOS real-time analysis modules with the possible inclusion of threads and tasks as required by the system designer. It also assumes that the system is built upon a COTS framework (the rectangle labeled framework). The key to this part of the diagram is the vertical, narrow rectangle that "separates" the framework from the algorithms. It is this "separation" that defines the basis for the algorithm standard.

The emphasis on the word "separation" is important. The standard focuses on the interfaces between the algorithm and the rest of the system, rather than the ability of algorithm writers to exploit their individual talents to achieve their goals of fastest, smallest, and cheapest.

Essentially, the core of the standard focuses on an abstraction of DSP resource management away from the algorithms themselves. Typically, resources on a DSP refer to memory usage and placement, along with I/O control such as the use of DMA channels, and possibly the use of key control registers.

When only a single algorithm runs on the DSP, one can make broad assumptions about the use of the DSP resources. Even in the case of a multiple-channel instantiation of that algorithm, provided basic re-entrancy exists, then the system should run just fine. However, when several algorithms are combined, problems occur (e.g., when one algorithm assumes the use of certain resources that are then "stolen" or “borrowed” by another algorithm in real time). It can be very difficult to pinpoint the source of a problem if the first algorithm was never designed to run in such a way and then doesn’t perform to its’ specification, or worse, performs sporadically.

In addition, algorithms cannot make calls to an underlying operating system for typical OS services. If this were true, then that algorithm could not be used in any other environment where the same OS were not present. However, the standard does allow limited use of the DSP/BIOS. Only the real-time analysis modules may be called to provide for visualization of the algorithm in real time. None of the threading or tasking APIs may be called.

With these basic premises, the algorithm standard enables a system integrator to more easily assemble production-quality systems from one or more algorithms.

# Who Stands to Gain With This Algorithm Standard?

With the basic premise for the standard being reduction in system integration time, everyone in the development cycle stands to win. However, there are other less obvious benefits to the parties involved.

## The Algorithm Writer

Today, many system integrators have their own algorithm integration methodologies and the algorithm writer must adhere to that methodology. This requires the algorithm vendor to support multiple interfaces. By getting everyone to agree to one standard, this issue is resolved.

A second major advantage to supporting the algorithm standard is that the system integrator can more quickly monitor the performance of an algorithm. If the algorithm vendor believes that he or she has a better solution than the one currently in use, the system integrator can quickly make an exchange to test out the new algorithm. This is very difficult if each algorithm follows a different interface standard.

## Customers and System Integrators

Many hard-to-find bugs simply go away because algorithm "black boxes" become a lot more predictable.

The system integrator now has more choices. With several standardized algorithms available, it becomes much easier to compare algorithm A to algorithm B in order to gauge performance, robustness, and size.

Individual algorithm interface methodologies adopted by system integrators can be eliminated.

# Scope of TMS320 DSP Algorithm Standard

The algorithm standard is comprised of several rules that, to be fully compliant, must be adhered to. Additional guidelines are also recommended. Many of the rules (approximately 25) are common-sense programming practices that have been in use for a long time and apply to all algorithms, regardless of the application area.

In addition to the base rules, there are Instruction Set Architecture (ISA) specific rules (e.g., use of specific control registers) for each of the major TI320 families of DSPs.

These general rules and guidelines, along with the ISA specific rules and guidelines, are contained in the *TMS320 DSP Algorithm Standard Rules and Guidelines*, literature number, SPRU352.

A second document*, TMS320 DSP Algorithm Standard API Reference*, literature number, SPRU360, contains information on the implementation of the one generic API that all algorithms must follow. Referred to as “IALG,” this API is primarily responsible for taking the memory management function away from the algorithm and placing it in the hosting framework. Thus, a negotiation occurs between the algorithm and framework to assign memory for that algorithm. Additional functions within this API allow such functions as shared memory blocks between algorithms and for the framework to move memory around while an algorithm is operating in the system.

A second optional API in this document is called ITRC. This is designed to standardize the algorithm's test mode.

Finally, to assure the practical use of an algorithm, it must be supplied with a specific API, appropriate to the function being performed. The algorithm standard does not require the use of any particular set of APIs.

However, to increase acceptance of the standard, TI will be distributing one or more algorithm standard demonstration applications that use several algorithms that are based on the standard. In addition, TI will provide documentation that describes specific APIs, based on the standard, for each of the included algorithms.

By using standard algorithms, a customer can “switch out” one of the algorithms and "link-in" another version of the same algorithm without having to recompile, although a re-link will be necessary. This can only be done when object code compatibility is ensured, and this can only be achieved if the second algorithm also uses the same specific API.

*Note:* An algorithm vendor may want to provide a version of the algorithm that follows the specified example APIs so that it can be evaluated in one of these algorithm standard demonstrations.

## New versus Retrofit?

There are clearly hundreds and perhaps thousands of algorithm implementations that exist for use on TI DSPs. The question often arises as to whether it makes sense to go back and retrofit an algorithm to be compliant with this new standard. Clearly, for new algorithms, it is prudent to follow the rules and guidelines right away. The answer to the retrofit question should be handled on a case by case basis. Issues to address include whether or not the algorithm will be used in different applications in the future. Will the algorithm be sold and integrated into a larger application? Will a particular customer demand that all algorithms be compliant with the new standard? If the answer to these or similar questions is yes, then it probably is worth looking at converting the algorithms. To assist with this effort, TI will provide a selection of tools that will make this task easier. If the answer to this type of question is no, you must ask yourself if altering a perfectly good piece of working IP simply for the sake of being compliant is feasible.

TI will provide tools and wizards that assist with some of the boiler plate code that is required for all algorithms, regardless of the application area.

## Summary

The TMS320 DSP Algorithm Standard is a bold initiative being undertaken by the world leader in DSP. TI has clearly recognized that as system software complexity grows at an exponential pace, this kind of standard initiative is vital if we are to maintain the time-to-market goals of our customers. While we recognize that many system integrators have such internal methodologies already, it is advantageous for everyone in our sphere to share and promote a single standard.

TI Contact Numbers

INTERNET

*TI Semiconductor Home Page*
www.ti.com/sc

*TI Distributors*
www.ti.com/sc/docs/distmenu.htm

PRODUCT INFORMATION CENTERS

*Americas*
Phone +1(972) 644-5580
Fax +1(972) 480-7800
Email sc-infomaster@ti.com

*Europe, Middle East, and Africa*
Phone
- Deutsch +49-(0) 8161 80 3311
- English +44-(0) 1604 66 3399
- Español +34-(0) 90 23 54 0 28
- Francais +33-(0) 1-30 70 11 64
- Italiano +33-(0) 1-30 70 11 67

Fax +44-(0) 1604 66 33 34
Email epic@ti.com

*Japan*
Phone
- International +81-3-3344-5311
- Domestic 0120-81-0026

Fax
- International +81-3-3344-5317
- Domestic 0120-81-0036

Email pic-japan@ti.com

*Asia*
Phone
- International +886-2-23786800
- Domestic
  - Australia 1-800-881-011
    - TI Number -800-800-1450
  - China 10810
    - TI Number -800-800-1450
  - Hong Kong 800-96-1111
    - TI Number -800-800-1450
  - India 000-117
    - TI Number -800-800-1450
  - Indonesia 001-801-10
    - TI Number -800-800-1450
  - Korea 080-551-2804
  - Malaysia 1-800-800-011
    - TI Number -800-800-1450
  - New Zealand 000-911
    - TI Number -800-800-1450
  - Philippines 105-11
    - TI Number -800-800-1450
  - Singapore 800-0111-111
    - TI Number -800-800-1450
  - Taiwan 080-006800
  - Thailand 0019-991-1111
    - TI Number -800-800-1450
  - Fax 886-2-2378-6808
  - Email tiasia@ti.com

## IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.