Ctai Revista de Automação e Tecnologia da Informação, v. 2, n. 2, p. 125-129, 2003
PRELIMINAR

Meta-Framework Proposto pela Texas Instruments para o Desenvolvimento de Sistemas Embutidos usando DSP

Rômulo S. de Oliveira, Daniel J. Pagano, Marcos V. Linhares

Departamento de Automação e Sistemas - DAS
Universidade Federal de Santa Catarina – UFSC, Florianópolis/SC
romulo@das.ufsc.br, marcos@das.ufsc.br

Resumo: **Este artigo descreve os elementos para o desenvolvimento eficiente de sistemas embutidos com processadores digitais de sinais (*DSP - Digital Sinal Processors*). Tomou-se como base um *meta-framework* elaborado pela Texas Instruments (TI) devido a alguns fatores que se fizeram determinantes: é o principal fabricante mundial de *DSP*; a disponibilidade de kits e ferramentas de desenvolvimento além da demanda existente no mercado catarinense atual onde várias empresas estão utilizando seus processadores e suas ferramentas.**

Palavras-chave: **DSP, eXpressDSP, Sistemas Embutidos, Texas Instruments.**

1 INTRODUÇÃO

OS benefícios de um software estruturado em módulos é bem conhecido, especialmente, em sistemas complexos onde vários módulos contribuem individualmente na formação de todo o sistema. Os esforços de desenvolvimento destes são realizados somente uma vez pois, eles podem ser reutilizados em projetos futuros. Para o engenheiro de controle esses módulos normalmente são conhecidos como blocos funcionais e devem possuir um comportamento previsível e serem compatíveis com outros blocos tanto no formato dos dados como funcionalmente, integrando o que os engenheiros chamam de diagrama de blocos (sistema). Para possuir estas características os algoritmos devem seguir um padrão, um conjunto de regras a serem utilizadas por todos os desenvolvedores.

As indústrias têm aprendido os benefícios desta metodologia e têm visto os impactos que esta pode causar sobre a integração de um sistema: tempo de depuração reduzido (pois os módulos já foram depurados exaustivamente, antes de serem liberados); rápida reconfiguração do sistema (os módulos são facilmente substituíveis por versões mais atualizadas ou retirados se estiverem causando algum problema à aplicação); além de prover uma visualização de maior nível do software, encapsulando detalhes de implementação dos módulos.

A TI (*www.ti.com*) com o objetivo de aumentar a performance no desenvolvimento de sistemas utilizando *DSP* criou um *meta-framework* para suportar a metodologia de desenvolvimento em módulos e o denominou *eXpressDSP*. O *eXpressDSP* é composto, basicamente, por elementos que têm como alvo o *DSP* e elementos que têm como alvo o PC. Entre os elementos que têm como alvo o *DSP* encontram-se o *TMS320 DSP Algorithm Standard*, o *kernel DSP/BIOS* e um *framework* de referência. Já entre as elementos que têm como alvo o PC estão o *Code Composer Studio* e o emulador *JTAG*. Na Figura 1 pode-se visualizar a divisão das diversas partes em seus respectivos alvos.

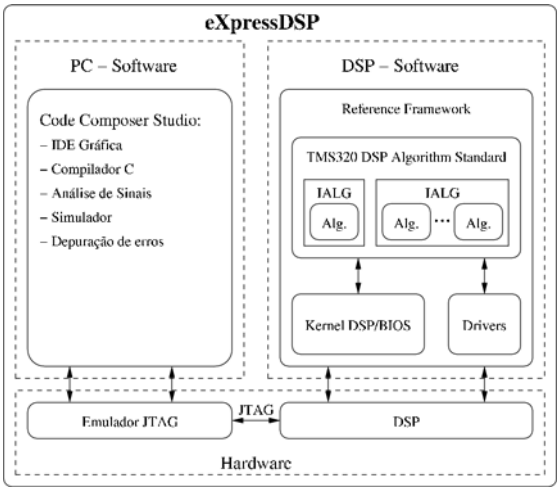

Figura 1: Elementos do *eXpressDSP*

a) O *TMS320 DSP Algorithm Standard* (também conhecido como *XDAIS*) - especifica um conjunto de regras gerais e linhas guia (*guidelines*) a serem aplicadas ao desenvolvimento de algoritmos para *DSP*. A proposta desta padronização é reduzir aqueles fatores que proíbem um algoritmo de ser facilmente integrado em um sistema, sem uma significante reengenharia de *software*;
b) *Kernel DSP/BIOS* - camada de *software* de "baixo nível" que provê abstração de *hardware* e gerencia os recursos físicos. Também provê suporte a interrupções, *threads* e outras funções;
c) *Reference Framework* (*framework de referência*) - provê ao desenvolvedor um

ponto de partida para o desenvolvimento de aplicações com *DSP* unindo em sua estrutura os *drivers*, os algoritmos e o *kernel* DSP/BIOS;

d) *Code Composer Studio* - ambiente integrado de desenvolvimento (IDE) que une as ferramentas de *software* necessárias para se trabalhar com *DSP*, sendo exemplos delas: emulação, monitoração e controle em tempo de execução; configuração visual de periféricos e módulos do DSP/BIOS; compilação, linquedição e *download* do software para o DSP; e outros;

e) Emulador *JTAG* (*Joint Test Action Group*) - *hardware* externo ligado à porta paralela ou ao barramento do PC e que permite emulação, por *software*, em tempo real, do *DSP* (ou *DSP*), através do padrão IEEE 1149.1 que especifica instruções de varredura e teste e o *hardware* necessário para efetuá-lo.

Este *meta-framework*, proposto pela Texas Instruments, provê uma sólida integração do software com o hardware de desenvolvimento. Os benefícios de ser obediente à ele é o acesso à uma extensa rede de parceiros desenvolvedores e de contribuições de propriedade intelectual, isto é, uma grande quantidade de algoritmos compatíveis disponíveis para reuso.

2 TMS320 DSP ALGORITHM STANDARD[1][2]

Os Processadores Digitais de Sinais (também denominados *DSP*) são muitas vezes programados como os microprocessadores 'tradicionais', onde predomina uma mistura de linguagens (C e Assembly) por razões como: performance, acesso direto a periféricos e registradores, etc. Na maioria das vezes é usado um pequeno ou mesmo nenhum sistema operacional. Assim como para os tradicionais microprocessadores, existe um uso muito pequeno, comercialmente (*comercial-off-the-shelf* - COTS), de componentes de software para *DSP*.

Entretanto, diferentemente dos microprocessadores de propósito geral os *DSP* são projetados para executar algoritmos sofisticados de processamento de sinais. Por exemplo, podem ser utilizados para reconhecimento de fala em um automóvel barulhento viajando a 100 km/h.

Tais algoritmos muitas vezes são o resultado de muitos anos de pesquisa. No entanto, por causa da falta de padrões consistentes é quase impossível reusar estes algoritmos em mais de um sistema sem uma significante reengenharia. Como poucas empresas podem dispor de um time de doutores em *DSP* e o reuso de algoritmos é muito trabalhoso, o tempo de lançamento no mercado de um novo produto baseado em *DSP* é medido em meses e muitas vezes, dependendo da complexidade, em anos.

O *XDAIS* especifica regras que devem ser seguidas para que um algoritmo obedeça ao *eXpressDSP*. Já as *guidelines*, entretanto, são expressamente recomendadas mas não exigidas para que o algoritmo seja obediente ao padrão.

A seguir são apresentadas as exigências do *TMS320 DSP Algorithm Standard*. Estas devem ser utilizadas durante todo o processo de desenvolvimento, justificando escolhas de projeto e ajudando também a esclarecer a intenção de muitas regras e *guidelines*. As exigências do padrão são:

a) Algoritmos de vários fornecedores podem ser integrados em um sistema único. É impossível para um desenvolvedor de produtos que quer participar destes mercados, que possuem um grande conjunto de algoritmos, obter todos eles de uma única fonte.

b) Os algoritmos são livres de contexto, isto é, o mesmo algoritmo pode ser eficientemente utilizado em, virtualmente, qualquer aplicação ou *framework*.

c) Os algoritmos podem ser utilizados em ambientes de execução puramente estáticos bem como dinâmicos.

d) Os algoritmos podem ser distribuídos em forma binária. É importante que os algoritmos sejam entregues em forma binária, isto não somente protege a propriedade intelectual do distribuidor do algoritmo como também melhora a reusabilidade.

e) A integração dos algoritmos não exige recompilação da aplicação do cliente, embora, seja necessário reconfiguração e religação (*relinking*).

Embora nem sempre seja possível atingir os objetivos perfeitamente, eles representam as principais preocupações que devem ser consideradas na definição dos elementos exigidos anteriormente. São eles:

a) Facilidade para aderir ao padrão;

b) Possibilidade de verificar conformidade ao padrão;

c) Permitir aos integradores de sistemas uma fácil migração entre *DSP* da Texas Instruments;

d) Permitir ferramentas para simplificar tarefas do integrador do sistema, incluindo configuração, modelagem da performance, conformação ao padrão e depuração de erros;

e) Ficar sujeito a pouco ou nenhum *overhead* causado pela inserção de indireções para abstração de *hardware*.

2.3 Divisão em níveis

O *TMS320 DSP Algorithm Standard* define regras e *guidelines* (linhas guia) em três dos quatro níveis ilustrados na Figura 2 e descritos a seguir:

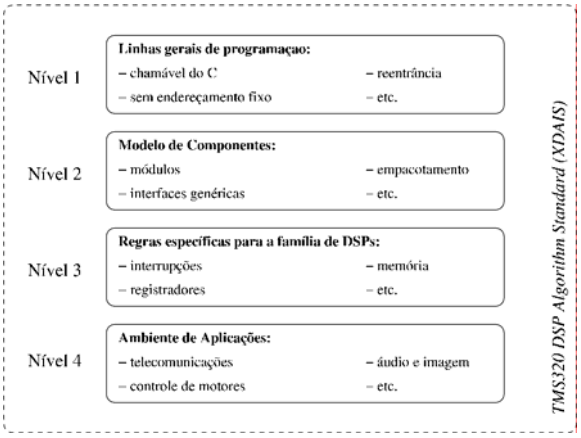

Figura 2: Elementos do *Algorithm Standard* [2]

a) Nível 1 - Contém as regras e linhas gerais de programação que são aplicadas a todos os algoritmos e arquiteturas de *DSP* não importando a área de aplicação. Quase todos os módulos de *software* desenvolvidos recentemente seguem, de acordo comum, estas regras e este nível apenas formaliza isso.
b) Nível 2 - Estabelece regras e *guidelines* que permitem a todos os algoritmos trabalharem harmoniosamente em um sistemas simples. Convenções são estabelecidas para o uso da memória de dados e nomes para identificadores externos pelos algoritmos, por exemplo. Adicionalmente, regras simples de como os algoritmos são 'empacotados' são também especificadas.
c) Nível 3 - Contém as *guidelines* para famílias específicas de DSP. Atualmente não existe um acordo quanto ao uso de algoritmos com respeito ao uso dos recursos do processador. Estas *guidelines* irão prover uma direção sobre o que fazer ou não nas várias arquiteturas.

3 KERNEL DSP/BIOS [3][4]

Tradicionalmente as aplicações para *DSP* eram muito simples, tipicamente utilizando um único programa para executar o processamento necessário. Ao longo do tempo, com o aumento da complexidade, as aplicações começaram a ter a necessidade de processamento concorrente e atualmente têm se tornado crítico que o *DSP* faça diversas tarefas simultaneamente. Além disso, as aplicações também evoluem (melhoramentos nos algoritmos de controle, implementação de mais funções como comunicação e outras) o que exige suporte para adicionar ou modificar suas características. Construir estas modernas aplicações usando o tradicional paradigma do laço único é, não só desafiador, como também torna muito difícil a manutenção.

O *kernel DSP/BIOS* é um pequeno *firmware*, proposto pela Texas Instruments, que executa em um processador digital de sinais (*DSP*) e que provê componentes de software de forma a permitir, aos desenvolvedores de aplicações, a capacidade de:

a) Monitoração e controle em tempo real - monitorar e controlar a execução e as variáveis do programa em tempo de execução;
b) Escalonamento tempo real - escalonamento e comunicação em sistemas *multi-thread* em tempo de execução.

Em sistemas simples o *software* de sistema consiste de uma inicialização básica de *hardware*, funções de acesso a periféricos e rotinas de tratamento de interrupção (*ISR - Interrupt Service Routine*). Já os sistemas mais complexos exigem um escalonamento para garantir a correta operação das diversas funções (algoritmo de controle, comunicação via rede, etc.) dependendo da prioridade de cada um. Além disso, as aplicações muitas vezes exigem acesso concorrente aos recursos de *hardware* (memória, E/S, etc.).

Em resumo, o DSP/BIOS é um produto da *Texas Instruments Inc.* que compreende um núcleo de tempo real projetado para aplicações que requerem escalonamento tempo real e sincronização, comunicação e instrumentação. Fornece um ambiente *multi-thread* preemptivo, abstração de hardware e ferramentas de configuração, provendo uma coleção de serviços que os desenvolvedores usam para gerenciar os recursos em nível de sistema e construir a infra-estrutura para as aplicações com *DSP*. Esses recursos são ajustados e otimizados de acordo com necessidades com tamanho e/ou performance, o DSP/BIOS, atualmente, está disponível para os *DSP* das famílias TMS320C5000 e TMS320C6000.

O DSP/BIOS provê uma ferramenta gráfica de configuração, integrada ao *Code Composer Studio*, que permite um acesso simples a configuração necessária de uma aplicação. Este processo de configuração estático reduz a memória final a ser utilizada pelo *DSP*, otimizando e eliminando funções do núcleo não utilizadas pela aplicação.

3.1 Módulos e Serviços

Para incrementar o suporte para análises e configuração de periféricos o DSP/BIOS inclui em seu núcleo os seguintes serviços:

a) Interrupções de *hardware* - interface entre as interrupções de hardware e o núcleo do DSP/BIOS;
b) Interrupções de *software* - *threads* 'leves' que usam a pilha de programa e não sofrem preempção;
c) *Tasks* - *threads* independentes da execução principal (*background*) que podem ceder o processador;
d) Funções periódicas - *threads* 'leves' disparadas em tempos determinados;

e) *Mailboxes* - sicronizam a troca de dados entre *threads*;
f) Semáforos;
g) Filas - listas ligadas atômicas;
h) *Clock* - interface dos *timers* de hardware;
i) *Streams* para comunicação entre *threads* e dispositivos de E/S;
j) Gerenciador de memória - baixo *overhead* na alocação dinâmica de memória.

Para prover a resposta rápida exigida por uma aplicação com *DSP* , o DSP/BIOS aumenta o tradicional modelo de tarefas com mecanismos adicionais. As interrupções de *software* são *threads* 'leves' que compartilham uma pilha comum. Isto resulta em um baixo *overhead* da memória e trocas de contexto mais rápidas já que não é necessário salvar e restaurar a pilha de tarefas. Funções periódicas são disparadas como *threads* de alta prioridade que podem facilmente processar os dados amostrados em intervalos fixos de tempo, simplificando o projeto de sistemas que possuem várias taxas de amostragem (*multirate*). Para facilitar o projeto de aplicações sofisticadas, o DSP/BIOS provê serviços de comunicação entre *threads* , incluindo semáforos, *mailboxes* e filas.

O *DSP/BIOS* provê módulos que permitem aos projetistas de sistemas realizar tarefas essenciais e em tempo de execução tais como: monitoração e controle, e escalonamento e comunicação. Tais módulos podem ser visualizados na Tabela 1 e descritos a seguir.

Tabela 1: Módulos principais do *Kernel DSP/BIOS*

Módulos para configuração do sistema	
GBL	*Global setting manager*
MEM	*Memory manager*
Módulos para controle e monitoração tempo real	
LOG	*Message log manager*
STS	*Statistic accumulator manager*
TRC	*Trace manager*
Módulos para escalonamento tempo real	
HWI	*Hardware interrupt manager*
SWI	*Software interrupt manager*
IDL	*Idle function and processing loop manager*
CLK	*System clock manager*
PRD	*Periodic function manager*
Módulos para comunicação tempo real	
PIP	*Data pipe manager*
HST	*Host input/output manager*
RTDX	*Real-time data exchange manager*

a) Módulos para configuração do sistema - define o hardware e o ambiente do sistema;
b) Módulos para controle e monitoração tempo real - provê um meio de enviar informações para o hospedeiro (host - o PC, por exemplo) enquanto o programa executa no DSP ;
c) Módulos para escalonamento tempo real - tem por função manusear uma thread particular em um ambiente tempo real. Estas rotinas podem ser algoritmos de controle implementados pelo desenvolvedor do sistema, por exemplo;
d) Módulos para comunicação tempo real - responsável por gerenciar os canais de comunicação entre threads e entre o DSP e o hospedeiro (PC, por exemplo).

Utilizando os módulos do DSP/BIOS os projetistas de sistemas podem desenvolver algoritmos de controle usando diagramas de blocos, onde cada bloco representa uma *thread* , similar ao que é encontrado no projeto de *hardware* utilizando circuitos integrados dedicados como blocos.

4 *FRAMEWORK* DE REFERÊNCIA[5]

Com a introdução das ferramentas e metodologias vistas (DSP/BIOS, XDAIS e Code Composer Studio), a Texas Instruments resolveu direcioná-las para o que chamou de '*Reference Framework*'.

Os *Reference Frameworks* provêm um ponto de partida para aplicações que utilizam o *DSP/BIOS* e o *TMS320 Algorithm Standard*. Os desenvolvedores primeiramente selecionam o *Reference Framework* que melhor se aproxima do seu sistema e de suas necessidades futuras e então adaptam ele populando-o com algoritmos obedientes ao padrão. Os elementos comuns como *drivers* de dispositivos, gerenciador de memória e outros já estão pré-configurados nestes *frameworks*. Os desenvolvedores podem centralizar a atenção unicamente nas necessidades da aplicação aumentando consideravelmente a produtividade.

4.1 Definindo as características dos *Reference Frameworks*

Algumas características são necessárias para se determinar uma arquitetura comum, apropriada para a maioria dos sistemas de um determinado *framework*. É importante notar que neste ponto as decisões arquiteturais são independentes dos detalhes das aplicações finais. Sendo assim é bom considerar as seguintes características:

a) Quantos algoritmos são utilizados no sistema?
b) O sistema exige criação dinâmica de objetos e alocação de recursos ou uma configuração estática é suficiente?
c) O sistema irá executar em uma taxa de freqüência ou em várias taxas?
d) Existem restrições quanto a quantidade de memória?
e) O sistema necessita de controle externo?

As respostas a essas perguntas esclarecem como o sistema será construído, compondo um *framework* de base, isto é, um *Reference Framework* (RF).

De acordo com as respostas obtidas a partir das questões anteriores pode-se dividir os RFs em diferentes níveis de complexidade, conforme pode ser visto na Tabela 2. A diferença fundamental entre cada RF são as quantidades de módulos do DSP/BIOS presentes, o que diminui as funcionalidades e o suporte do sistema mas também mantém o código total bastante otimizado.

Tabela 2: *Reference Frameworks* separados por nível [5]

Parâmetro de projeto	RF1	RF3	RF5
Configuração estática	✓	✓	✓
Criação dinâmica de objetos	-	-	-
Gerenciamento de memória	✓	✓	✓
Alocação dinâmica	-	✓	✓
Número de canais	1 a 3	1 a 10	1 a 100
Número de algoritmos	1 a 3	1 a 10	1 a 100
Memória absoluta mínima	✓	-	-
Única taxa de operação	✓	✓	✓
Múltiplas taxas de operação	-	✓	✓
Threads bloqueantes	-	-	✓
Controle externo	-	✓	✓

a) RF1 (*Framework* Compacto) - Este *framework* é projetado com o mínimo necessário. Utiliza configuração estática e não suporta criação dinâmica de objetos. O gerenciamento de memória é completamente estático e não suporta gerenciamento dinâmico. Não há preempção nem bloqueio de *threads* também não provê módulos do DSP/BIOS para controle ou comunicação.
b) RF3 (*Framework* Flexível) - Difere, significativamente, do nível anterior. O mínimo necessário é substituido pela flexibilidade. Suporta criação estática de objetos mas a criação dinâmica pode ser adicionada. Os buffers de dados podem ser configurados e gerenciados em tempo de execução. Suporta também múltiplas taxas de operação, isto quer dizer que diferentes algoritmos podem executar em diferentes freqüências (um a 10ms outro a 20ms, por exemplo). Provê também uma *thread* adicional que permite controle externo do *DSP*.
c) RF5 - (*Framework* Extendido) - Este *framework* é concebido para projetistas que procuram uma flexibilidade extendida sem se preocupar com restrições do processador. Suporta criação estática e dinâmica de objetos e também gerenciamento de memória estático e dinâmico. Suporta uma grande quantidade de algoritmos e permite múltiplas taxas de operação. Normalmente este nível requer mais de 70% dos módulos do *DSP/BIOS* para que o sistema suporte todas as funcionalidades requeridas por este RF.

4.2 Elementos de um *Reference Framework*

A Figura 3 ilustra os elementos que compõe o *Reference Framework*, tendo como alvo o *DSP*.

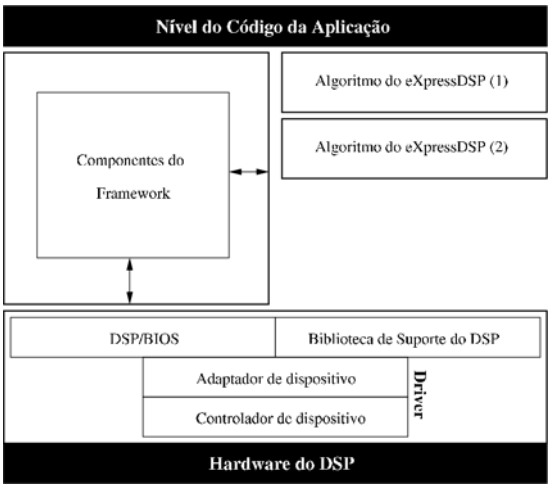

Figura 3: Elementos do *Reference Framework* [5]

5 CONCLUSÃO

Este artigo descreveu o *meta-framework* proposto pela TI com o intuito facilitar o desenvolvimento de sistemas embutidos utilizando processadores digitais de sinais (*DSP*). Pôde-se ter uma idéia da gama de técnicas que são utilizadas para o desenvolvimento eficiente de sistemas, desde metodologias que podem ser adotadas no projeto e codificação até as características importantes que um *kernel* de tempo real traz para aplicações multitarefas.

Pretendeu-se com este artigo mostrar a base da qual uma empresa partiria para o desenvolvimento do seu *framework* e também fazer um levantamento do que existe para não incorrer no típico erro de 'reinventar a roda'.

A informação contida neste artigo está espalhada por dezenas de documentos da TI, os quais nem sempre são consistentes entre si. Procurou-se portanto, definir um conjunto consistente de termos, embora isto signifique uma certa liberdade em relação à alguns documentos da TI, especialmente os mais antigos.

REFERÊNCIAS BIBLOGRÁFICAS

[1] S. Blonstein. *The TMS320 DSP Algorithm Standard*. Technical Document - White Paper. Texas Instruments, May 2002b. revision C.
[2] Texas Instruments, editor. *TMS320 DSP Algorithm Standard - Rules and Guidelines*. Technical Document - User Guide. Texas Instruments, October 2002. revision E.
[3] A. Thé, D. W. Dart, and S. Dirksen. *How to get started with the DSP/BIOS Kernel*. Technical

Document - Application Report. Texas Instruments, SFS/SDS, August 2001.

[4] H. Yiu. *Understanding Basic DSP/BIOS Features*. Technical Document - Application Report. Texas Instruments, China, April 2000.

[5] S. Blonstein. *Reference Frameworks for eXpressDSP Software: A White Paper*. Technical Document - Application Report. Texas Instruments, SDS, December 2002a. revision A.