VI Induscon - Conferência Internacional de Aplicações Industriais
Recife - PE, 09-12 de abril de 2006

Temporal Behavior Assessment of Automatically Generated DSP Code

Marcos V. Linhares, Rômulo S. de Oliveira, Ricardo B. Borges, Alexandre J. da Silva
Departamento de Automação e Sistemas, UFSC. Florianópolis - SC - Brasil
Emails: [marcos, romulo, rbb]@das.ufsc.br, alexandrejs@weg.com.br

***Abstract*— The aim of this work is to analyse the temporal behavior of the multitask system generated by the Simulink/MatLab. In order to achieve that goal, the following efforts were made: the specification of a multitask application; the code generation for this system; and the timing measuring and modeling to evaluate the behavior.**

Keywords - DSP, multitask, real-time, Simulink/MatLab, temporal behavior.

I. Introdução

A incorporação de sistemas eletrônicos em uma grande variedade de produtos, como automóveis, eletrodomésticos e outros, têm causado um extraordinário crescimento da indústria eletrônica. Produtos tradicionais ficaram mais eficientes, com melhor qualidade e mais baratos com a introdução desses sistemas.

Diversos componentes eletrônicos colaboraram para este grande crescimento da indústria e melhoraram significantemente a qualidade dos produtos finais. Entretanto, o projeto destes sistemas tornou-se mais complexo pois, agora, envolvem diversos elementos heterogêneos (componentes analógicos e digitais).

Existe uma tendência dos sinais analógicos serem processados como sinais digitais, isso devido a grande facilidade de integração entre os componentes digitais e a grande flexibilidade destes, pois podem envolver computação e controle em seu interior. Dentre estes podemos destacar, já que são objeto de nosso estudo, os microprocessadores e microcontroladores.

Quando se envolve componentes digitais que permitem algum tipo de computação em seu interior temos um sistema de computação, que pode ser classificado em [1]:

- Sistemas de computação de propósito geral – desta classe fazem parte os computadores tradicionais, caracterizados pela possibilidade do usuário final poder programar os sistemas.
- Sistemas de computação e controle dedicados – desta classe fazem parte os computadores de propósito específico, caracterizados por serem desenvolvidos para uma determinada aplicação onde o usuário final tem acesso limitado à programação do sistema.

Atualmente, desde simples máquinas domésticas até completas instalações podem ser controladas através do uso de sistemas de computação. Quando esses sistemas estão firmemente fixados aos equipamentos ao ponto de que, se forem retirados, o equipamento deixar de funcionar, então estes sistemas são chamados de sistemas embutidos (*embedded systems*) [2][3]. Normalmente, estes sistemas possuem restrições temporais na execução das tarefas para as quais foram projetados e programados. Quando isto acontece, tem-se um sistema embutido de tempo real. Os sistemas de tempo real são sistemas cujo funcionamento correto depende dos resultados produzidos e do instante no qual esses resultados são produzidos [4][5][6]. Exemplos de aplicações que requerem tempo real incluem: controle de uma planta química, controle de processos complexos de produção, aplicações automotivas e aeronáuticas, automação industrial e robótica, sistemas militares e muitos outros.

Muitos dos atuais sistemas embutidos de tempo real são projetados utilizando técnicas e abordagens empíricas, através de conhecimentos adquiridos por tentativa e erro. Diversas aplicações de controle com restrições de tempo são implementadas pela codificação de grandes programas em linguagem *assembly*, programando *timers*, manipulando em baixo nível dispositivos periféricos, tarefas e prioridades de interrupção. Apesar do código produzido por estas técnicas ser otimizado para uma execução eficiente, esta abordagem tem algumas desvantagens [7]:

- A implementação de grandes partes de código em linguagem *assembly* requer maior esforço que a programação em linguagens de alto nível;
- Poucas pessoas conseguem entender o funcionamento completo do *software* produzido;
- As modificações de grande parte do programa torna-se muito difícil, até mesmo para o programador original, devido à complexidade envolvida;
- Sem o suporte específico de ferramentas e metodologias as análises para verificação de restrições temporais são praticamente impossíveis.

A maior conseqüência deste abordagem é que o *software* de controle produzido por técnicas empíricas pode ser temporalmente imprevisível. Se as restrições de tempo não forem verificadas anteriormente e o sistema operacional, quando existente, não incluir características para manipular tarefas de tempo real, o sistema pode aparentemente funcionar corretamente por um período de tempo mas pode entrar em colapso em situações específicas. As conseqüências da falha do sistema dependem da criticidade da aplicação.

Para o projeto e implementação de um sistema de controle de tempo real embutido utiliza-se de diversos tipos de profis-

sionais que são especialistas cada um em sua área e que usam suas próprias ferramentas. Um sistema simples envolveria um engenheiro de *hardware* para projetar o *hardware* a ser utilizado, um engenheiro de controle para projetar o controlador e definir as restrições temporais e um engenheiro de *software* para implementar o controlador no *hardware* construído.

A interação entre o engenheiro de controle e o de *software* deve ser a mais eficaz possível já que um está projetando o que o outro irá implementar. E para tanto é de extrema valia que ambos troquem informações utilizando uma mesma linguagem. A ferramenta de projeto do engenheiro de controle são os diagramas de blocos funcionais onde estão contidos os modelos matemáticos da planta onde se está atuando e do controlador que se está projetando. De forma a facilitar a comunicação entre eles pode-se mapear os diagramas de blocos em diagramas de componentes de *software* onde o engenheiro de *software* pode retirar as informações para implementar o controlador.

As pressões do mercado exigem equipamentos não só robustos mas que ofereçam maiores funcionalidades a seus clientes. Isto representa a necessidade de se embutir, em um determinado dispositivo, um maior número de funcionalidades. O modelo monotarefa, composto por uma única tarefa que faz tudo (por exemplo, interface humano-máquina, laço de controle e comunicação em rede), começa a ser insuficiente para manter as restrições temporais no sistema final. Aqui o sistema multitarefa se mostra ideal. Mas, se tratanto de equipamentos a serem controlados, é obrigatório que as tarefas respeitem as restrições temporais e para tanto é necessário um núcleo operacional de tempo real para gerenciar as execução das tarefas e garantir o correto funcionamento temporal do sistema.

Com o objetivo de integrar o engenheiro de *software* com o engenheiro de controle, muitas ferramentas de desenvolvimento de controladores, que já utilizavam o diagramas de blocos, têm possibilitado a geração de código de forma automática através do simples clique de um botão. Isso não só aumentou a produtividade no desenvolvimento de sistemas controladores como também inseriu uma camada de abstração na qual o engenheiro deve se preocupar somente com o funcionamento correto do modelo simulado, escondendo as características específicas, de baixo nível, da plataforma para a qual ele está desenvolvendo, sem contudo retirar dispositivos periféricos essenciais para o funcionamento da aplicação desenvolvida.

A MathWorks [8], fabricante do Simulink/MatLab, possui uma ferramenta que gera códigos em linguagem C correspondente aos sistemas modelados em diagramas de blocos no Simulink, o RTW (*Real Time Workshop*). Como forma de direcionar este código para os principais fabricantes de plataformas de controle embutidas, inserindo características específicas de cada uma, como a Motorola e a Texas Instruments, a MathWorks introduziu o *Embedded Target*. O Simulink permite o projeto de um sistema multitarefa e, de forma complementar, gera um pequeno núcleo operacional tempo real baseado em um algoritmo escalonador do tipo taxa monotônica (*rate monotonic*).

O objetivo deste trabalho é avaliar o comportamento temporal do sistema multitarefa gerado pelo Simulink/MatLab juntamente com seu núcleo operacional tempo real. Isso possibilitará ao engenheiro que estiver desenvolvendo sua aplicação um maior controle do comportamento das tarefas, identificando problemas que são causados pela execução de um sistema multitarefa, como preempções, perda de *deadlines* e outros. Para tanto foram feitos os seguintes esforços:

- especificação de uma aplicação multitarefa usando o Simulink/MatLab;
- geração de código e implementação do sistema em um DSP previamente escolhido;
- monitoração de sinais, verificação e avaliação do comportamento temporal da aplicação.

II. Aplicação

Primeiramente é necessário especificar qual plataforma será utilizada como alvo. O *hardware* escolhido para a avaliação foi o LF2407 eZdsp (*Spectrum Digital*) que tem como processador o TMS320C2407A, um DSP da Texas Instruments (TI), destinado ao controle de motores e inversores, com entradas digitais e analógicas (via ADC), saídas PWM e digitais, além de interfaces de comunicação serial e CAN.

Escolhido o *hardware* alvo, foi especificada uma aplicação composta por cinco tarefas executando em uma unidade de processamento, para avaliar o comportamento temporal do código gerado pela ferramenta. As tarefas definidas foram as seguintes:

- Malha de controle de velocidade de um motor CC com controlador PI;
- Recebimento da referência da velocidade via rede CAN;
- Envio de dados amostrados (velocidade do motor, referência e saída do controlador);
- Interface humano-máquina (IHM), composta de um display e um teclado.

Para configurar o Simulink para a aplicação proposta, fez-se o seguinte: escolheu-se a plataforma alvo, pelo *Embedded Target* do Simulink/MatLab, para o LF2407 eZdsp. Configurou-se o *TimerClockPrescaler* para 128, que é o valor pelo qual o módulo do *timer* divide a fonte de *clock*, que no caso é de 40MHz, portanto, a frequência do *timer* que gera a interrupção é 312,5KHz; e o *FundamentalStepSize* (tempo fundamental) para 0.001 segundos, que é o valor da ocorrência da interrupção do tempo (1ms). Montou-se então o modelo das tarefas que seriam executadas, configurando-se seus períodos (*sample time* dos blocos) com os valores desejados para os períodos das tarefas, estes devem ser múltiplo do tempo fundamental (já que o modelo executa em tempo discreto). Feito isto gerou-se o código de forma automatizada pela própria ferramenta.

A figura 1 ilustra as tarefas especificadas no Simulink/MatLab. Cada um dos blocos contém os periféricos (CAN, ADC, entradas e saídas digitais, PWM e outros) necessários para o correto funcionamento de cada tarefa, assim como a

plataforma alvo escolhida. Na figura 2 pode-se ver a tarefa do controlador em detalhes. Percebe-se os elementos de *hardware* presentes (PWM, ADC, memória) bem como elementos de *software* (*buffer*, conversor de dados) além, é claro, do próprio controlador (PI). Esses elementos foram retirados diretamente da paleta de ferramentas do próprio Simulink/MatLab (sendo necessário uma mínima configuração).

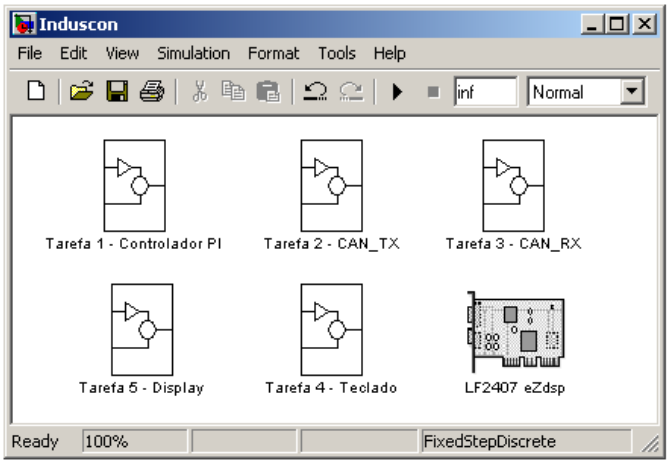

Fig. 1. Aplicação, especificada no Simulin/MatLab

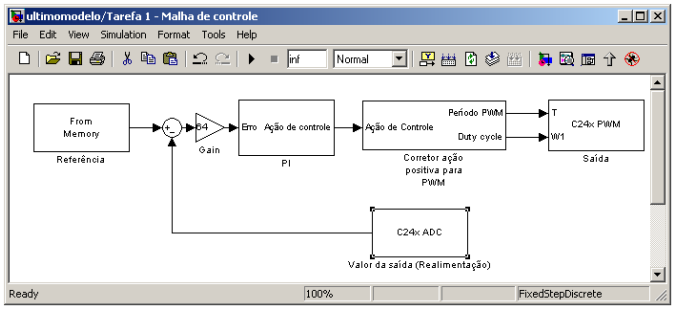

Fig. 2. Tarefa do Controlador

As outras tarefas seguem a mesma forma da tarefa de controle e estão ilustradas nas figuras 3 e 4, as tarefas da IHM (teclado e display) foram simuladas devido a não ter-se o *hardware* pronto para executá-las.

Não foi preocupação avaliar o código gerado pela ferramenta (complexidade, otimização e outros), mas sim o comportamento temporal da aplicação, isto é, o funcionamento das tarefas e núcleo operacional tempo real gerado por ela, o qual influencia diretamente na temporalidade da aplicação.

III. Comportamento Temporal

O Simulink/MatLab gera não só o código da aplicação mas também um pequeno núcleo operacional tempo real que é adicionado ao sistema desenvolvido. O núcleo é disparado por uma interrupção do tempo (*timer*) que é configurado dentro da ferramenta (*FundamentalStepSize*). Esta base de tempo é assumida para todas as tarefas da aplicação (tempo fundamental), isto é, as tarefas devem possuir tempos que

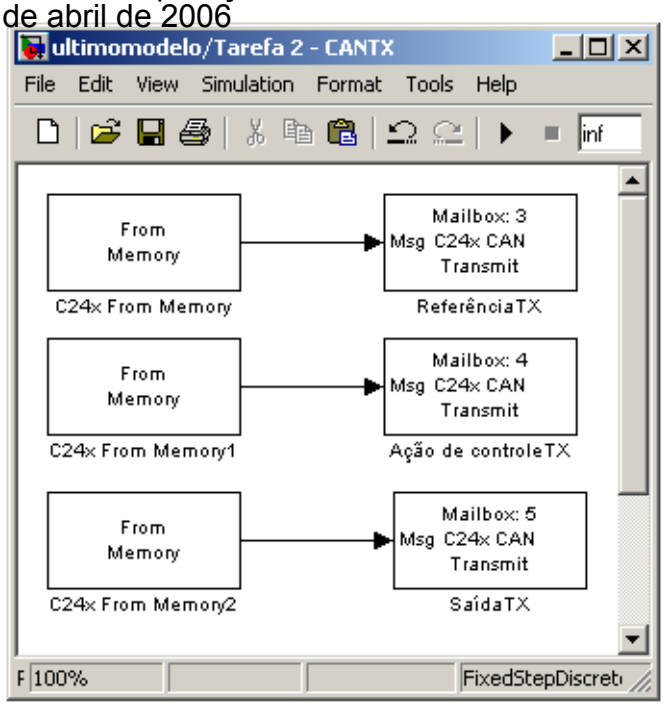

Fig. 3. Tarefa de Envio de dados via CAN

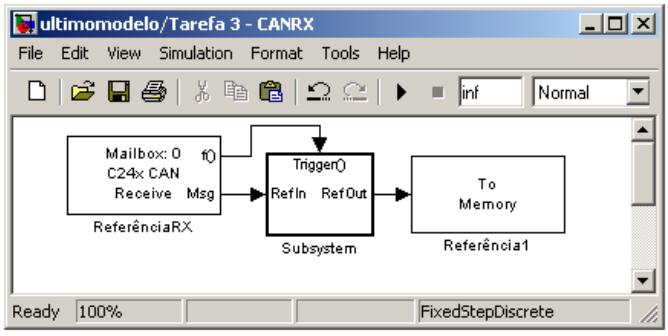

Fig. 4. Tarefa de Recebimento de dados via CAN

sejam múltiplos dessa base, obrigatoriamente. Tarefas com períodos (*sample time*) iguais são unidas em uma só quando o código é gerado. Qualquer coisa anormal (blocos de uma mesma tarefa com períodos diferentes, por exemplo) irá ser acusada como erro pela própria ferramenta.

O núcleo operacional gerado automaticamente pela ferramenta é responsável por gerenciar as diversas tarefas e garantir que estas executem nos períodos previamente configurados. O diagrama de atividades do núcleo operacional pode ser visto na figura 5.

Este pequeno núcleo operacional usa um algoritmo de escalonamento baseado no taxa monotônica (*rate monotonic*). O taxa monotônica é uma algoritmo que implementa uma regra simples que atribui prioridades de acordo com os períodos das tarefas. Especificamente, tarefas com períodos pequenos têm prioridades maiores. Já que os períodos são constantes o taxa monotônica é um algoritmo de prioridade fixa, ou

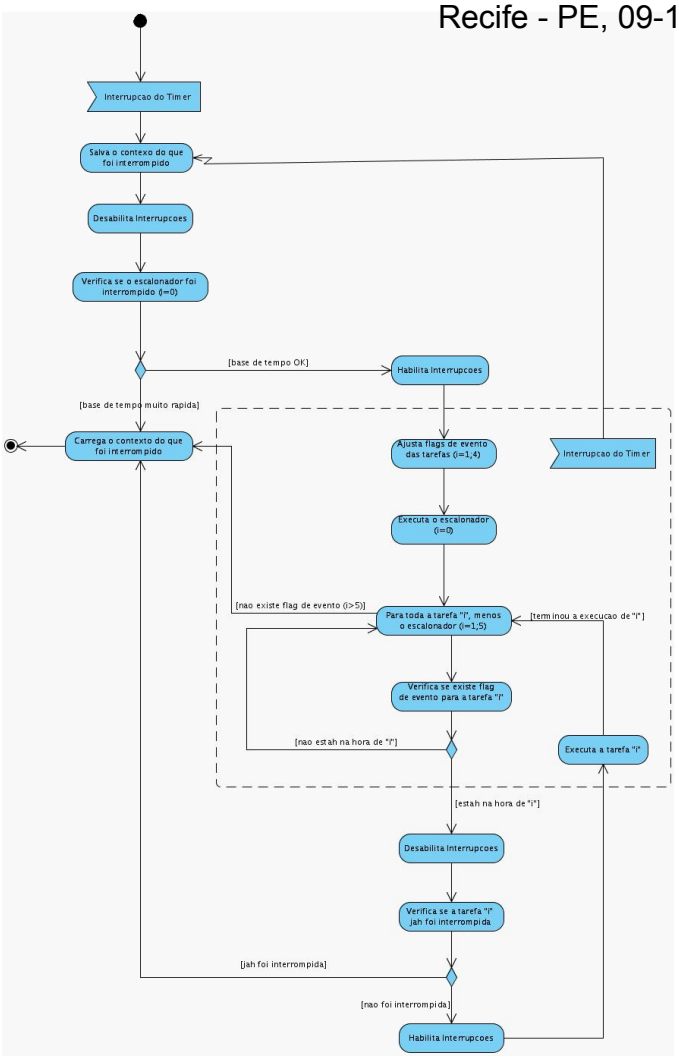

Fig. 5. Diagrama de Atividades do núcleo gerado pelo Simulin/MatLab

seja, as prioridades são atribuídas às tarefas antes da execução (em tempo de projeto) e não mudam no passar do tempo. Além disso, o taxa monotônica é intrinsecamente preemptivo, isto é, a tarefa que está executando é sempre preemptada (interrompida) por uma tarefa nova que tenha menor período (maior prioridade) [7].

Analisando o diagrama de atividades pode-se perceber que o núcleo é bastante simples e gerado sob medida para o conjunto de tarefas especificado na ferramenta. O próprio núcleo operacional é tratado como uma tarefa (i=0, de maior prioridade) e executada sempre na interrupção do tempo (tempo fundamental). As tarefas seguintes, no nosso caso cinco tarefas (i=1;5), são executadas nos seus períodos, pré-configurados, e devem ser múltiplos do tempo base.

Como forma de evitar que uma tarefa não execute nunca (*deadlock* ou postergação indefinida) o núcleo operacional implementa um laço que verifica, tarefa por tarefa, se cada uma está ou não no seu período. Por se tratar de um laço fixo, que depende do número de tarefas, sua interferência no tempo de resposta aumenta a medida que a tarefa tem menor prioridade (maior período).

Para prevenir que a tarefa inicie a sua execução e nunca acabe (*starvation*) o núcleo operacional implementa uma *flag* (*over run*), ou seja, se uma tarefa que já iniciou sua execução e não a completou at vencer novamente o seu período esta executará com as interrupções desabilitadas. Isso garante a execução de todas as tarefas mas do ponto de vista de tempo real pode-se ocasionar a perda de *deadlines* de tarefas de maior prioridade em detrimento da execução de tarefas de menor prioridade (inversão de prioridade).

De forma a contribuir para o desenvolvimento de sistemas utilizando o Simulink/MatLab, foram levantados dados relevantes sobre o comportamento temporal do sistema desenvolvido. Podemos considerar, para efeitos de cálculo do comportamento temporal da aplicação, os seguintes parâmetros:

- Tempo de computação das tarefas (C_x) - tempo que a tarefa gasta, exclusivamente, na sua execução;
- Tempo de interferência do núcleo operacional (I_k) - é o preço a ser pago pelas facilidades do multitarefa tempo real gerado pela ferramenta.
- Tempo de resposta das tarefas (R_x) - representado pela soma do tempo de computação da tarefa x, do tempo de interferência das tarefas de maior prioridade e dos tempos de interferência do núcleo de suporte até o término da execução da tarefa x;
- Tempo de interferência das tarefas de maior prioridade (I_x) - é a soma dos tempos de computação das tarefas mais prioritárias que a tarefa x durante seu tempo de resposta (R_x).

Indentificou-se que algumas sobrecargas (*overhead*) têm influência sobre C_x (tempo de computação) das tarefas, sendo elas:

1) Descobrir qual tarefa precisa executar, que gasta 14,8 μs de processamento por tarefa;
2) Selecionar a tarefa que precisa executar, que gasta 5,6 μs e depende diretamente da prioridade da tarefa.

Sendo assim pode-se definir um C'_x (tempo de computação real) composto da seguinte forma (considerando que x reflete a prioridade da tarefa):

$$C'_x = C_x + 14,8 + 5,6 \cdot x$$

A interferência causada pelo núcleo operacional (I_k) é representada pelo tempo gasto com o tratamento da interrupção do tempo e suas sobrecargas (*overhead*), sendo costituído por:

- rotinas de tratamento do tempo e execução do escalonador, gasta 85,8 μs;
- laço de escolha da tarefa a ser executada, gasta $n \cdot 5,8$ μs (onde n é a quantidade de tarefas do sistema projetado) quando nenhuma tarefa executa ou 5,8 $\cdot$ x μs para cada tarefa x, onde x reflete a prioridade da tarefa;
- rotinas de salvamento e recuperação de contexto, que gastam, respectivamente, 16 μs e 10 μs.

Devido ao núcleo operacional, gerado pela ferramenta, não executar de forma atômica e homogênea, para todas as

tarefas, dividiu-se sua execução em dois diferentes tempos de computação: C_k, quando a interrupção do tempo ocorre durante a execução de qualquer tarefa x, que reflete a latência máxima do núcleo operacional; e $C'_k(x)$, quando a interrupção do tempo é quem inicia a tarefa x. Sendo assim, I_k pode ser representado por:

$$I_k(x) = C'_k(x) + \left(\left\lceil \frac{R_x}{P_i} \right\rceil - 1 \right) \cdot C_k$$

onde C_k e $C'_k(x)$ são,

$$C_k = 16,0 + 85,8 + n \cdot 5,8 + 10,0$$

$$C'_k(x) = 16,0 + 85,8 + 5,8 \cdot x$$

A figura 6 deixa bem evidente o que foi explicado acima. Percebe-se que a utilização da escolha (*switch*) para selecionar qual tarefa vai ser executada sobrecarrega as tarefas de menor prioridade. Em situações de restrição temporal mais fortes (controle de alta potência, por exemplo) pode ser desejável iniciar a otimização do código por ali. Como foi dito, não foi preocupação otimizar o código gerado, isso com certeza alteraria o comportamento temporal do sistema e um novo modelo deveria ser feito para cálculo dos novos tempos de resposta (R_x) e de computação real (C'_x).

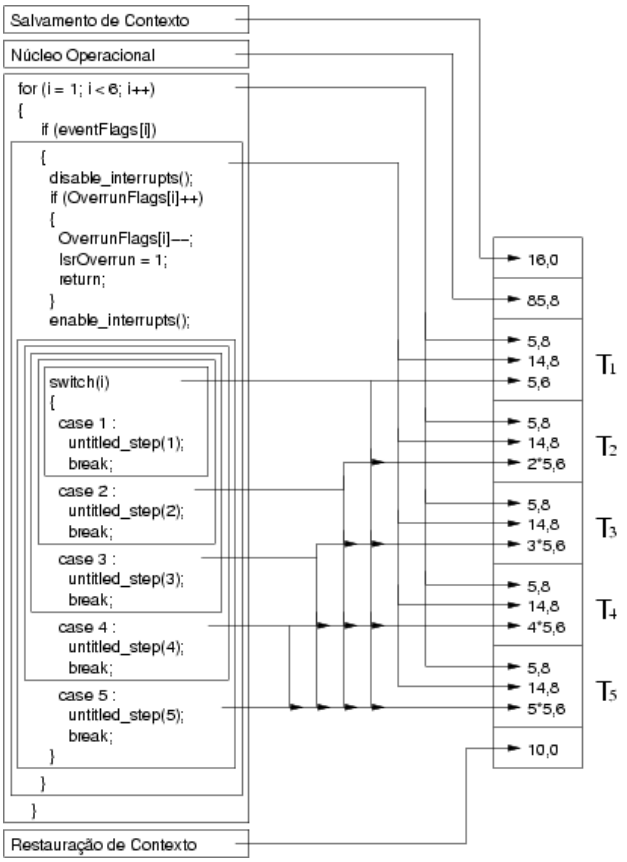

Fig. 6. Código gerado explodido conforme a utilização do processador.

Para o cálculo do tempo de resposta pode-se utilizar:

$$R_x = C'_x + I_k + I_x$$

onde I_x é dado por:

$$I_x = \sum_{i=1}^{x-1} \left\lceil \frac{R_x}{P_i} \right\rceil \cdot C_i$$

Na implementação da MathWorks, o núcleo de suporte tem prioridade máxima no sistema. Porém, se alguma tarefa for interrompida e não conseguir retornar até o início de seu novo período (*over run*) ela ganha o processador, desabilitando todas as interrupções, inclusive do próprio núcleo operacional. Isso pode ocasionar em algum momento a perda de *deadlines* (máximo de tempo no qual uma tarefa deve completar sua execução) de outras tarefas de maior prioridade.

De forma a complementar o estudo é importante verificar também, de forma simples, se haverá ou não a ocorrência de *over run* que, como foi dito acima, desestabilizaria todo o sistema. Portanto, considerando os períodos (prioridades) arbitrados, sabendo-se que o *deadline* da tarefa (D_x) é igual ao seu período e os tempos de resposta calculados, é necessário e suficiente, para um teste de escalonabilidade que:

$$\forall x, R_x \leq P_x \qquad onde, \quad D_x = P_x$$

Portanto, se torna muito importante o cálculo do comportamento temporal em tempo de projeto, para prevenir um comportamento indesejável e inclusive manter o sistema sob controle durante sua execução.

IV. Experiências

Com base na aplicação proposta e modelada no Simulink/MatLab pode-se validar o modelo do comportamento temporal equacionado. Esta seção descreve como foram adquiridos alguns parâmetros a partir dos quais executou-se a solução das equações apresentadas.

A tabela I descreve os períodos de todas as tarefas. O período de interrupção do tempo, ou execução do núcleo operacional, foi definido em 1 ms. Os períodos das tarefas foram atribuídos de forma a satisfazer as necessidades e requisitos da aplicação, mas nada impede que os mesmos sejam alterados para o desenvolvimento de outra aplicação.

x	**Tarefa (T_x)**	**Período (P_x)**
1	**controlador**	2 ms
2	**envio de dados via CAN**	3 ms
3	**recebimento de dados via CAN**	5 ms
4	**teclado (IHM)**	400 ms
5	***display* (IHM)**	500 ms

TABELA I
Período das tarefas especificadas no Simulink/MatLab.

Para realizar as medições dos tempos de computação (C_x) algumas saídas digitais disponíveis no DSP foram conectadas fisicamente às ponteiras de um osciloscópio. O código foi gerado automaticamente pelo RTW do Simulink e o suporte aos

periféricos é dado pelo *Embedded Target for TI C2000 DSP*. Cada tarefa coloca sua saída correspondente em nível alto no início de sua execução e nível baixo no final. Configurando-se, inicialmente, os períodos das tarefas com bastante folga (relativamente grandes) para se ter a garantia que nenhuma tarefa será interrompida por outra durante sua execução, o que iria mascarar o resultado do tempo de computação. A tabela II ilustra os tempos de computação (C_x) amostrados. A tabela apresenta também os tempos de comptução real (C'_x) para o conjunto de tarefas projetado, além do tempo de computação do núcleo operacional quando este inicia a tarefa x (C'_k), calculados segundo as equações apresentadas.

Tarefa (T_x)	$C_x(\mu s)$	$C'_x(\mu s)$	$C'_k(x)(\mu s)$
1	192, 8	213,2	107,6
2	79, 2	105,2	113,4
3	23, 5	55,1	119,2
4	1351	1388,2	125,0
5	9179	9221,8	130,8

TABELA II
TEMPOS DE COMPUTAÇÃO DAS TAREFAS DA APLICAÇÃO.

A latência máxima da interrupção do tempo pode ser considerada como o tempo de computação do núcleo operacional quando este acontece durante a execução de uma tarefa (C_k) que, calculado segundo a equação apresentada, para o conjunto de tarefas proposto, é de 140,8 μs.

Os tempos restantes foram calculados conforme as equações apresentadas, a resolução do tempo de resposta faz-se usando um processo iterativo que, ou diverge ($R_x > P_x$) ou converge em um número finito de passos ($R_x(m+1) = R_x(m) = R_x$). **Em se tratando de um processo iterativo para R_x o valor inicial escolhido define a quantidade de iterações necessárias para o valor convergir, ou seja, quanto mais próximo o valor incial está de convergir menor a quantidade de iterações para R_x. Sendo assim, pode-se considerar que o valor inicial ($R_x(0)$) ideal seja dado pela equação seguinte, visto que o tempo de resposta de uma tarefa de menor prioridade sofrerá influência do tempo de resposta da tarefa anterior e assim por diante.**

$$R_x(0) = C'_x + R_{(x-1)}$$

As equações abaixo ilustram a solução para todo o conjunto de tarefas proposto. Pode-se perceber que nas equações 1, 2 e 3, não há a interferência de C_k isso devido ao tempo de resposta destas ser inferior ao período da interrupção do tempo ($R_x < P_k$), sofrendo somente a interferência de C'_k para iniciar e das tarefas de maior prioridade. Já nas equações 4 e 5 percebe-se o tempo de resposta é maior que a interrupção do tempo ($R_x > P_k$), fazendo com que C_k também participe na interferência da tarefa, aumentando o tempo de resposta desta.

$$\begin{aligned} R_1(0) &= C_1 = 213,2\mu s \\ R_1(1) &= \overbrace{213,2}^{C'_1=C_1+14,8+5,6\cdot 1} + \overbrace{107,6}^{C'_k(1)=16+85,8+5,8\cdot 1} + \\ &\quad + \left(\left\lceil \frac{213,2}{1000} \right\rceil - 1\right) \cdot 140,8 = 320,8\mu s \end{aligned} \quad (1)$$

$$\begin{aligned} R_2(0) &= C_2 + R_1 = 426,0\mu s \\ R_2(1) &= \overbrace{105,2}^{C'_2=C_2+14,8+5,6\cdot 2} + \overbrace{113,2}^{C'_k(2)=16+85,8+5,8\cdot 2} + \\ &\quad + \left(\left\lceil \frac{426,0}{1000} \right\rceil - 1\right) \cdot 140,8 + \\ &\quad + \left\lceil \frac{426,0}{2000} \right\rceil \cdot 213,2 = 431,8\mu s \end{aligned} \quad (2)$$

$$\begin{aligned} R_3(0) &= C_3 + R_2 = 486,9\mu s \\ R_3(1) &= \overbrace{55,1}^{C'_3=C_3+14,8+5,6\cdot 3} + \overbrace{119,2}^{C'_k(3)=16+85,8+5,8\cdot 3} + \\ &\quad + \left(\left\lceil \frac{486,9}{1000} \right\rceil - 1\right) \cdot 140,8 + \\ &\quad + \left\lceil \frac{486,9}{2000} \right\rceil \cdot 213,2 + \\ &\quad + \left\lceil \frac{486,9}{3000} \right\rceil \cdot 105,2 = 492,7\mu s \end{aligned} \quad (3)$$

$$\begin{aligned} R_4(0) &= C_4 + R_3 = 1880,7\mu s \\ R_4(1) &= \overbrace{1388,2}^{C'_4=C_4+14,8+5,6\cdot 4} + \overbrace{125,0}^{C'_k(4)=16+85,8+5,8\cdot 4} + \\ &\quad + \left(\left\lceil \frac{1880,7}{1000} \right\rceil - 1\right) \cdot 140,8 + \\ &\quad + \left\lceil \frac{1880,7}{2000} \right\rceil \cdot 213,2 + \\ &\quad + \left\lceil \frac{1880,7}{3000} \right\rceil \cdot 105,2 + \\ &\quad + \left\lceil \frac{1880,7}{5000} \right\rceil \cdot 55,1 = 2027,5\mu s \\ R_4(2) &= \ldots = 2381,5\mu s \end{aligned} \quad (4)$$

$$
\begin{aligned}
R_5(0) &= C_5 + R_4 = 11603,3\mu s \\
R_5(1) &= \overbrace{9221,8}^{C'_5=C_5+14,8+5,6\cdot 5} + \overbrace{130,8}^{C'_k(5)=16+85,8+5,8\cdot 5} + \\
&\quad + \left(\left\lceil \frac{11603,3}{1000} \right\rceil - 1\right) \cdot 140,8 + \\
&\quad + \left\lceil \frac{11603,3}{2000} \right\rceil \cdot 213,2 + \\
&\quad + \left\lceil \frac{11603,3}{3000} \right\rceil \cdot 105,2 + \\
&\quad + \left\lceil \frac{11603,3}{5000} \right\rceil \cdot 55,1 + \\
&\quad + \left\lceil \frac{11603,3}{400000} \right\rceil \cdot 1388,2 = 14165,9\mu s \\
R_5(2) &= \ldots = 15108,9\mu s \\
R_5(3) &= \ldots = 15410,0\mu s
\end{aligned} \tag{5}
$$

A tabela III ilustra o comportamento temporal da aplicação desenvolvida e implementada na plataforma alvo selecionada. Tanto o tempo de resposta (R_x) como o tempo de interferência da(s) tarefa(s) de maior prioridade (I_x) e o tempo de interferência do núcleo operacional (I_k), foram calculados conforme as equações apresentadas.

Tarefa (T_x)	$R_x(\mu s)$	$C_x(\mu s)$	$I_x(\mu s)$	$I_k(\mu s)$
1	$320,8$	$213,2$	–	$107,6$
2	$431,8$	$105,2$	$213,2$	$113,4$
3	$492,7$	$55,1$	$318,4$	$119,2$
4	$2381,5$	$1388,2$	$586,7$	$406,6$
5	$15410,0$	$9221,8$	$3945,4$	$2242,8$

TABELA III
COMPORTAMENTO TEMPORAL DAS TAREFAS DA APLICAÇÃO.

No gráfico de Gantt da figura 7 percebe-se como as tarefas de menor prioridade (R_5 e R_4) sofrem com a interferência das tarefas de maior prioridade. Isto ocorre devido às tarefas de maior prioridade poderem preemptar as tarefas de menor prioridade interferindo em suas execuções. Esta abordagem garante a execução das tarefas mais importantes, mas pode tornar o tempo de resposta da tarefa de menor prioridade tão grande que poderia vencer o seu período, dependendo das restrições temporais da aplicação. Como já foi dito é importante estar atento a isto já que isto causaria, em particular na aplicação da MathWorks, uma inversão de prioridades podendo desestabilizar todo o sistema.

Observando a tabela IV pode-se ver que o modelo proposto reflete o comportamento temporal da aplicação para o pior caso. Medindo diretamente o tempo de resposta das tarefas no dispositivo e calculando pelo modelo (amostrado/calculado) pode-se detectar um erro máximo de, aproximadamente (no caso da aplicação desenvolvida), $3,5\%$ **para o pior caso, chegando alguns resultados ao erro mínimo de** $0,35\%$**. O erro é percebido, principalmente, nas tarefas com tempos de resposta menores que o período do tempo, já nas tarefas maiores esse erro é diluído no período da tarefa se tornando, por consequência, menos expressivo. O erro considerado foi o de limite superior (*upper bound*), mantendo assim uma margem de segurança para o projetista do sistema.**

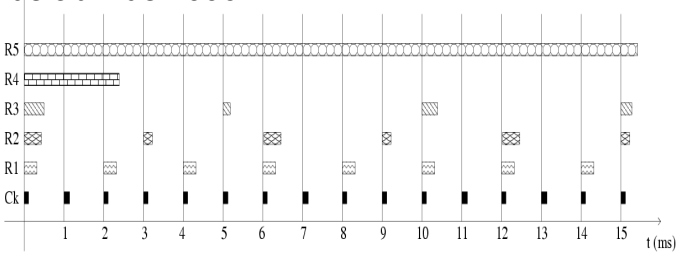

Fig. 7. Gráfico de Gantt do comportamento temporal da aplicação

Tarefa (T_x)	$R_xCalculado(\mu s)$	$R_xAmostrado(\mu s)$	$Erro(\%)$
1	$320,8$	$315,0$	$1,8$
2	$431,8$	$421,5$	$2,4$
3	$492,7$	$476,0$	$3,5$
4	$2381,5$	$2353,0$	$1,2$
5	$15410,0$	$15356,0$	$0,35$

TABELA IV
ERRO NOS TEMPOS DE RESPOSTA (CALCULADO VERSUS AMOSTRADO).

V. CONSIDERAÇÕES FINAIS

Este artigo consistiu da avaliação do comportamento temporal de uma aplicação projetada utilizando a ferramenta da MathWorks, o Simulink/MatLab. O objetivo foi avaliar este de forma que, ao utilizar a ferramenta para projetar uma aplicação, possa se entender como o comportamento temporal pode influenciar na aplicação ou sistema final. Isso tende a melhorar o processo de construção de sistemas com controladores embutidos.

Espera-se com este trabalho contribuir para a melhoria dos processos de desenvolvimento de *software* para sistemas embutidos no contexto do controle e automação. Especialmente aqueles de pequeno porte como controladores de motores elétricos e inversores.

REFERENCES

[1] C. A. Valderrama, M. E. D. Lima, S. Cavalcante, and E. Barros, *Hardware / Software Co-design: Projetando Hardware e Software Concorrentemente*. São Paulo: Escola de Computação, IME-USP, 2000.
[2] A. S. Berger, *Embedded Systems Design: An Introduction to Process, Tools and Techniques*. CMP Books, 2002.
[3] D. E. Simon, *An Embedded Software Primer*. Addison-Wesley, 1999.
[4] H. Kopetz, *Real-time systems: design principles for distributed embedded applications*. Kluwer Academic Publishers, 1997.
[5] Q. Li and C. Yao, *Real-time concepts for embedded systems*. CMP Books, 2003.
[6] J. Stankovic and K. Ramamrithan, Eds., *Tutorial on Hard Real-Time Systems*. IEEE Computer Society Press, 1988.
[7] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*, 4th ed., ser. The Kluwer international series in engineering and computer science. Real-time systems. Kluwer Academic Publishers, 2002.

[8] MathWorks. [Online]. Available: http://www.mathworks.com