IMPLEMENTAÇÃO DE UMA POLÍTICA DE GESTÃO ENERGÉTICA NO FREERTOS

CÉSAR A. M. DOS SANTOS, RODRIGO M. A. DE ALMEIDA, CARLOS H.V. MORAES

Grupo de Engenharia Biomédica, IESTI, Universidade Federal de Itajubá
Rua Coronel Rennó, número 7, Bairro Centro, Itajubá – MG, CEP 37500-050
E-mails: cesaraugusto@unifei.edu.br, rodrigomax@unifei.edu.br, valerio@unifei.edu.br

Abstract— This article demonstrates different power saving modes applied to real time operating systems, whenever no tasks are being executed. It is expected that low power modes supported by the microcontroller diminishes energy consumption, proportional to the number of internal modules deactivated. On the other hand, this economy can become invalid based on system processing load, because instant consumed current after changing from a low power state to active mode is higher than a same system under continuous operation. This way, a power saving policy will be presented as well as how to apply such, simultaneously minimizing energy consumption and balancing it with load processing.

Keywords— Real time systems, Embedded systems, Microcontrollers, Energy saving

Resumo— Este artigo demonstra a utilização de diferentes modos de economia de energia aplicados em sistemas operacionais de tempo real, quando o mesmo não possui tarefas a serem executadas. Espera-se que, ao entrar em modos de baixo consumo oferecidos pelo próprio microcontrolador, haja uma diminuição no consumo de energia, proporcional ao número de módulos internos que são desativados. Entretanto, a economia pode não se mostrar válida de acordo com a carga de processamento que o sistema possui, pois a corrente instantânea consumida após a transição do estado de baixo consumo para o modo ativo é superior ao consumo do sistema em operação constante. Desta forma, será apresentada uma política de economia de energia e como aplicá-la, simultaneamente minimizando o consumo energética e balanceando a exigência de processamento.

Palavras-chave— Sistemas de tempo real, Sistemas embarcados, Microcontroladores, Economia de energia

1 Introdução

Sistemas operacionais de tempo real (RTOS - *Real Time Operating Systems*) são uma forma de abstração utilizadas na programação de sistemas embarcados, para simplificar e facilitar a implementação de atividades com requisitos de tempo real, os quais possuem *kernels* capazes de atender demandas temporais. Em geral, um RTOS consome recursos de processamento para gerenciar uma lista de tarefas, realizando comparações entre o intervalo de reexecução definidos por cada tarefa e um relógio interno (*tick*) do próprio sistema.

A programação destes sistemas ainda exige uma série de cuidados distintos não existentes em programas voltados para *desktop*, geralmente por restrições de memória, processamento, dimensão, peso, rigidez, entre outros (Juang et al., 2002). Uma das características importantes em sistemas embarcados é o consumo de energia, visto que diversos sistemas são desenvolvidos para operar sob baterias.

Diversos trabalhos já propuseram soluções para determinar e resolver este impasse, seja ele por sistemas reconfiguráveis, que podem combinar a flexibilidade de um processador de propósito geral e a eficiência de um hardware dedicado através de tecnologia FPGA (Saha, Sarkar, Chakrabarti, 2015) ou através de técnicas que combinam software e hardware, tais como ajuste dinâmico de tensão e frequência de operação (DVFS - *Dynamic Voltage-Frequency Scaling*) ou gerenciamento dinâmico de energia (DPM - *Dynamic Power Management*) (Zhuralev et al., 2013).

Entretanto, ao se trabalhar com técnicas de economia de energia baseadas em modos de baixo consumo do microcontrolador, ocorre um aumento repentino de corrente ao retornar ao seu modo ativo, podendo anular a economia prevista.

Esta análise foi implementada no sistema operacional de código aberto FreeRTOS (FreeRTOS, 2016a). O artigo apresentará uma política de gestão energética, que define limites de carga de processamento para o consumo de corrente desejado. Foi utilizada uma placa com o microcontrolador MSP430F5172 como plataforma de testes e obtenção de resultados práticos.

2 Desenvolvimento

A Texas Instruments (T.I., 2015) define cinco modos de baixo consumo de energia (*low power mode* – LPM), além do modo ativo (*active*). Cada LPM desabilita uma certa quantidade de periféricos no sistema.

Para este trabalho, o modo LPM4 foi descartado por não possuir nenhum *timer* em funcionamento, requisito exigido pelo FreeRTOS. Os modos LPM0 e LPM2 também não foram analisados, visto que possuem tempos de *wake-up* similares, aos modos LPM1 e LPM3, respectivamente. Este tempo é importante, pois pode reduzir a capacidade de processamento do sistema. Os modos LPM1 e LPM3 foram escolhidos, em detrimento aos outros dois, por serem mais econômicos.

O cenário mencionado de aumento de consumo de corrente é demonstrado na Figura 1: um sistema,

após seu processo de inicialização, também chamado de POR (*Power On Reset*), executa suas tarefas, e quando a fila de execução está vazia, diz-se que o sistema está ocioso. Então, o mesmo pode aguardar para executar a tarefa assim que a mesma estiver disponível (Figura 1a), ou pode colocar seu microcontrolador para entrar em modo de economia de energia, retornando ao modo *Active* quando houverem tarefas na fila de execução (Figura 1b).

No segundo caso, nota-se que o consumo instantâneo de corrente é superior se comparado ao consumo do modo ativo em execução contínua.

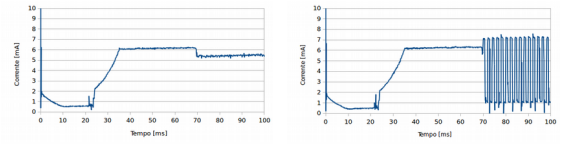

Figura 1. Ao retornar de um modo de economia de energia, o consumo de corrente instantâneo de um microcontrolador se mostra superior ao seu consumo em execução contínua.

O fenômeno mostrado não é esperado pelo programador do sistema, assim como não é previsto no *datasheet* do fabricante (T.I., 2015)(T.I., 2016).

Para que o problema seja formulado, é imprescindível a determinação dos intervalos de tempo de execução de uma tarefa (t_{ON}) e em economia de energia (t_{OFF}). Além disso, serão necessárias as medições de consumo de corrente do sistema em modo ativo (i_{ON}), em economia de energia (i_{OFF}) e em estado ativo após retorno do modo de economia de energia (i_{ON2OFF}). Conforme visto o fenômeno na Figura 1, as variáveis anteriormente definidas podem ser aproximadas pela Figura 2.

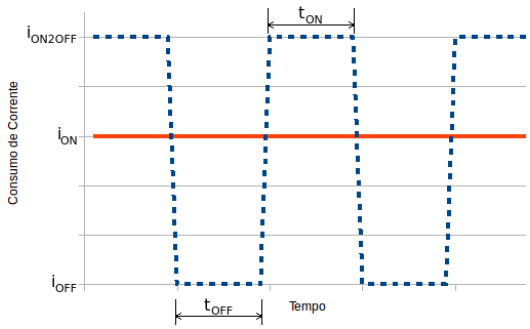

Figura 2. Definição das variáveis t_{ON}, t_{OFF}, i_{ON}, i_{OFF} e i_{ON2OFF}.

O consumo médio de corrente do sistema (i_{LPM}), para uma aplicação que utilize modos de baixo consumo de energia, pode ser equacionado de acordo com (1).

$$i_{LPM} = \frac{t_{ON} \cdot i_{ON\,2OFF} + t_{OFF} \cdot i_{OFF}}{t_{ON} + t_{OFF}} \quad (1)$$

Em (1) é considerado um sistema periódico com apenas uma tarefa em execução. Para generalizar o conceito, podemos adotar a ideia de carga de processamento ($C_{\%}$). Este pode ser definido como o somatório dos tempos consumidos (t_{ON_i}) pelas N tarefas do sistema, dividido pelo tempo total (t), conforme (2). Este somatório representa o tempo total ativo, consumindo a corrente i_{ON2OFF} .

$$C_{\%} = \frac{\sum_{i=0}^{N} t_{ON_i}}{t} \quad (2)$$

De modo análogo, o tempo ocioso do sistema é definido pelo somatório de todos os intervalos de tempo em economia de energia dividido pelo tempo total. O tempo ocioso também pode ser definido como o tempo total menos o somatório dos tempos em execução, conforme (3). Substituindo (2) em (3) tem-se (4).

$$C_{\%_{IDLE}} = \frac{\sum_{i=0}^{N} t_{OFF_i}}{t} = \frac{t - \sum_{i=0}^{N} t_{ON_i}}{t} \quad (3)$$

$$C_{\%_{IDLE}} = 1 - C_{\%} \quad (4)$$

Expandindo (1) para considerar todos os tempos das tarefas em execução, bem como o somatório dos tempos ociosos, determina-se (5).

$$i_{LPM} = \frac{\sum_{i=0}^{N} t_{ON_i} \cdot i_{ON\,2OFF} + \sum_{i=0}^{N} t_{OFF_i} \cdot i_{OFF}}{t} \quad (5)$$

Utilizando (2) e (4) em (5), obtém-se (6).

$$i_{LPM} = C_{\%} \cdot i_{ON2OFF} + (1 - C_{\%}) \cdot i_{OFF} \quad (6)$$

2.1 Política de economia de energia

O foco deste estudo é determinar em quais situações é válido adotar modos de economia de energia durante o período ocioso do sistema. Para tal, a partir de uma carga de processamento $C_{\%}$ estabelecida para o RTOS, define-se a política de gestão energética.

Através do equacionamento matemático dado por (6), para que o consumo de corrente médio do sistema seja válido, deve-se determinar se o sistema entrará em LPM enquanto estiver ocioso, ou se permanece em modo ativo, tal qual definido em (7).

$$i_{LPM}(C_{\%}) \begin{cases} >= & i_{ON} \Rightarrow Modo\ Ativo \\ < & i_{ON} \Rightarrow LPM \end{cases} \quad (7)$$

2.2 Hardware utilizado

O esquemático do hardware utilizado é mostrado na Figura 3. O mesmo possui somente resistores e capacitores necessários para sua operação, sem haver periféricos externos aderidos ao sistema.

O monitoramento dos testes foi viabilizado por duas modificações no hardware. A primeira foi utilizar um terminal do microcontrolador para supervisionar quando a tarefa *Idle* está em execução. A segunda alteração permite a análise das trocas de contexto através de outro terminal do microcontrolador.

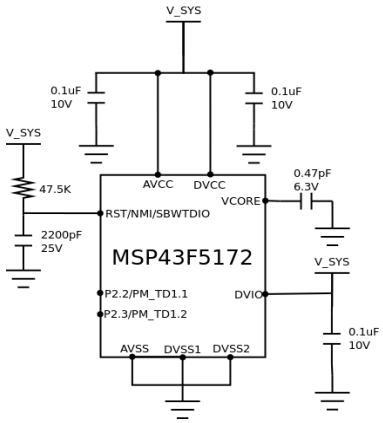

Figura 3. Esquemático do circuito básico para operação do MSP430F5172.

2.3 Implementação prática com o FreeRTOS

O FreeRTOS prevê a implementação de uma tarefa denominada *Idle*, que é executada quando não há tarefas ativas. A documentação sugere que o sistema seja colocado em modo de economia de energia quando a tarefa *Idle* estiver em execução (FreeRTOS, 2016b). A implementação desta tarefa está descrita no Código 1. A função foi desenvolvida para contemplar todos os modos de economia de energia desejados.

Código 1. Implementação de *Idle* do sistema FreeRTOS, voltado à economia de energia

```
// Escolhe o modo de consumo:
#define LPM_MODE 1

void vApplicationIdleHook (void) {
    /* Acende o LED de depuração */
    P2OUT |= 1 << 2;

    /* Não utilizar LPM */
    #if (LPM_MODE == -1)
        return;
    #endif

    /* Utiliza LPM1 */
    #if (LPM_MODE == 1)
        __bis_SR_register(LPM1_bits + GIE);
    #endif

    /* Utiliza LPM3 */
    #if (LPM_MODE == 3)
        __bis_SR_register(LPM3_bits + GIE);
    #endif
}
```

A aplicação-teste foi baseada em uma única tarefa a ser executada no sistema, a qual realiza algumas atividades, e ao seu término, é suspensa durante dois ciclos de *tick* do sistema. Logo, a carga de processamento do sistema permanece em 50%. A implementação desta tarefa é mostrada no Código 2.

2.4 Medições

Foram utilizados um osciloscópio Tektronix MDO4054 e uma *Source Meter* Keithley 2400. O primeiro foi ligado aos terminais mapeados para exibir informações da troca de contexto e da tarefa *Idle*, tal qual visto na Figura 4.

Código 2. Implementação de uma tarefa no sistema FreeRTOS

```
void vTaskCode() {
    portTickType xLastWakeTime;
    xLastWakeTime = xTaskGetTickCount();
    int i = 0;
    for (;;) {
        for (i = 0; i < 100; i++);
        vTaskDelayUntil(&xLastWakeTime, 2);
    }
}
```

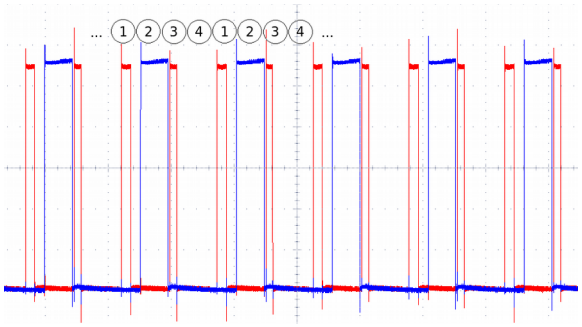

Figura 4. Visualização das trocas de contexto e da tarefa *Idle*.

Os tempos marcados com o número "2", na Figura 4, representam as várias execuções da tarefa *Idle*. Os intervalos marcados com "1" e "3" apresentam a troca de contexto entre as diferentes tarefas do sistema. O tempo indicado por "4" indica a execução da tarefa implementada no Código 2.

Com isso, é possível visualizar que, para cada troca de contexto, ou o sistema volta a executar a tarefa (marcação "4") ou entra em *Idle* (marcação "2"), o que configura uma carga de processamento $C_{\%}=50\,\%$, conforme esperado.

Quanto à corrente, foram obtidas as Figuras 5, 6 e 7. Foi destacada somente a área que contém execução do FreeRTOS, descartando-se o tempo de POR.

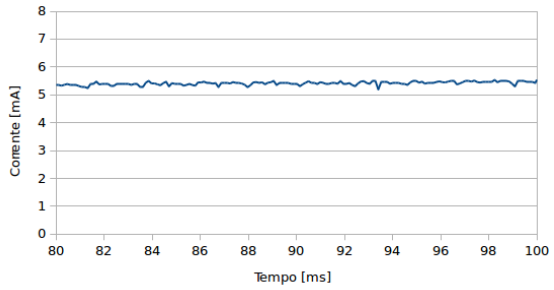

Figura 5. Consumo de corrente do sistema em *Active*.

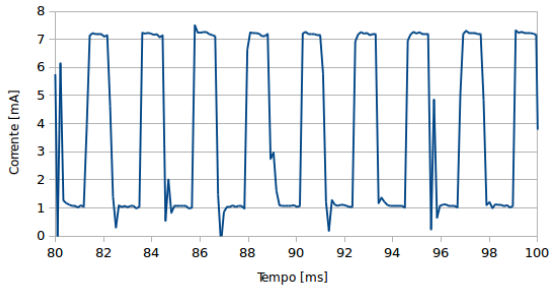

Figura 6. Consumo de corrente do sistema em LPM1, quando em *Idle*.

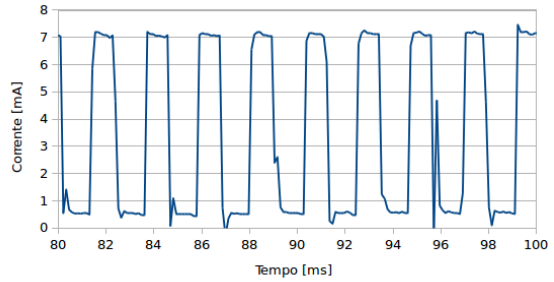

Figura 7. Consumo de corrente do sistema em LPM3, quando em *Idle*.

Os ruídos observados, que levam o sinal para níveis mais baixos que o patamar de 1mA (ou além de 7mA), são efeitos de *crosstalk* causado pelos terminais que são acionados justamente para monitorar o tempo de execução das tarefas, interferindo no sinal.

3 Resultados

A leitura obtida no modo LPM3 é bastante similar ao do LPM1, sendo a única diferença o consumo no estado de economia, em média 500µA em LPM3, em comparação aos 1mA do LPM1.

Um sistema operando permanentemente em modo *Active* consome cerca de 5,4mA. No entanto, quando o sistema opera em LPM, ele consome 7mA após retornar do modo de baixo consumo, representando um aumento de 23,5% em relação ao sistema continuamente em *Active*.

A Tabela 1 apresenta os valores encontrados para os consumos em cada um dos modos, considerando-se apenas os momentos sem execução de tarefas, para comparar os resultados com os informados pelo fabricante.

Tabela 1. Consumo de corrente para os diferentes modos de energia

Modo de energia	**Intervalos dados pelo fabricante**	**Consumo médio medido da placa em *Idle***
Active	3640 a 6150 µA	5382 µA
LPM1	85 a 104 µA	1076 µA
LPM3	1.2 a 3.0 µA	597.8 µA

Comparando os valores obtidos com os dados fornecidos pelo fabricante, percebe-se uma grande discrepância, principalmente para os modos de mais baixo consumo de energia.

De acordo com a Renesas (2013), "não é possível saber se os dados fornecidos pelo fabricante foram determinados executando uma instrução 'NOP', um *loop* infinito ou um algoritmo específico. Além disso, os fabricantes não informam quais periféricos estão ligados ou desligados. Deste modo, o que é fornecido em um *datasheet* representa uma referência, mas não uma realidade absoluta para todo e qualquer projeto".

Baseando-se na equação (6) e nas medições, obtém-se a Tabela 2. De acordo com estes dados, havendo uma carga de processamento de 50%, obtém-se uma economia de apenas 30%, em relação ao modo permanentemente em *Active*.

Tabela 2. Consumo de corrente médio do sistema para diferentes tipos de LPM

Estado	i_{OFF} [µA]	i_{ON2OFF} [µA]	i_{LPM} [µA]	Redução [%]
Active[1]	-	5382	5382	0
LPM1	1076	7035	3717	30,9
LPM3	598		3451	35,9

[1] neste caso, i_{LPM} representa i_{ON}, visto que não há mudança de estado

O consumo médio do sistema depende do modo LPM utilizado e da carga de processamento do sistema. Através de (6) pode-se inferir o ponto de operação C_{MAX}, para o qual o sistema, com LPM, consuma a mesma quantidade de energia do que o sistema permanentemente ligado. Para isto deve-se fazer $i_{LPM}=i_{ON}$ em (6), obtendo (8).

$$C_{MAX}=\frac{i_{ON}-i_{OFF}}{i_{ON2OFF}-i_{OFF}} \quad (8)$$

Para os diferentes modos de LPM do processador estudado, tem-se as curvas de carga de processamento contra o consumo de corrente, exibidas na Figuras 8.

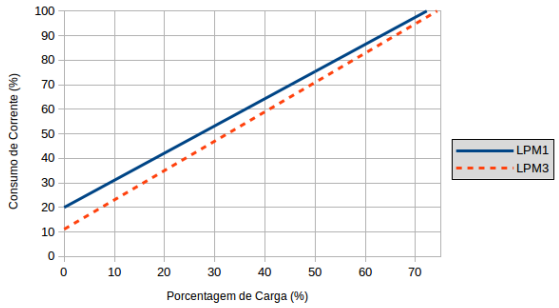

Figura 8. Relação entre carga de processamento e a variação no consumo de corrente, comparado com o sistema utilizando os modos LPM1 e LPM3.

Através da Figura 8, pode-se ver que um sistema que utiliza LPM1 como método de economia de energia dentro tarefa *Idle*, é capaz de reduzir o consumo em até 80%. No entanto, não realiza nenhuma tarefa útil ($C_{\%}=0$). Para LPM3, a diminuição de corrente é de 89%.

Observando o aspecto de carga máxima de processamento, em LPM1, o valor de C_{MAX} é 72,3%, ou seja, apesar de consumir a mesma quantidade de energia que um sistema permanentemente ativo, ele está limitado a 72,3% da capacidade de processamento. Se o sistema necessitar de maior capacidade de processamento, os ganhos com a economia de energia gerados pelo LPM1 são anulados. Para o LPM3, o valor crítico C_{MAX} é dado por 74,3% de capacidade de processamento.

4 Conclusões

Tendo em vista uma ascensão de tecnologias que necessitam minimizar seu consumo de energia, sem interferir na capacidade de processamento, uma alternativa prática é delinear uma estratégia de economia no próprio Sistema Operacional para atingir ambos objetivos. A definição de uma metodologia estática para a escolha dos métodos de economia de energia pode ser feita de maneira simples através da tarefa *Idle* do FreeRTOS.

Pelos resultados obtidos, percebeu-se que o sistema em estudo possui dois níveis distintos de consumo de energia na execução de tarefas. O consumo pós-LPM é superior ao do sistema permanentemente ligado, possivelmente devido ao religamento dos periféricos internos. Sendo assim, utilizando LPM para uma carga de processamento de 75%, consome-se a mesma quantidade de energia que o sistema com 100%, sem modo de baixo consumo de energia habilitado.

Assim, na definição de uma política de gestão energética é importante levar em conta esta diferença nos níveis de corrente, bem como a economia gerada nos modos de baixo consumo. De posse destes valores e da necessidade de processamento do sistema, é possível utilizar a política apresentada em (7) para minimizar o consumo de energia.

Conclui-se que não se deve analisar ou desenvolver uma política de gestão energética se baseando somente nos modos de economia de energia do processador. É necessário realizar uma análise conjunta da questão energética com a capacidade de processamento. Esta análise deve ser pautada em medições reais de consumo, visto que as informações fornecidas pelos fabricantes podem não contemplar todas as características do sistema analisado.

Além das questões abordadas, o modo de economia de energia gera um atraso cada vez que é ativado ou desativado. Este fato, aliado à carga máxima de processamento (C_{MAX}), pode reduzir ainda mais a quantidade de processamento disponível para execução de tarefas.

Para trabalhos futuros, é interessante analisar este comportamento para diferentes taxas de *clock* e modelos de microcontroladores. Outra abordagem seria desenvolver uma metodologia que analise o comportamento dinâmico da carga de processamento, adequando seu consumo de energia, segundo a política apresentada neste trabalho.

5 Agradecimentos

Agradecemos à Capes, Fapemig e CNPq pelo apoio prestado neste projeto.

Referências Bibliográficas

FREERTOS. (2016a) FreeRTOS – Market Leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions. [Online] Disponível em: http://www.freertos.org. [Acessado em: 12 de Abril de 2016].

FREERTOS. (2016b) Idle Task and Idle Hook. [Online] Disponível em: http://www.freertos.org/RTOS-idle-task.html. [Acessado em: 03 de Maio de 2016].

JUANG, P. et al. (2002) Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. ACM.

RENESAS (2013) The True Low Power Concept - Implementing Powerful Embedded Controls with Minimum Energy Requirements. *Renesas Electronics America*. [Online] Disponível em: http://am.renesas.com. [Acessado em 12 de Setembro de 2013].

SAHA, S.; SARKAR, A.; CHACKRABARTI, A. (2015) Scheduling Dynamic Hard Real-Time Task Sets on Fully and Partially Reconfigurable Platforms. IEEE Embedded Systems Letters, Vol. 7, NO. 1.

T.I. (2015) MSP430F51x2 and MSP430F51x1 Mixed-Signal Microcontrollers.

T.I. (2016) MSP430F5172 Device Erratasheet.

ZHURALEV, S. et al. (2013) Survey of Energy-Cognizant Scheduling Techniques. IEEE Transactions on Parallel and Distributed Systems. Vol. 24, NO. 7.