EA871 – LAB. DE PROGRAMAÇÃO BÁSICA DE SISTEMAS DIGITAIS

EXPERIMENTO 9 – Relógio em Tempo Real

Profa. Wu Shin-Ting

OBJETIVO: Apresentação do princípio de funcionamento de temporizadores e uma aplicação em relógios.

ASSUNTOS: Configuração e programação do MKL25Z128 para processamento de eventos temporais.

O que você deve ser capaz ao final deste experimento?

Entender a diferença entre diferentes formatos de representação de tempo e a conversão entre eles.

Entender o princípio de funcionamento de um temporizador.

Saber computar um intervalo de tempo com base nos dados de um temporizador.

Saber configurar um temporizador para geração de eventos periódicos.

Programar os temporizadores *SysTick* , *PIT*, *LPTMR* e *RTC* integrados em MKL25Z128.

Projetar um relógio digital com uso de um temporizador.

INTRODUÇÃO

Temporizadores constituem módulos básicos de qualquer microcontrolador para assistir a coordenação dos eventos em função de tempo [1]. O princípio de funcionamento de um temporizador consiste em **contar** a quantidade N de pulsos de um *clock* de frequência *f* conhecida, a partir dos quais podemos computar o intervalo de tempo *t* correspondente por

$$t = N \times \frac{1}{f}$$

Se substituirmos a fonte de *clocks* por uma fonte de eventos externos, o mesmo circuito pode ser aproveitado para fazer **contagem** de eventos externos. Temporzadores estão, portanto, intimamente relacionados com contadores.

O valor máximo REF de contagem de um contador depende da quantidade dos *bits* que ele possui. Quando a contagem atinge este valor máximo, dizemos que ocorreu um *Overflow*. Todos os temporizadores conseguem detectar este evento e, em conjunto com um circuito de interrupção, tratá-lo como uma interrupção. Muitos temporizadores dispõem de divisores de frequência, comumente denominados **prescaler**, para reduzir a frequência da fonte de *clock*. Alguns temporizadores tem integrados um circuito que divide a frequência de ocorrência de *overflows* do contador, de forma que a frequência da sua ocorrência seja reduzida para o temporizador. Este circuito é denominado **postscaler**. Levando em conta estes divisores de frequência, define-se como **período** de um temporizador o intervalo de tempo *T* gasto para fazer uma contagem completa até que gere um evento de *Overflow* do temporizador

$$T = REF \times \frac{prs}{f} \times pos$$

,

sendo prs e pos divisores de frequência prescaler e postscaler, respectivamente. Note que é possível controlar o período de um temporizador por 4 parâmetros: REF, prs, pos e *f*. Mais especificamente, para dados *f*, *prs* e *pos*, podemos controlar o período *T* de um temporizador através da configuração do valor de referência máximo de contagem

$$REF = T \times \frac{f}{prs \times pos} .$$

Objetivando a mostrar a aplicação destes conceitos em diversos temporizadores, apresentaremos de forma comparativa 4 módulos de temporizadores disponíveis em MKL25Z128: *SysTick* (Seção B.3.3, página 275, em [2]), *PIT (Periodic Interrupt Timer)* (Capítulo 32, página 573, em [3]), *LPTMR* (Low Power Timer (Capítulo 33, página 587, em [3]) e *RTC* (*Real Time Clock*) (Capítulo 34, página 597, em [3]). São levados em conta aspectos relacionados com fontes de *clock*, tamanho do seu contador em termos de *bits*, divisores de frequência, forma de contagem, eventos de interrupção e funções implementadas. Por exemplo, *SysTick* tem integrado um contador de 24 *bits*, *LPTMR* um de 16 *bits* e *PIT* e *RTC* um de 32 *bits*. No módulo *PIT*, é ainda possível encadear 2 temporizadores para formar um contador de 64 *bits.*

O módulo RTC é um temporizador dedicado para contar segundos, sob a condição de que a frequência da fonte de *clock* seja 32,768kHz. O seu contador TSR de 32 *bits* é atualizado a cada 2^{15} *ticks* do clock por meio de um prescaler TPR, ou seja,

$$Ciclo\ completo_{TPR} = 2^{15} \times \frac{1}{f} s = 2^{15} \times \frac{1}{32768} s .$$

Para exibir os horáriocs contabilizados, seja no formato de 24 horas ou de 12 horas, é necessário converter os valores numéricos na unidade de segundos para uma sequência de glifos cujo formato padrão é HH:MM:SS [4].

EXPERIMENTO

Neste experimento vamos implementar, com base no projeto **controle_relogio_digital**, o projeto **relogio_digital** em que o horário é mostrado no meio da primeira linha do visor do LCD no formato padrão HH:MM:SS (24 horas) (**estado normal**). É possível ajustar o horário por meio de três chaves (**estado de ajuste**). Sempre se passa **do estado normal para o estado de ajuste** quando se aciona a botoeira IRQ12. A chave IRQ12 seleciona cicilicamente os campos HH → MM → SS → HH. Como realimentação visual, o cursor do LCD é posicionado no campo do dígito menos significativo e fica piscando. A chave IRQ5 incrementa o valor no campo selecionado enquanto a chave NMI decrementa o valor. Se dentro de um intervalo de tempo correspondente a *timeout* nenhuma chave for acionada, o relógio volta ao estado normal.

1. No projeto **controle_relogio_digital** são distinguidos dois estados de operação para as atualizações do visor do LCD: NORMAL e AJUSTE. Em qual momento é chaveado o estado NORMAL para o estado AJUSTE? E em qual momento é chaveado o estado AJUSTE para o estado NORMAL? Justifique.

   Quais são as instruções executadas no chaveamento de NORMAL para AJUSTE? E quais são as instruções executadas no chaveamento de AJUSTE para NORMAL? Justifique o uso destas intruções.

2. Foi usado o módulo SysTick no pojeto **controle_relogio_digital** para implementar *timeout*. Para um *timeout* de 2s, qual valor pode ser setado no registrador SYST_RVR? Quantos ciclos completos são necessários para perfazer 2s? Justifique.

   Em qual ponto do seu programa deve ser iniciado a contagem de *timeout* pelo módulo SysTick e em qual ponto do programa o módulo SysTick deve ser desabilitado?

3. Para uma fonte de *clock* de 1kHz, escreva a expressão que compute horários, em segundos, a partir dos valores amostrados dos registradores RTC_TSR e RTC_TPR.

   Converta horários em **s_cur** para horas, minutos e segundos, em valores inteiros apropriados para armazenar no vetor **horario**. Faz sentido inserir esta conversão no seu fluxo de controle? Justifique.

   Converta horários em horas, minutos e segundos em valores inteiros para armazenar na variavel **s_cur**. Faz sentido inserir esta conversão no seu fluxo de controle? Justifique.

4. Como realimentação visual dos campos em atualização durante o estado AJUSTE, foi ativado o modo de piscada do cursor do LCD. Qual é a instrução do LCD qua faz esta ativação? E como disativá-lo?

   Estando os horários armazenados em horas, minutos e segundos no vetor **horario**, como são atualizados estes valores nos acionamentos das botoeiras IRQ5 e IRQ4?

   Ao entrar na rotina PORTA_IRQHandler pelas botoeiras IRQ4 ou IRQ5, só faz sentido atualizar os horários se o relógio estiver no estado AJUSTE. Esta condição não é considerada plenamente no projeto **controle_relogio_digital**. Veja o que acontece se você acionar IRQ4/IRQ5 antes de acionar IRQ12. Como você condicionaria o processamento dos eventos de IRQ4 e IRQ5? Justifique.

   Qual é a função das variáveis flagP e flagH no projeto **controle_relogio_digital**? O que acontece se estas duas variáveis forem removidas?

5. Não é integrada a função alarme no projeto **controle_relogio_digital**, por causa da imprecisão do valor setado em relação ao valor de interesse. Tanto que na forma como a função de alarme foi implementada no projeto **timer_rtc** os eventos de alarme podem deixar de serem gerados após um certo tempo. Explique estas afirmações.

6. Implemente o aplicativo **relogio_digital** em C.

7. Documente todas as funções que não foram geradas pelo IDE CodeWarrior.

RELATÓRIO

Para este experimento, responda as questões 1 a 5 num arquivo em **pdf**, implemente e documente o projeto **relogio_digital.** Exporte o projeto no ambiente IDE CodeWarrior para um arquivo em formato zip. Suba **os dois arquivos, em separado**, no sistema *Moodle*. Não se esqueça de identificar todos os seus arquivos de códigos com a palavra reservada "@author" de Doxygen.

REFERÊNCIAS

[1] Chandrasekaran, Siddharth. Timer/Counter Module – A Controller Independent Guide.
https://embedjournal.com/timer-modules-guide/
[2] ARMv6-M Architecture Reference manual
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/ARMv6-M.pdf
[3] *KL25 Sub-Family Reference Manual – Freescale Semiconductors (doc. Number KL25P80M48SF0RM)*, Setembro 2012.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf
[4] Convert seconds to HH:MM:SS
https://www.tools4noobs.com/online_tools/seconds_to_hh_mm_ss/

Novembro de 2020