EA871 – LAB. DE PROGRAMAÇÃO BÁSICA DE SISTEMAS DIGITAIS

EXPERIMENTO 11 – *TPM – Input Capture e Output Compare*

Profa. Wu Shin-Ting

OBJETIVO: Apresentação das funcionalidades *Input Capture* e *Output Compare* dos módulos TPMx (*Timer/PWM*).

ASSUNTOS: Controle com maior precisão os instantes de captura dos sinais de entrada e de mudança dos estados dos sinais (digitais) de saída, programação do MKL25Z128 para processamento destes sinais via módulos TPMx.

O que você deve ser capaz ao final deste experimento?

Saber programar MKL25Z128 para capturar o instante em que um evento externo causou uma interrupção (*Input Capture*).

Saber programar MKL25Z128 para mudar o estado de um pino digital de saída num instante pré-programado (*Output Compare*).

Saber aplicar essas técnicas na implementação de dispositivos que exigem precisão em tempo.

INTRODUÇÃO

Vimos no experimento 10 que o módulo TPM suporta três funções: PWM (*Pulse With Modulation*), *Input Capture* e *Output Compare*. Vimos ainda que, com a técnica de modulação por largura de pulso, podemos controlar a potência média transmitida a uma carga e a aplicamos na geração de uma variedade de cores com base no princípio aditivo. Na prática, essa técnica é bastante utilizada no controle da velocidade como *coolers*. Além da capacidade de gerar sinais modulados pela largura de pulso, o módulo TPM consegue **rotular os eventos de interrupção com os instantes em que ocorreram** (*InputCapture*) e **gerar sinais digitais em pinos de saída nos instantes pré-programados** (*Output Compare*). Usando como base o contador `TPMx_CNT` do módulo, a função

Input Capture captura o valor deste contador no registrador `TPMx_CnV` no momento em que um evento pré-especificado ocorrer. Em paralelo à captura é ativado o estado de interrupção do canal em 1 (Seção 31.4.4 em [1]). Para que o sinal de entrada seja corretamente amostrado, sua frequência deve ser 2 vezes menor que a frequência de contagem do relógio do móduloTPMx.

Output Compare coloque no pino do canal um sinal pré-configurado quando o valor setado no registrador `TPMx_CnV` iguale o valor de contagem no contador `TPMx_CNT[COUNT]`. Neste momento, o estado de interrupção do canal é ativado em 1 (Seção 31.4.5 em [1]).

A função *Input Capture* é bastante útil para medir com precisão o intervalo de tempo `t` entre dois eventos, se capturarmos os valores T1 e T2 do contador `TPMx_CNT` no momento de suas ocorrências. Junto com a contagem `N` de ciclos de contagem completa, detectáveis através de eventos de *Overflow* gerados por todos os temporizadores, inclusive módulos TPMx, pode-se derivar para

$$\text{T1} >= \text{T2}: \quad t = \left(\frac{(MOD + T2 - T1)}{MOD} + (N-1)\right) \times Período$$

$$\text{T1} < \text{T2}: \quad t = \left(\frac{(T2 - T1)}{MOD} + N\right) \times Período \quad ,$$

onde MOD é a contagem máxima que corresponde a um ciclo completo de contagem e o *Período* corresponde ao intervalo de tempo deste ciclo completo de contagem. Vale comentar que não há um contador de *Overflow* nos módulos TPMx. A contagem deve ser feita por *software*.

A função *Output Compare* permite gerar, por *hardware*, um sinal de saída num específico instante. Isso aumenta a precisão e a velocidade da resposta do sistema.

EXPERIMENTO

Neste experimento vamos desenvolver o projeto **cronometro** em que a contagem se inicia ao acionarmos na chave IRQ5 e pára a contagem quando a chave é reacionada. Para sinalizar visualmente o período de tempo em que acontece a contagem o *led* D4 do shield FEEC 871 começar piscar em torno de 10us após o primeiro acionamento e mudar para o estado "apagado" após o segundo acionamento. O valor medido é mostrado, em segundos, na segunda linha do visor do LCD junto com o bitmap do cronômetro no canto superior direito (endereço 0x0F).

1. No projeto **timer_tictoc** foram usados dois módulos, `GPIO` e `TPM0 (output compare)`, para controlar o *led* D4. É, porém, possível controlar os dois estados do *led* D4, piscante e apagado, com somente o módulo TPM0 mantendo o canal 3 sempre habilitado. Como deve ser configurado o registrador de controle e de estado do canal, TPM0_C3SC, para que a saída do sinal no pino sejam alternado? E para ficar no nivel 0? Qual deve ser o estado do canal para modificar o sinal de saída do seu pino?
2. Em qual ponto do seu programa você salva o valor de contagem do contador `TPMx_CNT` capturado no registrador `TPMx_CnV` como o instante inicial T1 do intervalo de tempo em medição? E onde é salvo o instante final T2? Justifique.
3. Como você pode contar a quantidade `N` de ciclos completos de contagem (*overflow*)? E em qual ponto do programa é incluído o incremento de `N`? Determine o valor TPMx_MOD e o valor Prescaler que permitam contabilizar um intervalo na ordem de 10us (resolução) e minimizar a quantidade de *Overflows*..
4. Quantas interrupções você precisa habilitar no módulo TPM0? Quantas IRQs você precisa habilitar no NVIC? Quantas rotinas de serviço você precisa definir? Qual seria a ordem de prioridade de atendimento das interrupções habilitadas?
5. Represente numa linha de tempo como você alternaria, em função dos instantes de acionamento da chave IRQ5 e dos eventos de *overflow*, o estado piscante e o estado apagado do pino PTC4 em que o *led* D4 está conectado. (Lembre-se de que todas as contagens são cíclicas.)
6. Como você processa na função **main** as informações a serem exibidas no LCD de forma coordenada com as suas capturas na(s) rotina(s) de serviço?
7. Implemente o aplicativo **cronometro** em C.
8. Documente todas as funções que não foram geradas pelo IDE CodeWarrior.

RELATÓRIO

Para este experimento, responda as questões 1 a 6 num arquivo em **pdf**, implemente e documente o projeto **led_multicores.** 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] KL25 Sub-Family Reference Manual – Freescale Semiconductors (doc. Number KL25P80M48SF0RM), Setembro 2012.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf

Revisado em Novembro de 2020
Revisado em Fevereiro de 2017
Agosto de 2016