UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E COMPUTAÇÃO

EA871 - LAB. DE PROGRAMAÇÃO BÁSICA DE SISTEMAS DIGITAIS

EXPERIMENTO 8 – TPM

Profª Wu Shin-Ting
Aluno: Tiago Eidi Hatta RA: 160388

Proposta do Experimento

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

Metodologia:

Os seguintes conceitos foram abordados para resolver o problema proposto pelo experimento:

Módulo TPM (Timer/PWM): Timer constituído de um contador LPTPM (Low-Power TPM) de 16 bits, também de baixo consumo, com cada um dos seus 6 canais programável para operar numa das 3 funções: Input Capture, Output Compare e PWM (pulse-width modulation).

Input Capture: função na qual sinais no pino (de entrada) de um das seus canais podem disparar interrupções com base em alguma característica pré-especificada e os instantes destas ocorrências capturados e armazenados.

Output Compare: função que gera um nível de sinal no pino (de saída) de um dos seus canais quando o valor do contador do módulo TPM fica igual a um valor pré-definido.

PWM: chamada técnica Modulação de Largura de Pulso, ou seja, através da largura do pulso de uma onda quadrada é possível o controle de potência ou velocidade.[8]

Representação numérica do sistema de Cores RGB: Uma cor no modelo de cores RGB pode ser descrita pela indicação da quantidade de vermelho, verde e azul que contém. Cada uma pode variar entre o mínimo (completamente escuro) e máximo (completamente intenso). Quando todas as cores estão no mínimo, o resultado é preto. Se todas estão no máximo, o resultado é branco. Uma das representações mais usuais para as cores é a utilização da escala de 0 à 255, bastante encontrada na computação pela conveniência de se guardar cada valor de cor em 1 byte (8 bits).[9]

Proposta de Solução

Módulos e Registradores Configurados [1]

Módulo SIM:
- SIM_SCGC5 e SIM_SCGC6: responsáveis por habilitar o clock para cada uma das portas relacionadas aos pinos da placa utilizados no experimento.
- SIM_SCGC4 (System Clock Gating Control Register 4), habilita o clock para o módulo UART0.
- SIM_SOPT2 (System Options Register 2), configura a fonte de relógio.

Módulo PORT: Responsável por dar suporte ao controle das portas da placa principal do kit usado. Registradores:

- PCRn: de acordo com a porta utilizada do pino n, configura a multiplexação para GPIO e o modo de disparo de interrupção de acordo com o tipo de borda do sinal.
Foram configuradas:
- Porta A, pinos 4, 5 e 12: push-buttons da placa auxiliar.
- Porta B, pino 18, led vermelho, MUX = 3 (TPM2_CH0)
- Porta B, pino 19: led verde, MUX = 3 (TPM2_CH1)
- Porta D, pino 1: led azul, MUX = 3 (TPM0_CH1)
- Porta C, pinos 0 a 7: Dados GPIO
Os pinos do LCD são ligados ao da porta C da placa.
- Porta C, pinos 8 a 10: RS, E do LCD e LE do latch, respectivamente.

- ISFR: o bit setado corresponde ao pino que detecta (valor lógico 1) ou não (valor lógico 0) uma interrupção configurada.
Foram configuradas:
- Porta A, pinos 4, 5 e 12.

Módulo GPIO: Comunicação digital do processador do MCU. Registradores:
- PDDR: configura os pinos como saída (lcd) ou entrada (push-buttons) para GPIO.
- PDOR: configura os pinos como saída (lcd) ou entrada (push-buttons) para propósitos gerais.

NVIC (Nested Vectored Interrupt Controller): Controlador de Interrupções
NVIC_ISER: habilita as interrupções de acordo com os pinos configurados
NVIC_ICPR: remove estados de interrupção pendentes
NVIC_IPR4: configura nível de prioridade
NVIC_IPR3: configura nível de prioridade

Módulo UARTx (Universal Asynchornous Receiver/Transmitter): contém um circuito de transmissão e um de recepção. Através destes circuitos os bytes a serem transmitidos são serializados e os bits recebidos são reagrupados em bytes automaticamente, a uma frequência derivada da fonte de clock selecionada.

UART0_C1: Configura diversos recursos de UART0. Deve ser configurado com transmissor e receptor desabilitados.
UART0_C2: Habilita e desabilita transmissor e receptor de UART0.
UART0_C4: Configura a taxa de amostragem, setando o valor de OSR (Over Sampling Ratio).
UART0_C5: Configura o Both Edge Sampling, que permite receber dados em ambas bordas de subida e descida do clock.
UART0_BDH: juntamente com UART BDL, controla o divisor para a taxa de transmissão UART (UART baud rate).
UART0_BDL: similar a UART0_BDH.
UART0_S1: Configura o transmissor.

Módulo TPM: Responsável por configurar cada pino segundo uma determinada função: Input Compare, Output Compare e PWM.

TPMx_SC: indica status de overflow, através de uma flag e controla os bits usados para configurar o enable de interrupções, a configuração do módulo e o fator pré-escala.
TPMx_MOD: contém o valor do módulo do contador do LPTPM. Quando o contador alcança o valor de modulo e incrementa, a flag overflow é setada.

TPMx_CNT: contém o valor do contador LPTPM. Reset limpa o registrador do contador. Escrevendo qualquer valor nos 15 bits menos significativos também limpa o contador.
TPMx_CnSC: contém a flag de estado/interrupção do canal correspodente e controla os bits usados para configurar a habilitação de interrupção, configuração geral do canal e também a função do pino (IC, OC, PWM).
TPMx_CnV: contém o valor capturado do contador LPTPM para modo input ou um valor de comparação para modo output.

Abaixo estão os pseudo-códigos das duas partes do experimento, bem como a explicação da solução baseada nos códigos fontes em C.

Pseudo-Código

```
// Funcao main
MAIN:
INÍCIO
        CONFIGURA PINOS (GPIO, INICIA LCD E NVIC)

        INICIALIZA SYSTICK

        CRIA CARACTERE ê (chama criaCaractere)

        ALOCA BUFFER TRANSMISSOR
        ALOCA BUFFER RECEPTOR

        PARA LOOP VERDADEIRO
                Inicio:                 // rotulo de inicio

                SE estado_pta5 OU estado_pta12 OU estado_terminal iguais a 1
                        SE estado_pta5 ou estado_terminal igual a 1
                                Seleciona a cor de acordo com o tempo apertado da botoeira ou do
        lido pelo terminal
                        FIM_SE
                        SE estado_pta12 ou estado_terminal igual a 1
                                Seleciona frequencia de acordo com aperto da botoeira ou do lido no
        terminal

                        SE estado_terminal = 1
                                Manipula caracteres para selecionar cor ou frequência segundo
        entrada do terminal

                                CONVERTE PARA ASCII O VALOR FLOAT DA FREQUENCIA DE
                        PISCADA

                        FIM_SE

                        MANDA STRING PARA O VISOR: “Cor: “
                        MANDA STRING PARA O VISOR: NOME DA COR SELECIONADA DO LED
                        MANDA STRING PARA O VISOR: “Frequ”
                        MANDA STRING PARA O VISOR: “ê”
                        // (criado anteriomente e armazenado em CGRAM)
                        MANDA STRING PARA O VISOR: “ncia: ”
                        MANDA STRING PARA O VISOR: FREQUENCIA CONVERTIDA EM ASCII

                        MANDA STRING PARA O TERMINAL: “Cor: ”
                        MANDA STRING PARA O TERMINAL: NOME DA COR SELECIONADA DO
        LED
```

```
                MANDA STRING PARA O TERMINAL: “Frequência: ”
                MANDA STRING PARA O TERMINAL: FREQUENCIA CONVERTIDA EM
ASCII
        FIM_SE

        SE estado_pta5 e estado_pta12 e estado_terminal igual a 0
                // botoeiras soltas e sem mensagem recebida no terminal

                ACENDE LED DE ACORDO COM O VALOR DE INDICE ATRAVÉS DE
        MODULO TPM

                TEMPO: INVERSO DA FREQUENCIA SELECIONADA POR NIVEL

                SE  estado_pta5 e estado_pta12 e estado_terminal iguais a 1
                        // aperto de botoeira ou mensagem no terminal detectados
                        // delay é função demorada, por isso a verificação
                        VÁ PARA inicio
                        // retorna para rotulo inicio

                DELAY10US passado como parametro TEMPO determinado de delay

                APAGA LED

                SE  estado_pta5 e estado_pta12 e estado_terminal iguais a 1
                        // aperto de botoeira ou mensagem no terminal detectados
                        // delay é função demorada, por isso a verificação
                        VÁ PARA inicio
                        // retorna para rotulo inicio

                DELAY10US passado como parametro TEMPO determinado de delay
        FIM_SE

    FIM PARA
FIM
```

Código Fonte

Enviado como anexo ao relatório pelo Ensino Aberto. Documentado de acordo com o especificado pela Doxygen.

Testes

Para os testes, foram realizadas as seguintes etapas:

1) Teste da botoeira dada por PTA12 para mudança de cor dos leds segundo pwm e resposta por Input Compare;
2) Teste da botoeira dada por PTA5 para mudança de frequência de piscadas dos leds segundo resposta por Input Compare;
3) Teste do led da placa auxiliar para Timeout, segundo programado o pino apropriado para Output Compare;
4) Teste das funções anteriores com o LCD programado para mostrar a cor e a frequência dos leds;
5) Aperto da botoeira PTA12 e, posteriormente de PTA5, uma de cada vez, durante um tempo até aparição da mensagem “TIMEOUT” no Terminal e no LCD;

6) Teste através do terminal para alteração da frequência e das cores dos leds;
7) Adaptação do código para rotina inicio, que melhorou a resposta do MCU para frequências mais baixas;
8) Teste final com todas as funções definidas, trabalhando com variedade de opções de mudança dos leds (pelo terminal, botoeira, tanto cores como frequências).

Conclusão

O módulo TPM mostrou-se extremamente útil para resposta em relação ao usuário. O timer tem flexibilidade pois pode ser utilizado em situações diferentes (Input Compare, Output Compare e PWM).

É interessante citar a importância do PWM, ilustrado de forma simples no experimento através das cores dos leds. A técnica é empregada em diversas aplicações eletrônicas, sendo utilizada nas velocidade de motores, controle de luminosidade, controle de servo motores e diversas outras aplicações [8].

Além disso, uma limitação que o código teve foi não ter se conseguido utilizar o led da placa auxiliar [7] para indicar Timeout no pressionamento das botoeiras, ao ser detectado por Output Compare. Para isso, tentou-se multiplexar o pino de tal led dinamicamente, dentro da rotina de tratamento de interrupção. Como isso não teve resultados desejados, a alternativa foi mostrar no LCD e no Terminal uma mensagem de TIMEOUT.

Uma outra alteração no código foi a inclusão de um rótulo indicando o tratamento da mudança de cor ou frequência dos leds. Antes da função delay10us, houve a verificação se havia alguma mudança desses parâmetros a ser realizada, pois tal função é demorada e ocupa tempo suficiente do processador para que o usuário sinta essa diferença de resposta. Com isso, caso existisse demanda, o fluxo do programa era jogado para o rótulo inicio, permitindo assim uma maior fluidez para a execução do código.

Referências

[1] FRDM-KL25Z User's Manual.

ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/FRDMKL25Z.pdf

[2] KL25 Sub-Family Reference Manual

ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf

[3] ARMv6-M Architecture Reference Manual – ARM Limited.

ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/ARMv6-M.pdf

[4] EA871 – Descrição do Hardware da Placa Auxiliar.

ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/DescricaoDoHardware.pdf

[5] LCD Tutorial for Interfacing with Microcontrollers. Rickey's World.

http://www.8051projects.net/lcd-interfacing/index.php

[6] Datasheet do display LCD.

ftp://ftp.dca.fee.unicamp.br/pub/docs/ea079/datasheet/AC162A.pdf

[7] EA871 - Descrição do Hardware da Placa Auxiliar

ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/DescricaoDoHardware.pdf

[8] PWM - Modulação Por Largura de Pulso

http://www.mecaweb.com.br/eletronica/content/e_pwm

[9] Sistema de Cores RGB: Wikipedia

https://pt.wikipedia.org/wiki/RGB