UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E COMPUTAÇÃO

01/04/2016

EA871 - LAB. DE PROGRAMAÇÃO BÁSICA DE SISTEMAS DIGITAIS

EXPERIMENTO 3 – Linguagem de Montagem (Assembly)

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

Proposta do Experimento

Utilizar um pseudo-código e um código-fonte previamente escritos que configura o Microcontrolador para controlar o led vermelho e, a partir de alterações neles, implementar a função delay10us(t) para que a cada uma de suas chamadas, há um período de t*ciclo de clock do núcleo do MCU de atraso.

Metodologia:

Inicialmente, o módulo GPIO deve ser configurado corretamente de acordo com seus endereços [2], para cada porta a ser utilizada.
A partir disso, o problema central é instruir o MCU para conseguir executar a função delay10us de forma eficiente. Sabe-se que a frequência de um ciclo de clock do núcleo é de 20.97MHz, o que faz com que cada um dure cerca de 47 ns. Dividindo 10us por esse valor, encontra-se aproximadamente 21. Esse resultado indica que para cada 21 ciclos de clock, passam-se 10us.
Ao saber dessas informações, é nítido que precisa-se ter um contato mais direto ao MCU do que apenas pela linguagem C. Como a utilizaríamos para encontrar a solução?
Por isso, uma alternativa inteligente é ter em um mesmo código instruções em C e também em Assembly (nas partes cruciais relacionadas ao controle de tempo), através de asm [3].

Proposta de Solução

Abaixo estão os pseudo-códigos das duas partes do experimento, bem como a explicação da solução baseada nos códigos fontes em C e (a parte em vermelho que deve ser escrito) em assembly.

Parte 1 – Controle de Leds resultando em luz branca

Pseudo-Código

```
INÍCIO: delay10us
        ENTRADA: número total de iterações
        ENQUANTO número total de iterações é maior que 0
                x←21
                ENQUANTO x é diferente de 0
                        Passa um ciclo do núcleo
                        DECREMENTA x
                DECREMENTA número total de iterações
        FIM ENQUANTO
FIM: delay10us
```

INÍCIO

CONFIGURE registrador de controle SIM_SCGC5[10] de forma a habilitar clock do módulo GPIO
CONFIGURE registrador de controle PORTB_PCR18[8:10] de forma que o pino 18 da porta B sirva GPIO
CONFIGURE registrador de controle PORTE_PCR23[8:10] de forma que o pino 23 da porta E sirva GPIO

CONFIGURE registrador de controle GPIOB_PDDR[18, 19] de forma que os pinos 18 e 19 da porta B sejam de saída
CONFIGURE registrador de controle GPIOE_PDDR[23] de forma que o pino 23 da porta E seja de saída

ENQUANTO VERDADEIRO
ACENDE led vermelho
ESPERA(chama delay10us com número de iterações)
FIM ENQUANTO

FIM

Código Fonte

Sobre o código fonte, haverá apenas a explicitação da função delay10us(t):

```
void delay10us(unsigned int t)
{
    /*! Deve-se utilizar a instrucao NOP 21 vezes
     *! para assim ter 21 ciclos de relogio e consequentemente 10us de delay
     *! Usa-se __asm__ para conseguir programar assembly dentro deste codigo em C
     *! */
    while(t!=0){
        __asm__(
            "mov r0, #21 \n\t"
            "loop: \n\t"
            "nop \n\t"
            "sub r0, r0, #1 \n\t"
            "cmp r0, #0 \n\t"
            "bne loop \n\t"
        );
        t--;
    }
}
```

Acima, pode-se ver que __asm__ [3] é utilizada dentro do loop que garante as t iterações. A partir disso, segue-se o proposto escrito no pseudo-código com loop análogo a rotina enquanto. Porém, o detalhe mais importante está no uso da instrução NOP (A6.7.47 em [4]), que é semelhante a “mov r0, r0”, ou seja, que não tem uma aplicação direta nenhuma, mas que gasta tempo de um ciclo de clock do núcleo.

Testes

Para comprovar a acurácia do experimento, foi utilizado um osciloscópio para identificar a frequência de uma onda em uma das saídas da placa auxiliar para a KL25Z (PTE23). Não foi possível verificar isso diretamente na porta do led vermelho (PTB18), e por conveniência, os sinais foram repassados igualmente para cada uma (motivo para a inicialização da porta E citada no pseudo-código).

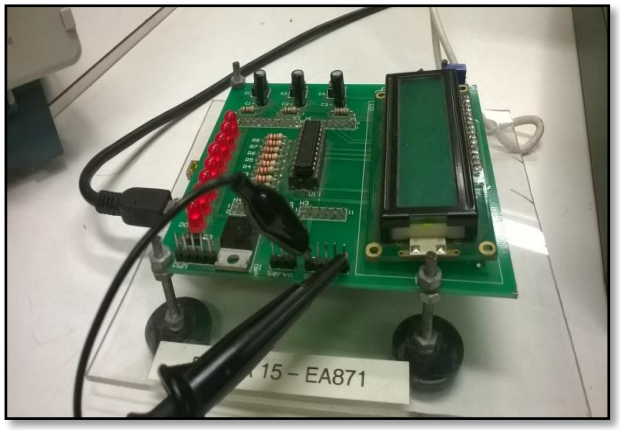

Figura 1. Esquema de medição dos sinais de PTE23 pelo osciloscópio

Enfim, para os testes, foram realizadas as seguintes etapas:

1) Depois de consultar os manuais do MCU e configurar a porta E corretamente, fez-se Build e Debug, e assim o programa foi executado. O led vermelho funcionou corretamente.
2) Ajustou-se t para o valor de 1000, de forma que o resultado fosse uma onda de frequência igual a 100 Hz.
3) Ligou-se o osciloscópio, e colocou-se as pontas de prova nos pinos da placa auxiliar, como mostrado pela figura 1.
4) Como resultado, obteve-se a figura 2, indicando proximidade com o resultado desejado.
5) Ajustou-se t para o valor de 200000, e seguiu-se o passo 3.
6) O resultado foi a figura 3.

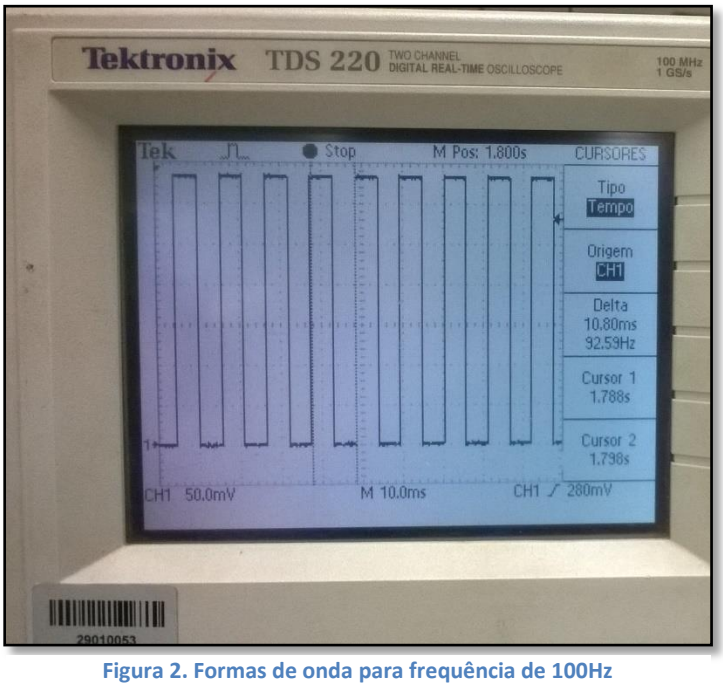

Figura 2. Formas de onda para frequência de 100Hz

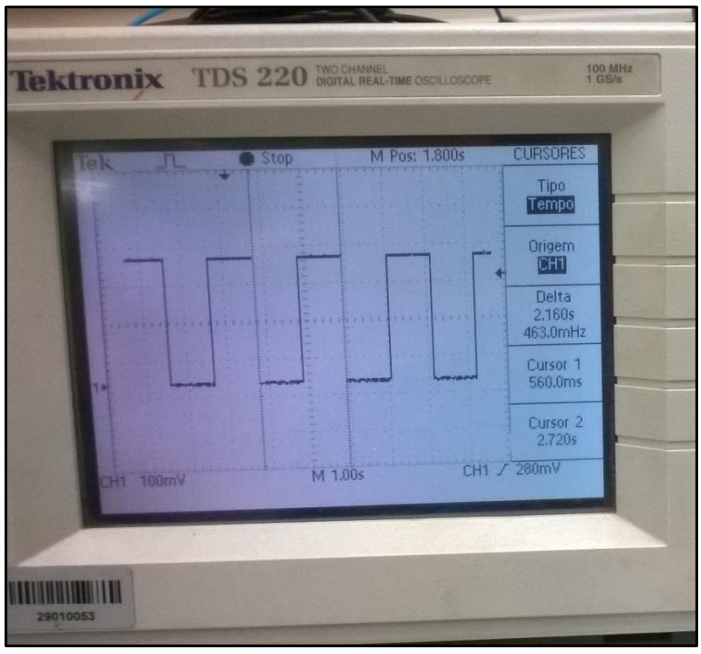

Figura 3. Formas de onda para frequência de 0.5 Hz

Conclusão

O resultado obtido e melhor exemplificado pelas figuras 2 e 3, pode ser considerado próximo, porém não é o ideal. Há algumas possibilidades para essa disparidade entre o esperado e o experimental. Uma delas é o atraso com as subrotinas em C e em Assembly para conseguir passar os 10us, ao se utilizar outras instruções além de NOP. Além disso, o próprio sistema físico da placa não é perfeito, e perdas através dele e do osciloscópio também influenciam no resultado.

Uma alternativa para resolver isso é com o uso de interrupções, mecanismo do MCU muito mais eficaz para esse tipo de manipulação de clock.

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] Extended Asm - Assembler Instructions with C Expression Operands

https://gcc.gnu.org/onlinedocs/gcc/Extended-Asm.html

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

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