Microcontroladores e Microprocessadores
Prática 02: Display de 7segmentos (Catodo e Anodo Comum), TIMER2 e Interrupção TIMER0

Ana Keylla da Fonseca Sousa[1]
Marcela Barbosa Batista[2]
Natália Alves de Araújo[3]

[1]IFCE – Instituto Federal de Educação, Ciência e Tecnologia do Ceará
CEP 60.040-531 Fortaleza (CE)
anakeyllasousa@yahoo.com.br

[2]IFCE – Instituto Federal de Educação, Ciência e Tecnologia do Ceará
CEP 60.040-531 Fortaleza (CE)
marcela_barbosa2801@hotmail.com

[3]IFCE – Instituto Federal de Educação, Ciência e Tecnologia do Ceará
CEP 60.040-531 Fortaleza (CE)
natialves635@hotmail.com

Resumo:

De acordo com a disciplina de microcontroladores e microprocessadores do curso de Engenharia de Computação do IFCE, iremos descrever nesse artigo o funcionamento do display de 7 segmentos (catodo e anodo comum), TIMER2 e Interrupção TIMER0 no PIC18F2550. Essa descrição será necessária para o desenvolvimento da prática 2, usando para a simulação o software PROTEUS-ISIS e para desenvolvimento do código o software MPLAB versão 8.80 e compilador C18 versão 3.4 da Microchip.

Palavra-chave: microcontrolador, PIC18F2550, interrupções TIMER0,TIMER2,Display de 7 segmentos, simulações.

1. Introdução

Um **microcontrolador** (também denominado **MCU**) é um computador num chip, contendo um processador, memória e periféricos de I/O. É um microprocessador que pode ser programado para funções específicas, em contraste com outros microprocessadores de propósito gerais, eles são embarcados no interior de algum outro dispositivo para que possam controlar as funções ou ações do dispositivo. O microcontrolador integra elementos adicionais em sua estrutura interna, como memórias RAM, ROM e EEPROM, dispositivos periféricos como conversores ADC, conversores DAC em alguns casos; e, interfaces I/O, clocks e osciladores.

No segundo tópico iremos abordar o funcionamento do display de 7 segmentos (catodo e anodo comum), como eles funcionam nas diferentes configurações. No terceiro tópico mostraremos o funcionamento TIMER0, a configuração da interrupção do mesmo e como utilizar no projeto. No quarto tópico veremos o funcionamento do TIMER2, cujo evento gerado colabora para o funcionamento do

circuito. A demonstração, montagem e compilação serão no quinto tópico, junto com os conhecimentos extras que utilizamos para concluir o projeto com sucesso. No sexto tópico, nossas considerações finais e a conclusão de aprendizado.

2. Display de 7 segmentos:

O display de sete segmentos é um invólucro com sete leds(ligados numa ordem pré-definida) com formato de segmento, posicionados de modo a possibilitar a formação de números decimais e algumas letras utilizadas no código hexadecimal.

Desse modelo existem variações, envolvendo a estrutura da ligação dos LEDs. Os LEDs presentes no display não estão isolados um dos outros. Para possibilitar uma montagem mais prática, todos os LEDs possuem um terminal em comum.

O display pode ser do tipo **ânodo comum**, ou seja os terminais ânodo de todos os segmentos estão interligados internamente e para o display funcionar, este terminal comum deverá ser ligado em Vcc, enquanto que o segmento para ligar precisa de estar ligados no GND.

Já o display **cátodo comum**, é o contrário, ou seja, o terminal comum, deverá ser ligado ao GND e para ligar o segmento é necessário aplicar Vcc ao terminal.

Como os segmentos são leds, então precisamos de limitar a corrente, para isso devemos usar uma resistência em cada segmento. A corrente utilizada, depende do brilho que queremos do display, normalmente utilizam-se resistências entre 220 e 560 ohms, para uma fonte de 5Volt, o que equivale a uma corrente entre 9mA a 20mA. Não devemos usar valores de resistência muito baixo, pois estaremos a reduzir a vida útil do display, inclusive podemos queimar o segmento. Se for usar um display, teste antes cada segmento, para ter a certeza que não está a usar um display com algum segmento queimado.

O display de sete segmentos, é formado por sete leds, dispostos em forma de oito. Quando se necessita de acender algum número ou caractere, ligam-se os leds correspondentes ao digito, através dos segmentos a, b, c, d, e, f. Como mostra a tabela a seguir:

	Segmentos de saída						
DISPLAY	**a**	**b**	**c**	**d**	**e**	**f**	**g**
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	0	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	0	0	1	1
A	1	1	1	0	1	1	1
B	0	0	1	1	1	1	1
C	1	0	0	1	1	1	0
D	0	1	1	1	1	0	1
E	1	0	0	1	1	1	1
F	1	0	0	0	1	1	1

3. TIMER0

O registo T0CON controla todo o aspectos do funcionamento do módulo, incluindo seleção do prescaler e o que vier a ser lido e escrito.

O módulo TIMER0 possui as seguintes características:

O módulo do Timer 0 possui as seguintes características:

•Contador/Timer de 8 bits;

•Registrador de contagem de escrita e leitura;

•Prescaler (divisor de freqüência) programável;

•Utiliza sinal de clock interno ou externo;

•Gera interrupção quando a contagem muda de FF para 00;

•Permite selecionar o tipo de transição quando usa clock externo (Borda de subida ou descida).

O modo de operação do Timer 0 é definido a partir do bit T0CS (bit 5 do Reg. OPTION). Nomodo timer, o Timer 0 incrementa a cada ciclo de instrução (sem pré-escalonador). Se ocorrer uma escrita no registrador TMR0, o incremento é inibido pelos dois ciclos de instrução seguintes.O modo de contagem é selecionado setando o bit T0CS. No modo de contagem , o Timer

0incrementará a cada subida ou descida do sinal no pino RA4/TOCKL. A borda é definida pelo bitT0SE (bit 4 do Reg. OPTION).

Prescaler

Há um único pré-escalonador que é compartilhado com o WatchDogTimer de forma mutuamenteexcludente (se o Timer 0 usa o pré-escalonador o WatchDog não pode usar e vice versa). Os bitsPSA e PS2:PS0 (bits 3 a 0 do Reg. OPTION) determinam o fator de escalonamento da freqüênciado sinal de clock do Timer 0. Quando atribuído ao Timer 0, todas as instruções que escrevem para o registrador TMR0 (CLRF 1, MOVWF 1, BSF 1,x...) apagarão o contador do pré-escalonador mas não alterarão a sua configuração.

Interrupção do Timer 0

A interrupção TMR0, utilizada para a realização dessa prática, é gerada quando o registrador TMR0 estoura (overflow), conta de FF para00. Este overflow seta o bit T0IF (bit 2 do Reg. INTCON). A interrupção pode ser mascarada resetando o bit T0IE (bit 5 do Reg. INTCON). O bit T0IF deve ser apagado por software (narotina de serviço de interrupção do Timer 0) antes de reabilitar a própria.

TIMER0		
Enable bit	TMR0IE (INTCON<5>)	1-Habilita interrupção 0-Desabilita interrupção
Flag bit	TMR0IF (INTCON<2>)	1- Interrupção ocorreu 0- Interrupção não ocorreu
Priority bit	TMR0IP (INTCON2<2>)	1-High priority 0-Low priority

4. TIMER2

O Timer 2 é um timer de 8 bits com um pré-escalonador e pós-escalonador. O mesmo pode ser usado como a base de tempo no modo PWM dos módulos CCP.

O TMR2 é um registrador de escrita e leitura e é apagado em qualquer evento de reset. A entrada de clock (Fosc/4) possui uma opção de escalonamento de 1:1, 1:4 ou 1:16, selecionada pelos bits de controle T2CKPS1:T2CKPS0 (bits 1 e 0 do Reg. T2CON), que faz com que o sinal de clock do timer 2 seja dividido, respectivamente, por 1, 4 e 16.

O módulo do Timer 2 possui um registrador de período de 8 bits PR2, que indica o valor máximo que pode ser atingido pelo TMR2, assim o timer 2 incrementa de 00h até atingir PR2 e então reseta para 00h no ciclo de incremento seguinte. PR2 é um registrador de escrita e leitura e é inicializado com FFh após qualquer RESET. O sinal de saída resultante da comparação do PR2 com o TMR2 passa por um pós-escalonador (escalas de 1:1, 1:4 e 1:16) para gerar a interrupção TMR2 (quando ocorre, o bit 1 (TMR2IF) doReg. PIR1 é setado), ou seja, quando a interrupção está habilitada a mesma ocorrerá após 1, 2, 3,4, ... ou 16 vezes em que o valor do contador do timer 2 (TMR2) atingir o valor gravado no registrador PR2.

O Timer 2 pode ser desligado resetando o bit de controle TMR2ON (bit 2 do registrador T2CON).

Prescaler e Postscale do TIMER 2

Os contadores do pré-escalonador e do pós-escalonador são apagados quando algum dos eventosabaixo ocorre:

•Uma escrita para o registrador TMR2;
•Uma escrita para o registrador T2CON;
•Qualquer reset do dispositivo (POR, MCLR reset, WDT reset ou BOR).

O Timer 2 não é apagado quando ocorre uma escrita para registrador de configuração do timer 2T2CON.

5. Prática 2 – Simulação software PROTEUS-ISIS

De acordo com o que foi solicitado na prática 2, foi montado no software PROTEUS-ISIS um circuito semelhante ao ilustrado na proposta do projeto, verificamos o acionamento de todos os componentes corretamente, a fim de testar o funcionamento do firmware e a interface gráfica de simulação.

1 - Hardware

Na IDE do MPLAB, modificamos o código do evento TIMER2, pelo seguinte trecho de código mostrado abaixo:

```
/**
*@fn TIMER2_EVENT_handler()
*@brief TIMER 2 Event Handler
*@param \a void
*@return \a void
*/
void TIMER2_EVENT_handler(void)
{
      T2_Counter++;
      if(T2_Counter >= 31)
     {
        //atualiza estado do LED1
         T2_Counter = 0;
         LED1 ^= 1;
         PORTB++;
     }
}
```

Observa-se que com a execução desse código, a cada mudança de estado do LED, a PORTB é incrementada, enviando para sua saída uma representação binária, fazendo o display de 7 segmentos atuar como um contador binário, sendo sua leitura feita da seguinte forma: gfedcba, acendendo apenas o led q representa 1 no binário em questão. Por exemplo, na primeira interação o contador binário estará em 0000000, assim o display estará apagado, já na segunda interação quando a PORTB é incrementado em 1, será enviado para a porta o seguinte binário 0000001, acendo, assim, apenas o led do display representado por ‘a’, e assim sucessivamente para a contagem de todos os binários.

2 – Contador Hexadecimal

Como pedido neste momento da prática, foi alterado o código, do evento do TIMER2, para que ele se tornar-se um contador hexadecimal.

Para isso foi adicionado ao código o seguinte trecho:

```
PORTB = VETOR[P];
      P++;
      if(P==16)
            P=0;
```

Onde VETOR[P] é um vetor de 16 posições que possui a combinação binária de todos os caracteres que aparecem no contador hexadecimal.

Após a inserção desse trecho o evento TIMER2 passou a ter o seguinte código:

```
/**
*@fn TIMER2_EVENT_handler()
*@brief TIMER 2 Event Handler
*@param \a void
*@return \a void
*/
void TIMER2_EVENT_handler(void)
{
      T2_Counter++;
      if(T2_Counter >= 31)
     {
        //atualiza estado do LED1
         T2_Counter = 0;
         LED1 ^= 1;
         PORTB = VETOR[P];
         P++;
         if(P==16)
                P=0;
    }
}
```

3 – Display Anodo Comum

Ao substituímos o display de catodo comum por um de anodo comum, observamos que para os mesmo evento TIMER2, feito no procedimento 1, ele enviará para a porta a negação do binário enviado, porém continua a ser um contador binário, só que agora o

número 1 do binário será representado pelo led apagado e o 0 será representado pelo led acesso. O mesmo ocorre com o procedimento 2, ele continuará a ser um contador binário, porém negando a saída, logo o caractere que deveria aparecer no display, será aquele formado pelos leds apagados.

4 – Utilizando Múltiplos Displays 1

Montando o display proposto no procedimento 4, ao executarmos a função obtida no procedimento 2, observa- se que o display que possui sua chave abaixada será o contador hexadecimal, porém os dois displays não podem funcionar juntos, ou seja, se os dois displays estiverem com suas respectivas chaves abaixadas, nenhum deles irá funcionar, mantendo – se apagados até uma das chaves seja levantada.

5 – Utilizando Múltiplos Displays 2

No procedimento 5 foi criado um nove evento que é gerado a parti da interrupção do TIMER0, ele realiza a comutação entre os displays. Foi ainda testado nesse momento as diferentes freqüências de interrupção do TIMER0 como mostra a tabela abaixo:

Frequência	Tempo(s)
1Hz	0,0039
10Hz	3,932
100Hz	3,921
400Hz	9,803

Observou - se quanto menor a freqüência mais rápida será chamado o evento de comutação dos displays.

6 – Contador Composto

A partir do evento TIMER2, foi escrito um novo evento, esse evento trata – se de um contador de 00 a 99. O evento escrito utiliza-se ainda do evento de interrupção do TIMER0 para realizar a comutação entre os displays, sendo o display 1 setado para o próximo número somente após o display 2 ter contado de 0 a 9.

Abaixo, mostra – se o evento escrito para gerar o contador de 00 a 99.

```
void TIMER0_CONTADOR(void)
{
        T0_Counter++;
        if(T0_Counter >= 10)
        {
                BUTTOND2 = 1;
                BUTTOND1 = 0;
                PORTB = VETOR[P];

                T0_Counter++;
        }
                if(T0_Counter>=20){
                    T0_Counter=0;
                    BUTTOND2 = 0;
                    BUTTOND1 = 1;
                    PORTB = VETOR[dez];
                    P++;
                }
                if(P==10){
                        P=0;
                        dez++;
                }
                if(dez==10)
                        dez=0;

}
```

7 – Múltiplos Displays Anodo Comum

Assim como ocorrido no procedimento 3, os números enviados pelo contador serão representados pelos leds apagados.

No circuito houve o acréscimo de um transistor de controle do anodo deverá ser modificado, por um que suporte uma corrente de coletor/emissor superior a corrente total consumida pelo display. O “beep” (*buzzer*) é do tipo piezo elétrico, com faixa de alimentação de 12 a 30V.

6. Considerações Finais

Nesse projeto aprendemos a definição e funcionamento do TIMER2, sistema de interrupção do TIMER0 e o funcionamento do displays de 7 segmentos, ainda aprimoramos de lógica de programação de microcontroladores, devido a implementação da prática em questão.

7. Referências

- Data sheet Microchip - PIC18F2455/2550/4455/4550

- http://pessoal.utfpr.edu.br/amauriassef/arquivos/Timers.pdf

- http://pt.scribd.com/doc/22751176/MODULO-8