FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

Aula 15
Timers

Microcontroladores PIC18 – Programação em C

Prof. Ítalo Jáder Loiola Batista
Universidade de Fortaleza - UNIFOR
Centro de Ciências Tecnológicas - CCT
E-mail: italoloiola@unifor.br

Jan / 2011

Roteiro

- Funções de *delay*
- Timers e Contadores
- Contagem do Tempo no PIC
- Código-fonte

Funções de *Delay*

- Funções de *delay* nativas do MPLAP C18, utilizadas na medição de tempo por software;
- A medição é feita pela contagem de ciclos de instrução (TCI) que depende da freqüência do oscilador do clock;
- TCI = 1 / (TOSC/4)
- Ex.: Suponha FOSC = 8Mhz.
  - **Delay10TCY(20); //** Gera delay de 100us
  - 10 x 20 x 500ns = 100us

Função	Descrição
Delay1TCY	*Delay* de um ciclo de instrução
Delay10TCYx	*Delay* de 10 ciclo de instrução
Delay100TCYx	*Delay* de 100 ciclo de instrução
Delay1kTCYx	*Delay* de 1.000 ciclo de instrução
Delay10kTCYx	*Delay* de 10.000 ciclo de instrução

Temporizador

- Temporizador é um contador que é incrementado a partir de um pulso de relógio externo ou de um oscilador interno do uC;
- Quando o conteúdo do temporizador atinge um valor máximo:
  - Ocorre o evento denominado overflow;
  - O conteúdo do temporizador é resetado;

TIMERS do PIC18F4520

- Timers contam tempo;
- Contadores contam eventos;
- O PIC18F4520 possui 3 timers/contadores e um 1 timer com características diferentes de funcionamento:
  - TIMER 0;
  - TIMER 1;
  - TIMER 2.
  - TIMER 3

TIMERS do PIC18F4520

- O que varia de um para o outro?
  - Limite de contagem;
  - Modo de operação (como contador/timer);
  - Tipo de incremento;
  - *Prescales e Postscales;*
  - A geração de interrupções;
  - Os periféricos a eles associados.

Registradores SFR do TIMER

FFFh	TOSU	FDFh	INDF2(1)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(1)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(1)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(1)	FBCh	CCPR2H	F9Ch	—(2)
FFBh	PCLATU	FDBh	PLUSW2(1)	FBBh	CCPR2L	F9Bh	OSCTUNE
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—(2)
FF9h	PCL	FD9h	FSR2L	FB9h	—(2)	F99h	—(2)
FF8h	TBLPTRU	FD8h	STATUS	FB8h	BAUDCON	F98h	—(2)
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	PWM1CON(3)	F97h	—(2)
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCP1AS(3)	F96h	TRISE(3)
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD(3)
FF4h	PRODH	FD4h	—(2)	FB4h	CMCON	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	HLVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—(2)
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRGH	F90h	—(2)
FEFh	INDF0(1)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—(2)
FEEh	POSTINC0(1)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—(2)
FEDh	POSTDEC0(1)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(3)
FECh	PREINC0(1)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(3)
FEBh	PLUSW0(1)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—(2)	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—(2)
FE7h	INDF1(1)	FC7h	SSPSTAT	FA7h	EECON2(1)	F87h	—(2)
FE6h	POSTINC1(1)	FC6h	SSPCON1	FA6h	EECON1	F86h	—(2)
FE5h	POSTDEC1(1)	FC5h	SSPCON2	FA5h	—(2)	F85h	—(2)
FE4h	PREINC1(1)	FC4h	ADRESH	FA4h	—(2)	F84h	PORTE(3)
FE3h	PLUSW1(1)	FC3h	ADRESL	FA3h	—(2)	F83h	PORTD(3)
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

TIMER 0

- TIMER 0 (registrador TMR0)
- Temporizador / Contador de 8 ou 16 bits.
- Pode ser lido e escrito, ou seja, permite ser inicializado.
- Funcionamento: Incremental (somente).
- Incremento de 2 formas distintas:
  - Contador: A cada transição (TOCKI: pulso de clock externo).
  - Timer: A cada ciclo de máquina.
- TMR0 muda de estado, segundo o valor do *Prescaler (PS).*
  - Prescaler permite um recurso de contagem além do limite do registrador do timer TMR0.
  - Ex.: PS configurado como 1:4. São necessários 4 ciclos de máquinas ou 4 pulsos externos, para que o TMR seja incrementado de 1 unidade.
  - O PS é de 8 bits, mas não está disponível para leitura nem escrita!

TIMER 0

- Toda vez que se escreve em TMR0, PS é zerado!
- Para a utilização do PS em TMR0:
  - 1. Configurar PSA (T0CON<3>)
    - PSA = 1: Prescale desativado.
    - PSA = 0; Prescale ativado.
  - 5. Configurar o fator do PS em:
    - T0PS2:T0PS0 (T0CON<2:0>)
- Bit de estouro
  - TMR0IF (INTCON<2>), será setado;

TIMER 0: Configuração do código-fonte exemplo para um timer de 4ms

- Timer configurado em 8 bits / TMR0L = 256;
- Prescaler em 32 bits;
- Freqüência de clock: 8MHz;
- Ciclo de instrução = 0,5us;
- O TMR0L será incrementado a cada 16us;
- N_incrementos = 256 – valor_inicial
- Inicializado com o valor 5;
- 256 – 6 = 250 vezes até o estouro;
- Tempo_estouro = 250 x 16us = 4ms

TIMER 0: Configurado como temporizador/contador de 16bits

- Exemplo: Configurar o TMR0 (8 bits) para que gere interrupções a cada 1 segundo.
  - Vamos considerar que o CLK da CPU = 4 MHz.
    - O clock interno será de 1 MHz. Logo, Tcpu = 1 us, ou seja, a cada 1us TMR0 avança uma unidade.
  - Como queremos gerar interrupções a cada 1 segundo, a freqüência de geração dessas interrupções deverá ser de 1 Hz.
  - Entretanto o clock interno funciona em uma freqüência 1.000.000 maior que 1Hz.
  - Usar o TMR0 sem o recurso do PRESCALER, necessitaria contar 1.000.000 / 256 = 3906,25 interrupções.

TIMER 0: Configurado como temporizador/contador de 16bits

- Exemplo: Configurar o TMR0 (8 bits) para que gere interrupções a cada 1 segundo:
  - Vamos considerar que o CLK da CPU = 4 MHz. Logo o CLK interno é de 1 MHz.
  - Se o PRESCALER estiver programado em 1:64, a freqüência de entrada no TMR0 será de 1 MHz : 64 = 15625 Hz.
  - Se programarmos o TMR0 para dividir esse sinal 15625 por 125, teremos um sinal de saída de 125 Hz, para isso, basta carregá-lo a cada estouro de contagem com o valor: 256 (28) – 125 = 131.

TIMER 0:
Configurado como temporizador

1. Ligar o TMR0
   - Setar o bit: TMR0ON (T0CON<7>);
2. Para interromper a contagem a qualquer momento
   - Basta apagar o bit: TMR0ON (T0CON<7>);

Registrador do Timer 0 – T0CON

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-Bit/16-Bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is not assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

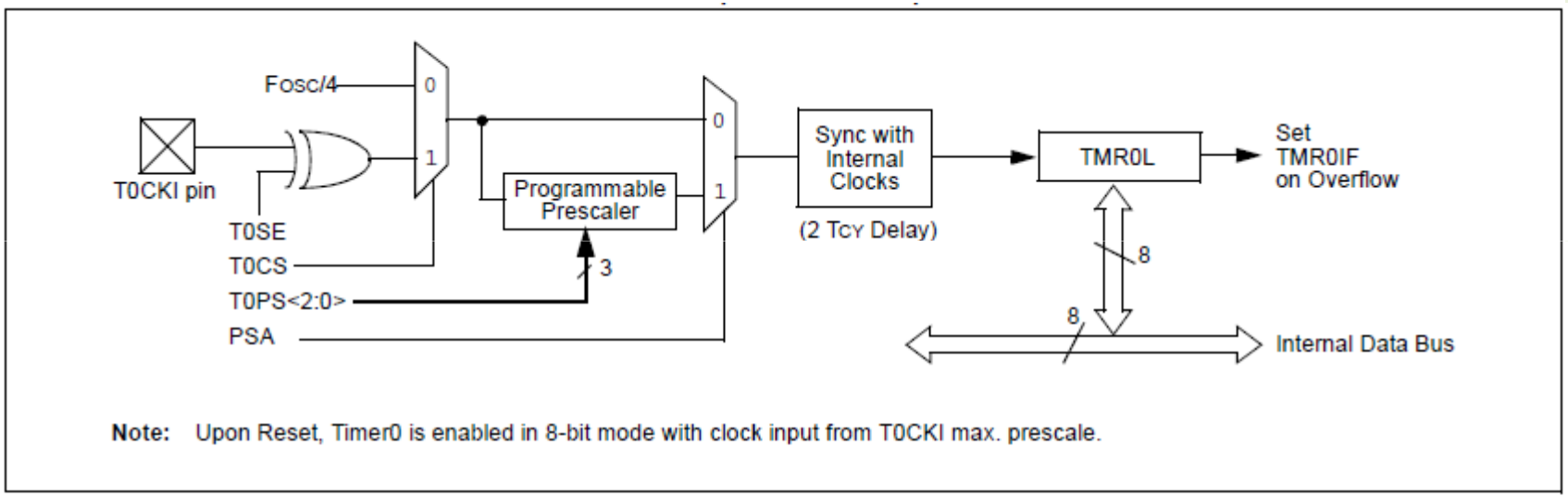
bit 2-0 **T0PS<2:0>**: Timer0 Prescaler Select bits
111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8 Prescale value
001 = 1:4 Prescale value
000 = 1:2 Prescale value

Registradores Associados - Timer 0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TMR0L	Timer0 Register Low Byte								50
TMR0H	Timer0 Register High Byte								50
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	50
TRISA	RA7(1)	RA6(1)	RA5	RA4	RA3	RA2	RA1	RA0	52

TIMER 0 - Diagrama de Blocos

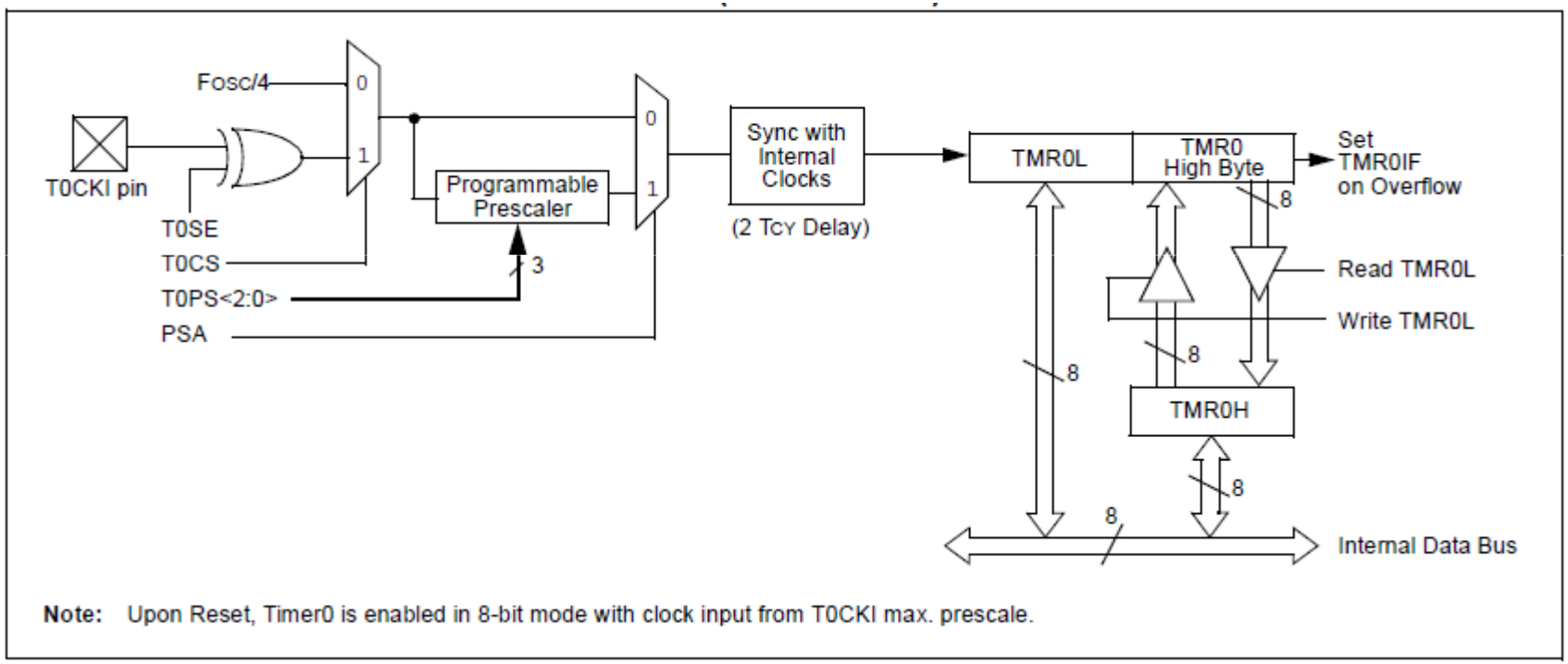
- Modo: 8 bits



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

TIMER 0 - Diagrama de Blocos

- Modo: 16 bits



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

TIMER 1

- Temporizador / Contador de 8 ou 16 bits.
  - Definido pelo bit TMR1CS (T1CON<1>).
- Podem ser lidos e escritos pelo programador.
- Origem do sinal: interno ou externo.
- Para operar com sinais externos, bit T1OCEN (T1CON<6>) ;
  - Cristal nos pinos RC0 (T1OSO) e RC1 (T1OSI).
  - Sinal pulsado no pino RCO (T1CKI).
- Para operar com sinais internos, bit T1RUN (T1CON<3>) ;
  - Incremento interno através dos ciclos de máquina.
- Possuí um Prescaler para configurar o incremento.

TIMER 1

- Possuí um sincronismo do CLK externo com o CLK interno
  - T1SYNC = 1 (sincronismo desligado);
  - T1SYNC = 0 (sincronismo ligado).
- Quando o sincronismo esta desligado, permite que a contagem continue mesmo que o PIC esteja em modo SLEEP.
- Para ler os registradores do TIMER1 (TMR1H e TMR1L):
  - Modo convencional: pare a contagem e leia.
  - Modo alternativo: leia TMR1H, guarde em uma variável temporária, depois leia TMR1L. Depois da leitura de TMR1L compare TMR1H com o valor da variável temporária. Caso afirmativo: OK. Caso negativo: faça isso novamente.
- A escrita reseta o Prescaler.

Registrador do Timer 1 – T1CON

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7 **RD16:** 16-Bit Read/Write Mode Enable bit
1 = Enables register read/write of TImer1 in one 16-bit operation
0 = Enables register read/write of Timer1 in two 8-bit operations

bit 6 **T1RUN:** Timer1 System Clock Status bit
1 = Device clock is derived from Timer1 oscillator
0 = Device clock is derived from another source

bit 5-4 **T1CKPS<1:0>:** Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable bit
1 = Timer1 oscillator is enabled
0 = Timer1 oscillator is shut off
The oscillator inverter and feedback resistor are turned off to eliminate power drain.

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Select bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T13CKI (on the rising edge)
0 = Internal clock (Fosc/4)

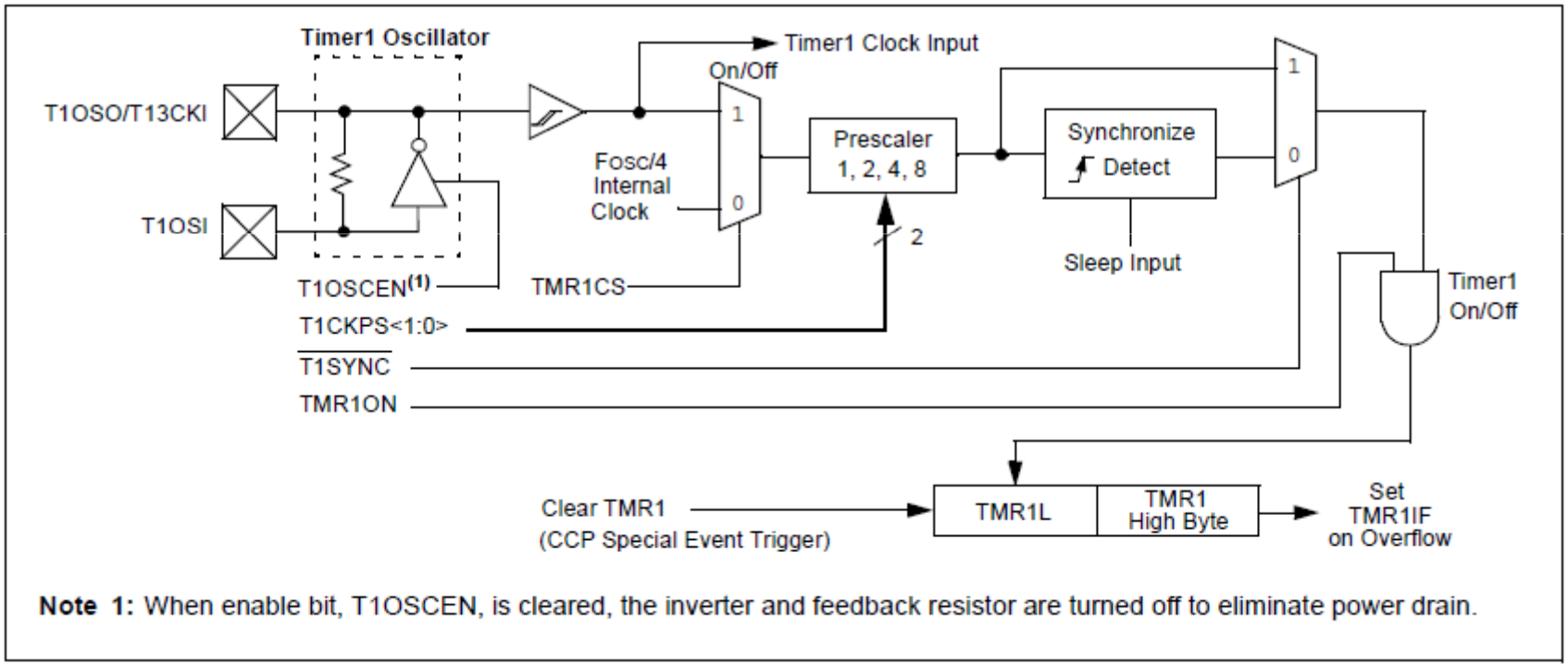
bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Registradores Associados - Timer 1

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
TMR1L	Timer1 Register Low Byte								50
TMR1H	Timer1 Register High Byte								50
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	50

TIMER 1 - Diagrama de Blocos

- Modo: 16 bits



Note 1: When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

TIMER 2

- Timer de 8 bits:
  - Pré-escalonador
  - Pós-escalonador
- Pode ser usado como a base de tempo no modo PWM dos módulos CCP;

TIMER 2

- Temporizador de 8 bits.
- Incremento somente relacionado com o CLK interno.
- Possuí um *Prescale;*
- Possuí um *Postscale;*
- Diferente dos 2 Timers anteriores:
  - Não conta de 0 até 255.
  - Conta do conteúdo do registrador PR2 até 255.
- O *Postscale define o número de vezes que o TMR2 irá* contar.
- O *Postscale é incrementado sempre que TMR2 = PR2.*
- Quando o *Postscale terminar a INT referente ao Timer 2* será gerada.
- O ajuste do *Prescale é feito com T2CON.*
- O ajuste do *Postscale é feito com o T2CON.*

TIMER 2

- O Prescaler e o Postscaler serão zerados:
  - Quando escrever em TMR2;
  - Quando escrever em T2CON;
  - Quando houver um RESET (diferente dos outros Timers 1 e 0).
- Possuí chave específica para habilitar ou desabilitar o incremento, bit TMR2ON.

Registrador do Timer 2 – T2CON

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **T2OUTPS<3:0>:** Timer2 Output Postscale Select bits

0000 = 1:1 Postscale
0001 = 1:2 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON:** Timer2 On bit

1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS<1:0>:** Timer2 Clock Prescale Select bits

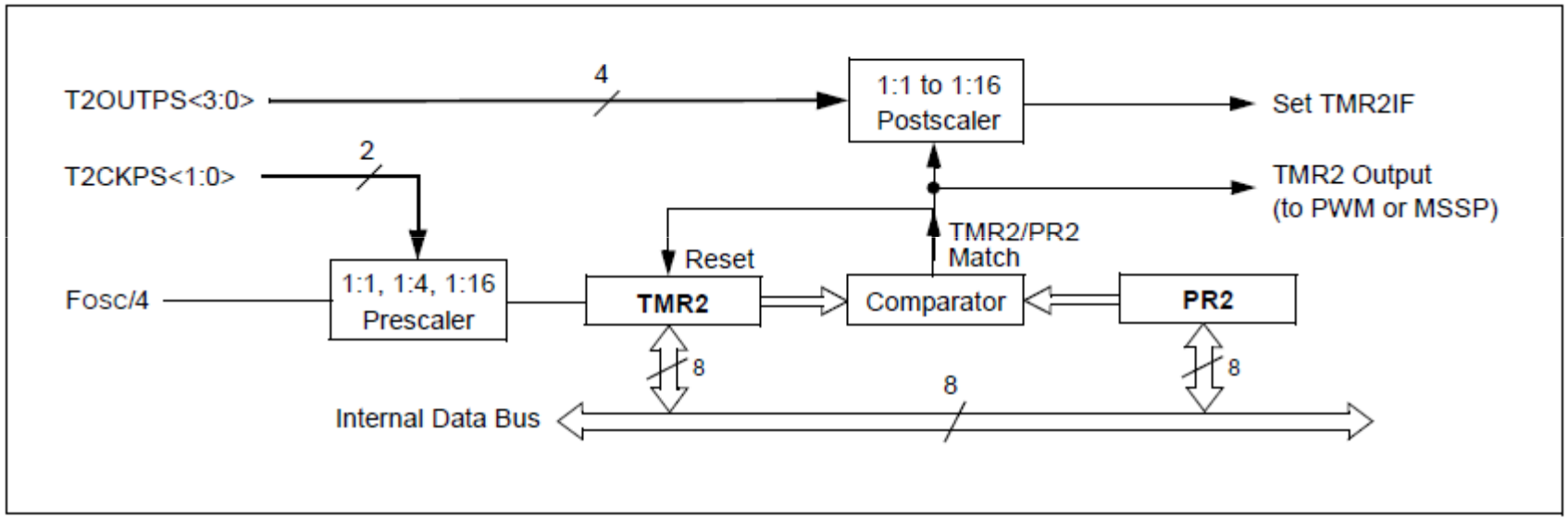
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

Registradores Associados - Timer 2

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
TMR2	Timer2 Register								50
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	50
PR2	Timer2 Period Register								50

TIMER 2 - Diagrama de Blocos

- Modo: 16 bits



TIMER 3

- O módulo Timer3 é muito semelhante ao módulo Timer1;
- Ele pode operar como Temporizador/Contador de 8 ou 16 bits;
- O Timer3 pode ser usado como origem de clock para o periférico CCPx;
- O registrardor T3CON é o responsável pela configuração do Timer3;

Registrador do Timer 3 – T3CON

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
bit 7							bit 0

bit 7 **RD16:** 16-Bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer3 in one 16-bit operation
0 = Enables register read/write of Timer3 in two 8-bit operations

bit 6,3 **T3CCP<2:1>:** Timer3 and Timer1 to CCPx Enable bits
1x = Timer3 is the capture/compare clock source for the CCP modules
01 = Timer3 is the capture/compare clock source for CCP2;
Timer1 is the capture/compare clock source for CCP1
00 = Timer1 is the capture/compare clock source for the CCP modules

bit 5-4 **T3CKPS<1:0>**: Timer3 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 2 **$\overline{\text{T3SYNC}}$:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the device clock comes from Timer1/Timer3.)
When TMR3CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR3CS = 0:
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit
1 = External clock input from Timer1 oscillator or T13CKI (on the rising edg
0 = Internal clock (Fosc/4)

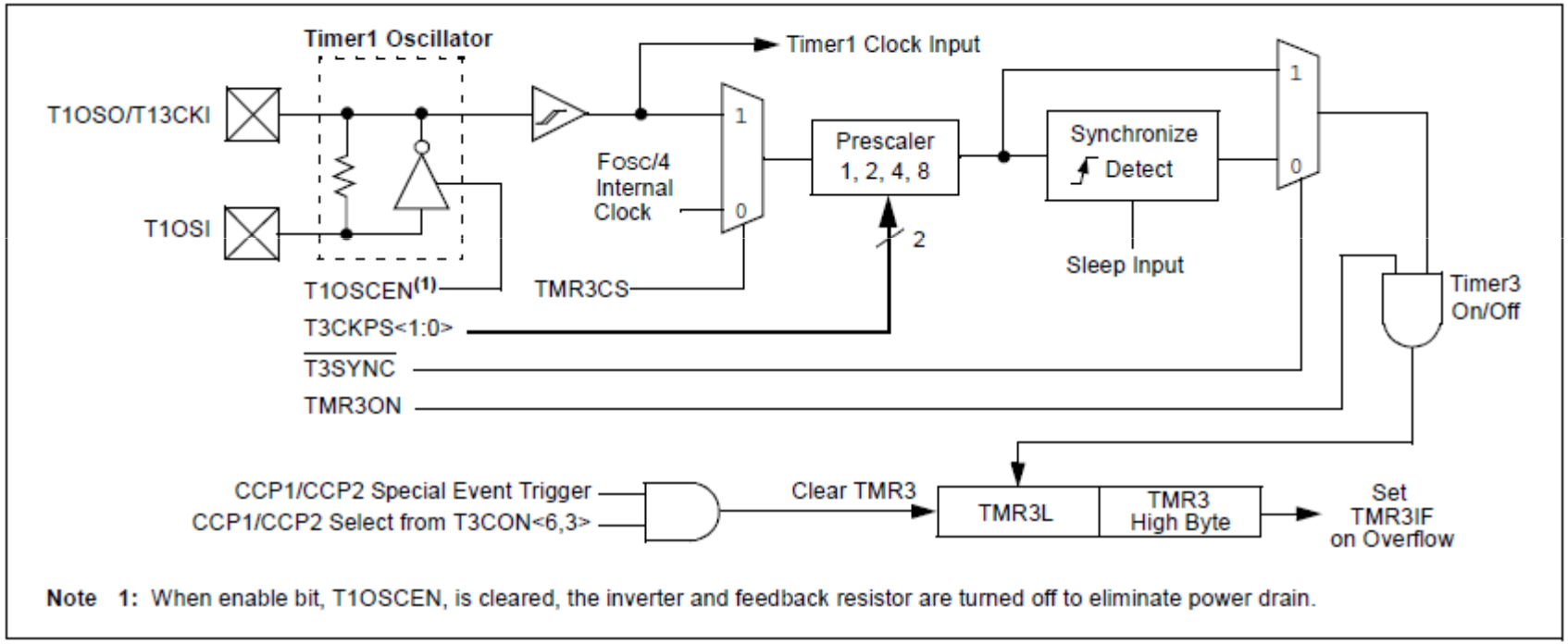
bit 0 **TMR3ON:** Timer3 On bit
1 = Enables Timer3
0 = Stops Timer3

Registradores Associados - Timer 3

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
PIR2	OSCFIF	CMIF	—	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	52
PIE2	OSCFIE	CMIE	—	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	52
IPR2	OSCFIP	CMIP	—	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	52
TMR3L	Timer3 Register Low Byte								51
TMR3H	Timer3 Register High Byte								51
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	50
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	51

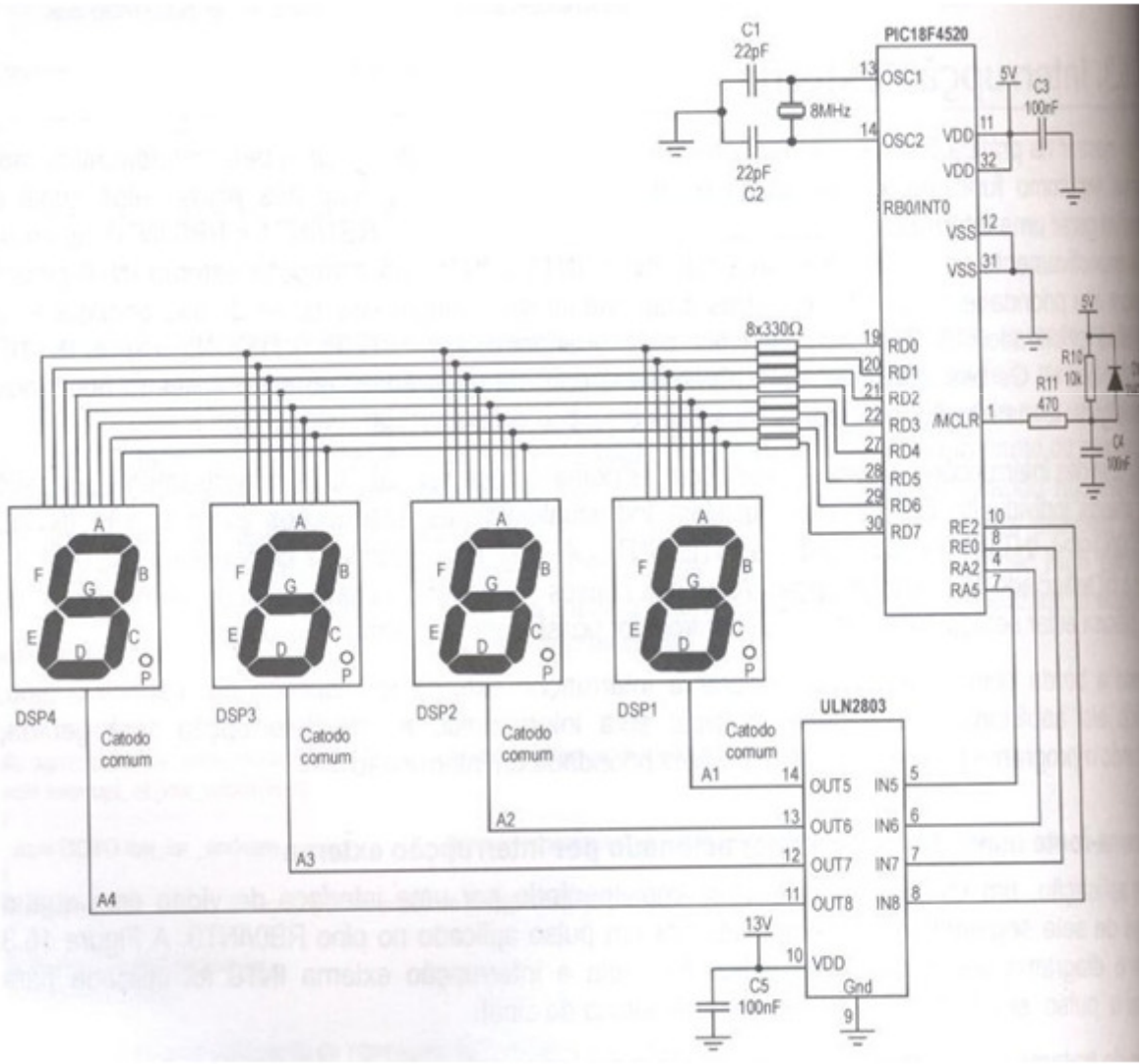
TIMER 3 - Diagrama de Blocos

- Modo: 16 bits



Note 1: When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

Esquema Elétrico



Timer 0 / Código-fonte

- DSP_7Seg_x4.h

  Arquivo cabeçalho com as definições dos pinos nos quais serão conectados os pinos do display.

- DSP_7Seg_x4.c

  Arquivo que compõe a biblioteca que contém a função que fará a atualização do display;

- Main_34.c

  Arquivo principal responsável por realizar a aplicação de um contador de 0 a 9.999 utilizando uma interface de vídeo com quatro displays multiplexados, onde a temporização é implementada pelo módulo temporizador do TIMER 0;

DSP_7Seg_x4.h

Display Multiplexado / Código-fonte

Esse identificador impede que a definição a seguir seja duplicada se o arquivo cabeçalho foi incluído em outro arquivo-fonte associado ao projeto.

```
8  #ifndef __DSP_7SEGx4_H
9  #define __DSP_7SEGx4_H
10 //*********************************************************************
11 #include <p18cxxx.h>      //diretiva de compilação
12 //*********************************************************************
13 //definições
14 #define DSP_1    PORTAbits.RA5
15 #define DSP_2    PORTAbits.RA2
16 #define DSP_3    PORTEbits.RE0
17 #define DSP_4    PORTEbits.RE2
18 #define L_DADOS TRISD
19 #define DIR_A1    TRISAbits.TRISA5
20 #define DIR_A2    TRISAbits.TRISA2
21 #define DIR_A3    TRISEbits.TRISE0
22 #define DIR_A4    TRISEbits.TRISE2
23 //*********************************************************************
24 void Aciona_DPS_7_seg (unsigned char Dsp4, unsigned char Dsp3, unsigned char Dsp2, unsigned char Dsp1);
25 #endif
```

DSP_7Seg_x4.c

Display Multiplexado / Código-fonte

```
8   #include <p18cxxx.h>          //diretiva de compilação
9   #include "DSP_7Seg_x4.h"       //diretiva de compilação
10  //*********************************************************************
11  void Aciona_DPS_7_seg (unsigned char Dsp4, unsigned char Dsp3, unsigned char Dsp2, unsigned c
12  {
13  static unsigned char Atual_Dsp = 1;          //declaração  de variável local static inicializa
14  const char tabela[] = {
15                      0x3F,    // número 0
16                      0x06,    // número 1
17                      0x5B,    // número 2
18                      0x4F,    // número 3
19                      0x66,    // número 4
20                      0x6D,    // número 5
21                      0x7C,    // número 6
22                      0x07,    // número 7
23                      0x7F,    // número 8
24                      0x67,    // número 9
25                      0x00    //apaga display
26                     };
27  //*********
28  //configuração dos pinos
29     L_DADOS = 0x00;          //configura pinos das linhas de dados como saída
30     ADCON1 = 0x0F;          //configura Port A e Port E como pinos digitais
31     DIR_A1 = 0;             //configura linha de endereço A1 como saída
32     DIR_A2 = 0;             //configura linha de endereço A2 como saída
33     DIR_A3 = 0;             //configura linha de endereço A3 como saída
34     DIR_A4 = 0;             //configura linha de endereço A4 como saída
35  //*********************************************************************
```

Embora a função Aciona_DSP_7_seg() receba os valores de cada display, apenas um display é atualizado em cada chamada.

DSP_7Seg_x4.c

Display Multiplexado / Código-fonte

```
36  //atualiza display
37  if(Atual_Dsp==1)            //atualizar display 1
38      {
39            PORTD = tabela[Dsp1];    //atualiza display 1
40            DSP_1 = 1;               //ativa linha A1
41            DSP_2 = 0;               //desativa linha A2
42            DSP_3 = 0;               //desativa linha A3
43            DSP_4 = 0;               //desativa linha A4
44         Atual_Dsp = 2;           //aponta endereço para o próximo display
45      }
46      else    if(Atual_Dsp==2)    //atualizar display 2
47      {
48            PORTD = tabela[Dsp2];    //atualiza display 2
49            DSP_1 = 0;               //desativa linha A1
50         DSP_2 = 1;               //ativa linha A2
51          DSP_3 = 0;               //desativa linha A3
52          DSP_4 = 0;               //desativa linha A4
53         Atual_Dsp = 3;           //aponta endereço para o próximo display
54      }
```

Main_31.c

Display Multiplexado / Código-fonte

```
55      else   if(Atual_Dsp==3)   //atualizar display 3
56     {
57        PORTD = tabela[Dsp3];   //atualiza display 3
58        DSP_1 = 0;             //desativa linha A1
59        DSP_2 = 0;             //desativa linha A2
60        DSP_3 = 1;             //ativa linha A3
61        DSP_4 = 0;             //desativa linha A4
62        Atual_Dsp = 4;          //aponta endereço para o próximo display
63     }
64      else   if(Atual_Dsp==4)   //atualizar display 4
65     {
66        PORTD = tabela[Dsp4];   //atualiza display 4
67        DSP_1 = 0;             //desativa linha A1
68        DSP_2 = 0;             //desativa linha A2
69        DSP_3 = 0;             //desativa linha A3
70        DSP_4 = 1;             //ativa linha A4
71        Atual_Dsp = 1;          //aponta endereço para o próximo display
72     }
73  //*************************************************************************
74  }
```

Main_34.c

Timer 0 / Código-fonte

```
8   #include <p18f4520.h>          //diretiva de compilação
9   #include <delays.h>             //diretiva de compilação
10  #include "DSP_7Seg_x4.h"       //diretiva de compilação
11  //*****************************************************************
12  //protótipos de funções
13  void Inic_Regs (void);
14  void high_isr (void);
15  //*****************************************************************
16  //variáveis globais
17  volatile unsigned char Dsp1=0;   //declaração  de variável global inicializada
18  volatile unsigned char Dsp2=0;   //declaração  de variável global inicializada
19  volatile unsigned char Dsp3=0;   //declaração  de variável global inicializada
20  volatile unsigned char Dsp4=0;   //declaração  de variável global inicializada
21  int x = 0;                      //declaração  de variável global inicializada
22  //*****************************************************************/
23  //vetor de interrupção de alta prioridade
24  #pragma code high_vector=0x08
25  void interrupt_at_high_vector(void)
26  {
27      _asm GOTO high_isr _endasm      //desvia programa para rotina de tratamento da
28  }
29  #pragma code
30  //*****************************
```

Observe que as variáveis Dsp1, Dsp2, Dsp3, Dsp4 foram declaradas como volatile.

Istoé feito por recomendação do fabricante do MPLAB C18 porque elas são manipuladas dentro e fora da rotina de tratamento de interrupção ;

Main_34.c

Timer 0 / Código-fonte

```
24  #pragma code high_vector=0x08
25  void interrupt_at_high_vector(void)
26  {
27      _asm GOTO high_isr _endasm      //desvia programa para rotina de tratamento d
28  }
29  #pragma code
30  //*****************************
31  //Rotina de tratamento de interrupção (ISR)
32  #pragma interrupt high_isr
33  void high_isr (void)
34  {
35     if(!INTCONbits.TMR0IF);
36     else
37     {
38        INTCONbits.TMR0IF=0;                  //interrupção de estouro de TMR0?
39        TMR0L = 5;                           //inicializa TMR0
40        Aciona_DPS_7_seg (Dsp4, Dsp3, Dsp2, Dsp1);   //chamada à função: atualizar
41        if(!(x==250))x++;                     //se x diferente de 250, incrementa
```

Main_34.c

Timer 0 / Código-fonte

```
42        else                                  //senão, incrementa contador
43        {
44           x=0;                              //x=0
45           Dsp1+=1;                           //incrementa unidade
46           if(Dsp1==10)                        //unidade estourou?
47           {
48           Dsp1=0;                             //sim, zera unidade
49           Dsp2+=1;                           //incrementa dezena
50           }
51        if(Dsp2==10)                           //dezena estourou?
52        {
53           Dsp2=0;                             //sim, zera dezena
54           Dsp3+=1;                           //incrementa centena
55        }
56        if(Dsp3==10)                           //centena estourou?
57        {
58           Dsp3=0;                             //sim, zera centena
59           Dsp4+=1;                           //incrementa unidade de milhar
60           }
61           if(Dsp4==10)                        //unidade de milhar estourou?
62           {
63              Dsp4=Dsp3=Dsp2=Dsp1=0;              //sim,zera contador
64           }
65        }
66     }
67  }
```

Main_34.c

Timer 0 / Código-fonte

```
69  void main(void)                                  //função main
70  {
71     Inic_Regs ();                                 //configurar SFRs
72     while(1);                                     //loop infinito
73  }
74  /*********************************************************************
75  Esta funcao inicializa os registradores SFRs.*/
76   void Inic_Regs (void)
77  {
78     TRISA = 0x00;                  //PORTA saída
79     TRISB = 0x00;                  //PORTB saída
80     TRISC = 0x00;                  //PORTC saída
81     TRISD = 0x00;                  //PORTD saída
82     TRISE = 0x00;                  //PORTE saída
83     ADCON1 = 0x0F;                  //configura pinos dos PORTA e PORTE como digitai
84     PORTA = 0;                     //limpa PORTA
85     PORTB = 0;                     //limpa PORTB
86     PORTC = 0;                     //limpa PORTC
87     PORTD = 0x00;                  //apaga displays
88     PORTE = 0;                     //limpa PORTE
89  //habilita interrupção de estouro de TMR0
90     INTCONbits.GIE = 1;               //liga chave geral de interrupção
91     INTCONbits.TMR0IE = 1;            //liga chave individual de interrupção externa
92     INTCON2bits.TMR0IP = 1;            //interrupção externa 0 ocorrerá na borda de
93  //configura TMR0 para operar como temporizador e estourar a cada 4ms
94     T0CON = 0b11000100;               //Timer0 configurado como temporizador<5>
95                               //prescaler ativado<3>
96                               //fator de prescaler de 1:32<2:0>
97                               //liga TMR0
98     TMR0L = 5;                     //inicializa TMR0
99  }//*********************************************************************
```

Timer 0 / Código-fonte (Ex. 8.2)

```
1  #include <p18f4520.h>
2  #include <stdio.h>
3  #include "pic_simb.h"
4
5  #pragma config OSC = XT, WDT = OFF, MCLRE = OFF
6  #pragma config DEBUG = OFF, LVP = OFF, PWRT = ON, BOREN = OFF
7
8  #pragma code isr = 0x000008
9  #pragma interrupt ISR
10 void ISR(void)
11 {
12    INTCONbits.TMR0IF = 0;    // apaga o flag de interrupção
13    TMR0H = 0x85;
14    TMR0L = 0xEE;
15    LATBbits.LATB0 = !LATBbits.LATB0;    // inverte o estado do led
16 }
17 #pragma code
18
19 void main(void)
20 {
21    ADCON1 = 0x0F;              // desliga entradas analógicas
22    TRISBbits.TRISB0 = 0;       // RB0 como saída
23    TMR0H = 0x85;
24    TMR0L = 0xEE;
25    T0CON = bTMR0ON | bT0CLK_PRE32;
26    INTCON = bGIE | bTMR0IE;    // habilita GIE e TMR0IE
27    while(1);                   // aguarda uma interrupção
28 }
```

Próxima Aula

Aula 16

Periféricos Analógicos