FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

Aula 18
Modulação PWM

Microcontroladores PIC18 – Programação em C

MICROCHIP

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

Jan / 2011

Módulo CCP (Captura/Comparação/PWM)

- Módulos CCP
  - Captura
  - Comparação
  - Modulação por largura de pulso (PWM)

Módulo CCP (Captura/Comparação/PWM)

- Os módulos CCP são formados por um registrador (CCPRx) de 16 bits que pode ser
  - Registrador de captura de 16 bits;
  - Registrador de comparação de 16 bits
  - Registrador de “duty-cycle” no modo PWM
- Dois módulos
  - CCP1 e CCP2

Módulo CCP 1

- Registrador CCPR1
  - registrador CCP do módulo CCP1
- dois registradores de 8 bits
  - CCPR1L
  - CCPR1H

Módulo CCP 2

- Registrador CCPR2
  - registrador CCP do módulo CCP2
- dois registradores de 8 bits
  - CCPR2L
  - CCPR2H
- Registrador CCP2CON
  - Registrador que controla a operação do CCP2

Módulos de Captura

- Captura de quê?
- O valor de 16 bits do registrador TMR1 é copiado para os registradores CCPR1H:CCPR1L e/ou CCPR2H:CCPR2L quando ocorre um dos eventos:
  - A cada borda de descida do sinal no pino RC2, RC1
  - A cada borda de subida do sinal no pino RC2, RC1
  - A cada 4 bordas de subida do sinal no pino RC2, RC1
  - A cada 16 bordas de subida do sinal no pino RC2, RC1

Módulos de Captura

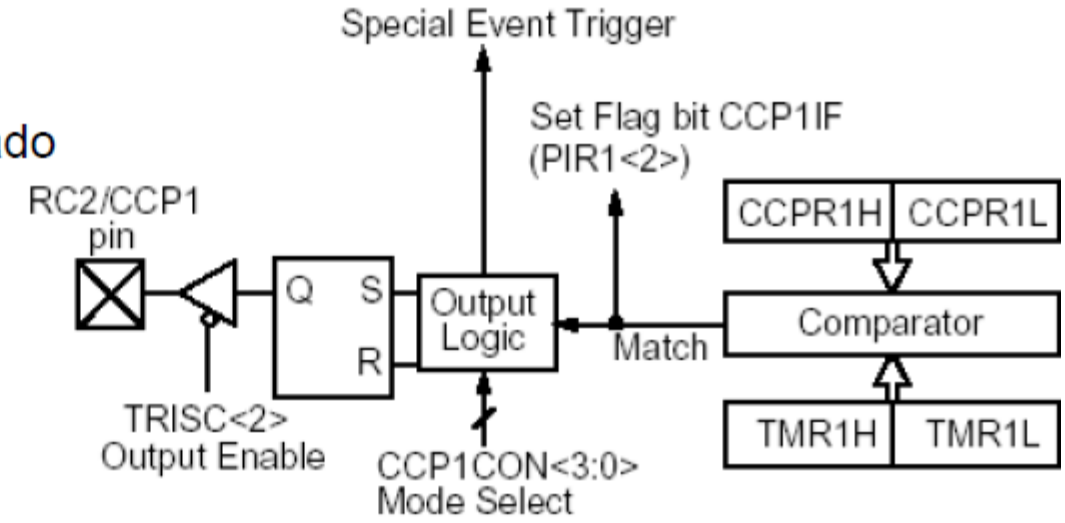
- Os pinos RC2/CCP1, RC1/CCP2 devem ser configurados como entrada
  - Setar o bit 1 e 2 do registrador TRISC
- O Timer 1 deve estar operando
  - no modo timer
  - ou no modo contador com sincronização do sinal de clock externo com o clock interno

Módulos de Captura

- Seleção do tipo de evento
  - bits de controle CCP1M3:CCP1M0 (CCP1CON <3-0>).
- Quando uma captura é realizada, o flag CCP1IF (PIR1<2>) é setado
  - Ao final da rotina de interrupção, deve ser apagado por software
- Se uma outra captura ocorrer antes que o valor no registrador CCPR1 for lido, o valor anterior é perdido

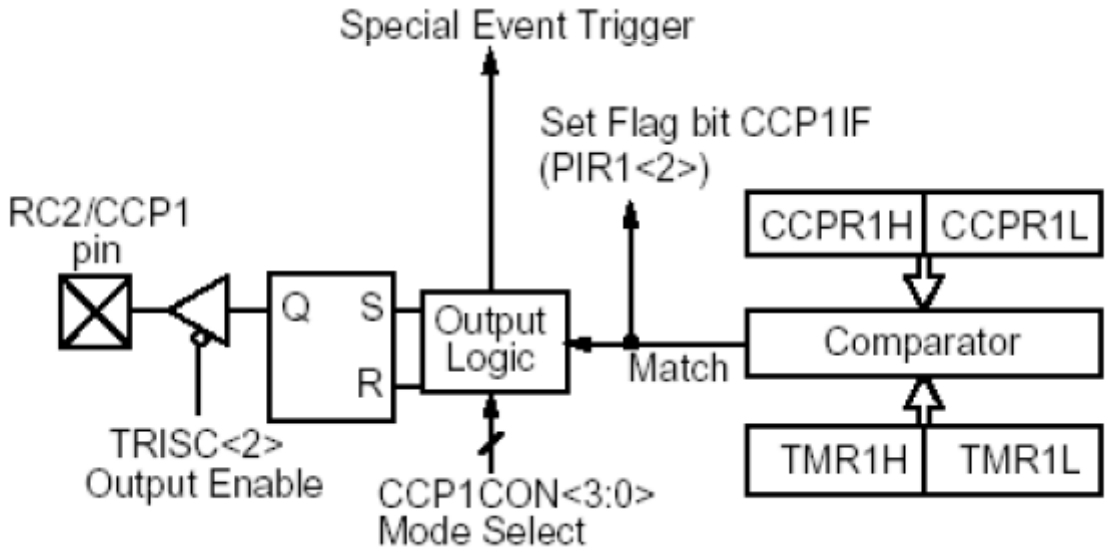
Módulos de Comparação

- Comparação de quê?
- O valor de CCPR1 (16 bits) e/ou CCPR2 (16 bits) é constantemente comparado com o valor de TMR1
- Quando os valores são iguais, o pino RC2/CCP1 e/ou RC1/CCP2 fica em:
  - Nível alto
  - Nível baixo
  - Ou permanece inalterado



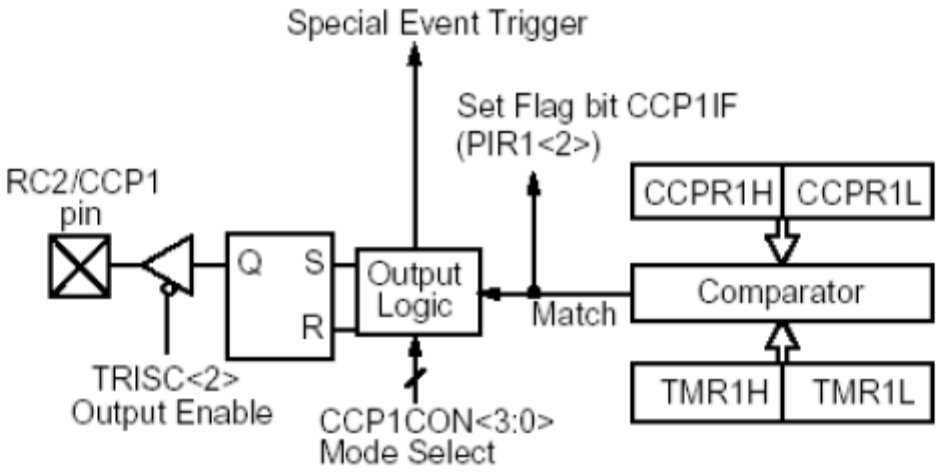
Módulos de Comparação

- A ação é configurada pelos bits de controle CCP1M3:CCP1M0 (bits de 3 a 0 do Reg. CCP1CON)
- O flag CCP1IF é setado, se a interrupção estiver habilitada

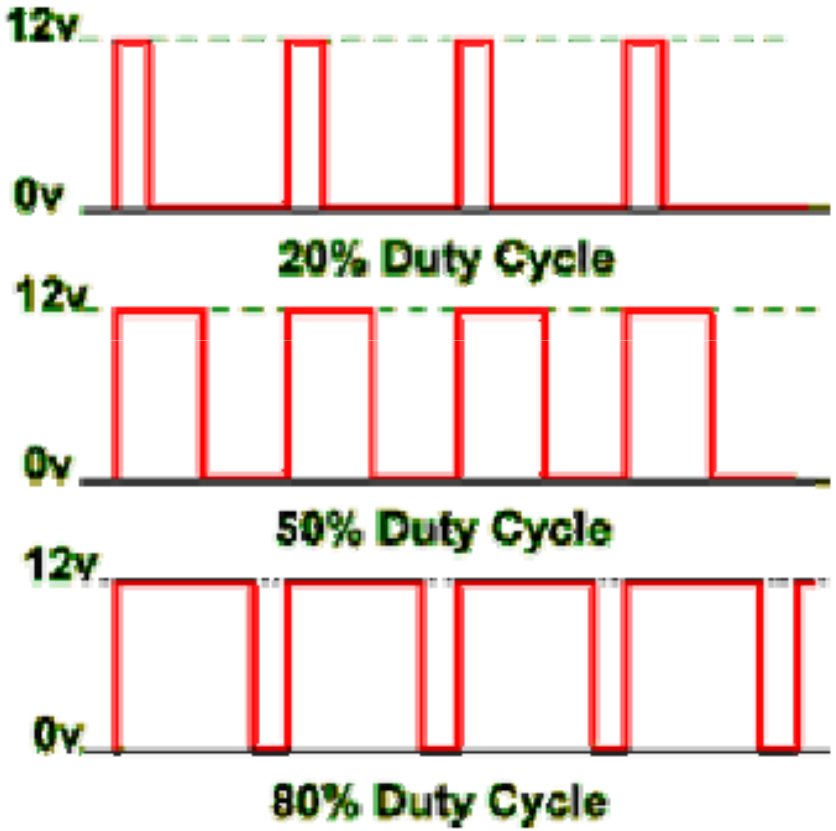


Módulos de Comparação

- Os pinos RC2/CCP1 e RC1/CCP2 devem ser configurados como saídas
  - limpar os bits 1 e 2 do registrador TRISC
- O Timer 1 deve estar operando no modo timer ou no modo contador com sincronização do sinal de clock externo com o clock interno

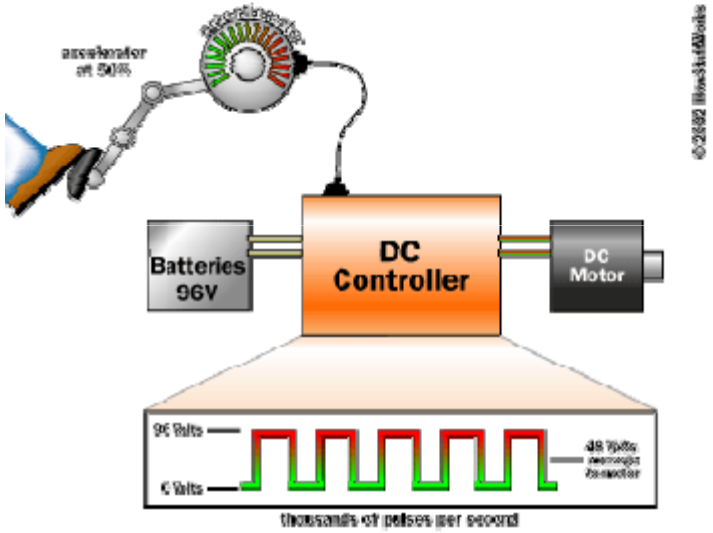


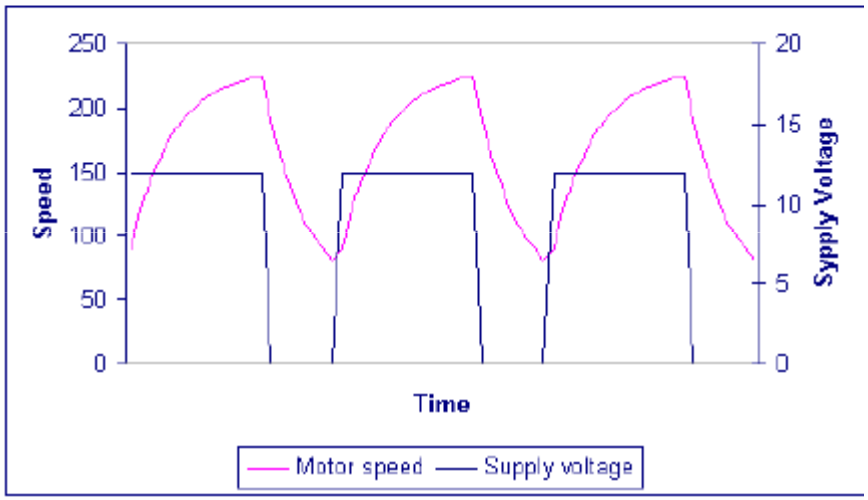
Modo PWM



Modo PWM

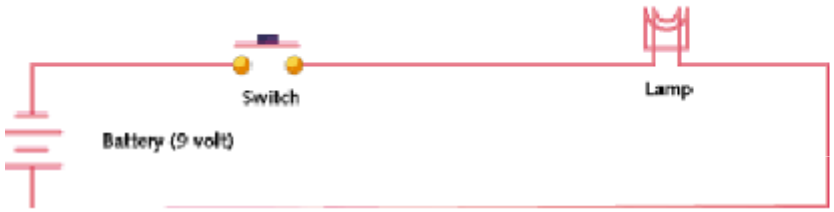
- Ex. 1: motor DC

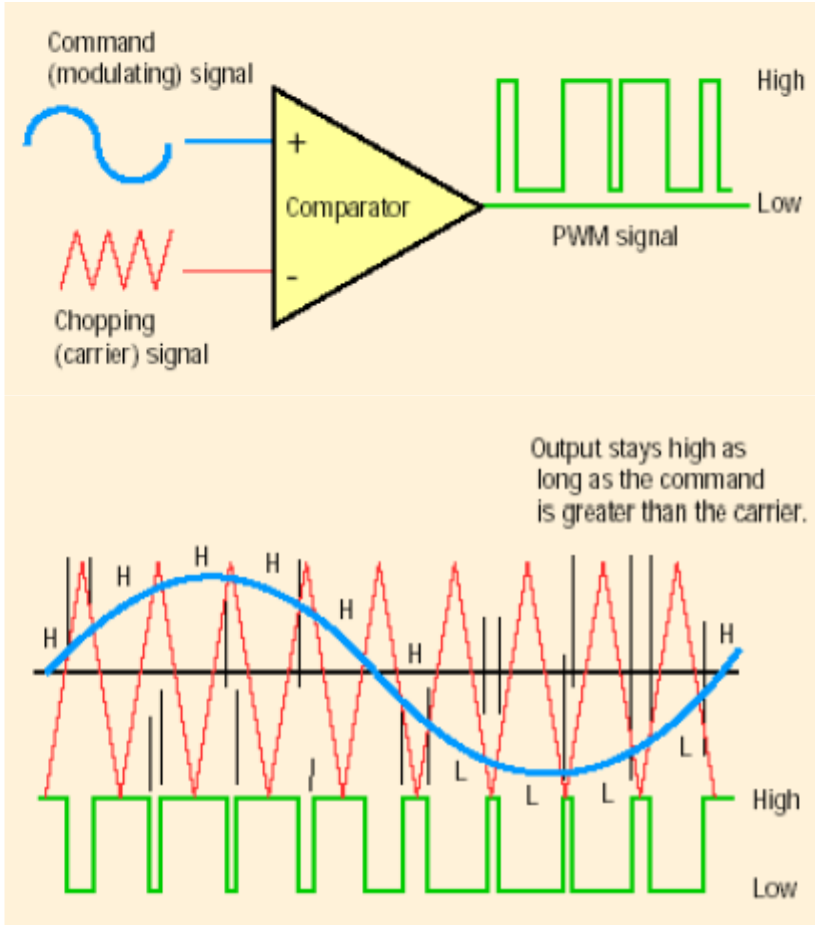




Modo PWM

- Ex. 2: Lampada



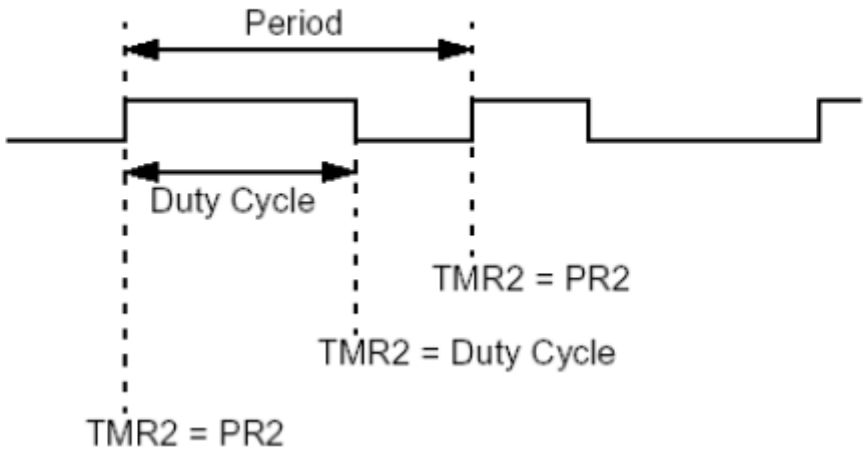


Modo PWM

- O pino CCPx produz um sinal de PWM com uma resolução de até 10 bits
- Os pinos RC2/CCP1, RC1/CCP2 devem ser configurados como saídas
  - Limpar os bits 1 e 2 de TRISC

Modo PWM

- Período PWM = [(PR2) + 1] • 4 • Tosc • (pré-escalonador do TMR2)
- Duty Cycle do PWM = (CCPR1L:CCP1CON<5:4>) • Tosc • (pré-escalonador do TMR2)



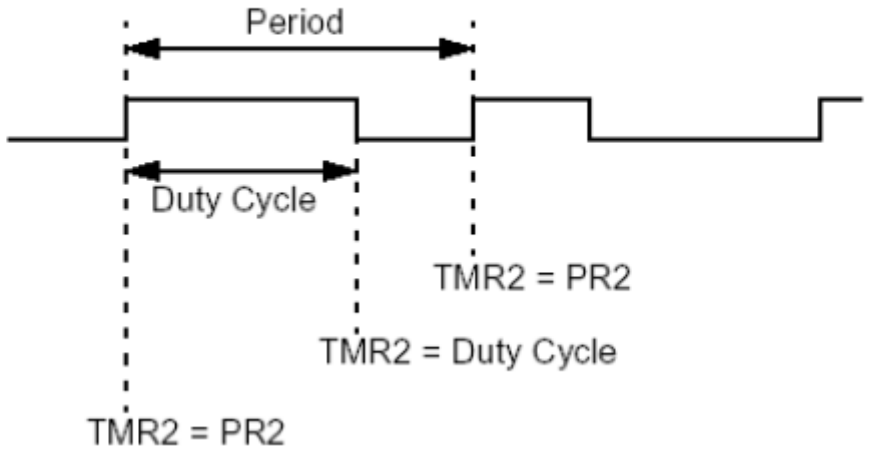
Modo PWM

$$\text{PWM Period} = [(PR2) + 1] \bullet 4 \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

$$\text{PWM Duty Cycle} = (CCPRxL:CCPxCON\langle 5:4\rangle) \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

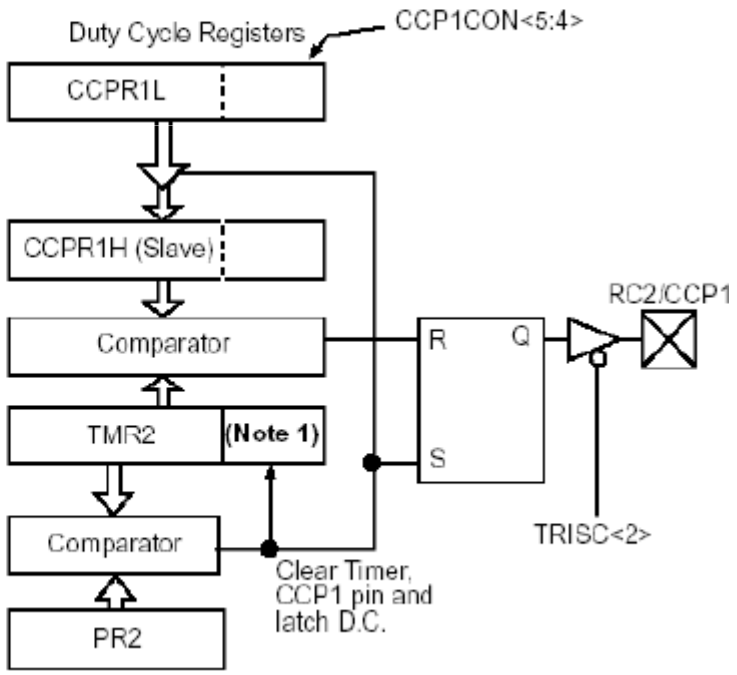
Modo PWM

- Quando TMR2 é igual a PR2:
  - O TMR2 é apagado;
  - O pino CCP1 é setado (exceto se o duty-cycle do PWM = 0%, quando o CCP1 não será setado);
  - O PWM duty-cycle é copiado de CCPR1L para CCPR1H



Modo PWM

- O pino CCPx produz um sinal de PWM com uma resolução de até 10 bits
- Os pinos RC2/CCP1, RC1/CCP2 devem ser configurados como saídas
  - Limpar os bits 1 e 2 de TRISC



Passos para ajustar o Modo PWM

- Ajustar o período do sinal PWM
  - Escrever em PR2
- Ajustar o duty-cycle do sinal PWM
  - Escrever em CCPR1L e nos bits 5 e 4 de CCP1CON
- Configurar os pinos RC2/CCP1 e/ou RC1/CCP2 como saídas
  - Limpar o bit 1 e/ou 2 do reg. TRISC
- Ajustar o valor do pré-escalonador e habilitar o Timer 2 configurando T2CON
- Configurar o módulo CCP1 e/ou CCP2 para operação no modo PWM
  - CCPxCON< CCPxM3: CCPxM0> = 11xx

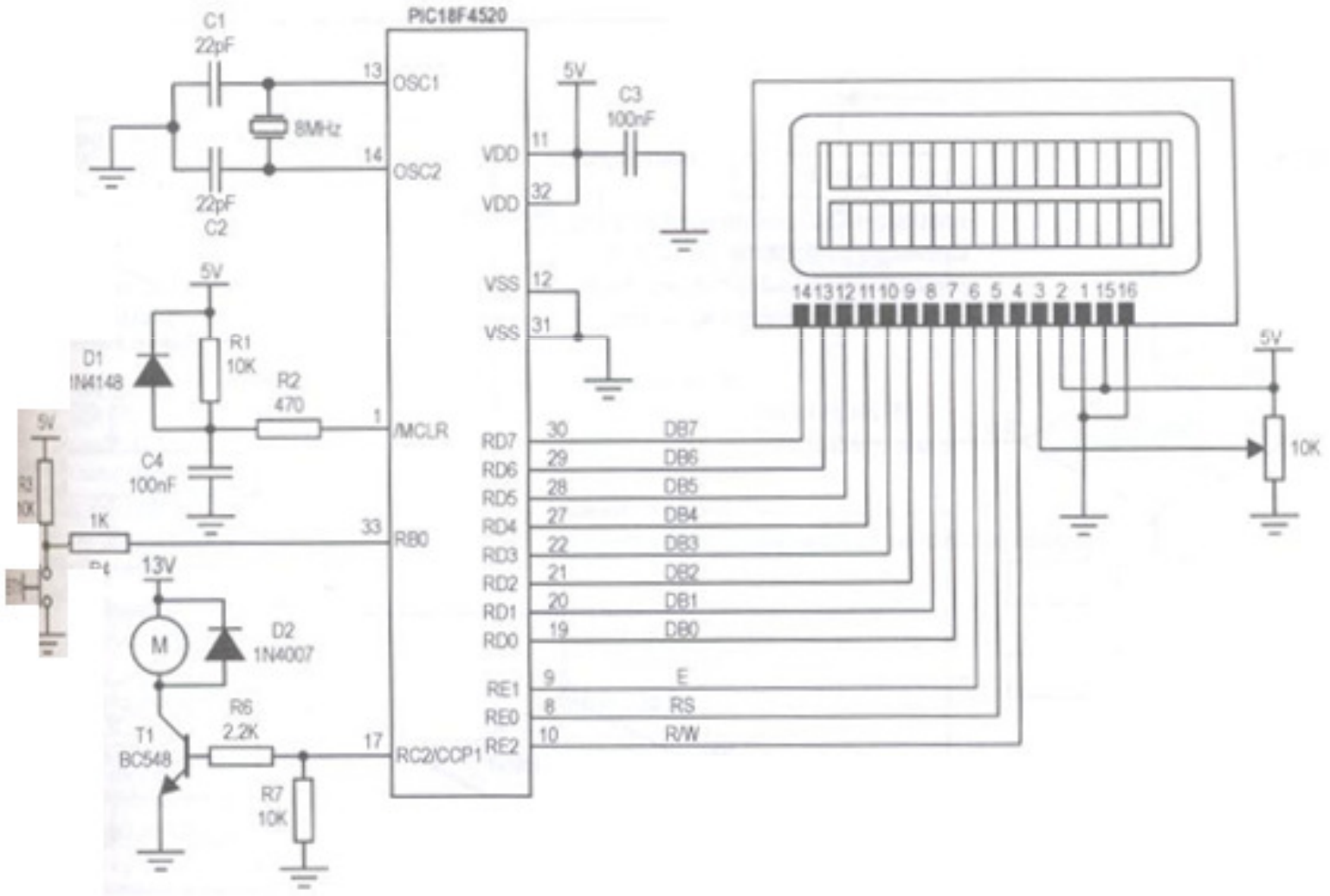
Módulo ECCP (Captura/Comparação/PWM)

- Os PIC18F4520 incluem também alguns modos especiais de operação do módulo CCP1 que o tornam um ECCP, ou seja um CCP melhorado;
- Na verdade, os melhoramentos estão relacionados apenas ao modo de PWM (sída simples, meia-ponte e ponte-completa);
- As funcionalidades de captura e de comparação operam da mesma forma;

Registradores Associados com PWM e TIMER2

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
RCON	IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	48
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
TRISB	PORTB Data Direction Register								52
TRISC	PORTC Data Direction Register								52
TMR2	Timer2 Register								50
PR2	Timer2 Period Register								50
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	50
CCPR1L	Capture/Compare/PWM Register 1 Low Byte								51
CCPR1H	Capture/Compare/PWM Register 1 High Byte								51
CCP1CON	P1M1(1)	P1M0(1)	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	51
CCPR2L	Capture/Compare/PWM Register 2 Low Byte								51
CCPR2H	Capture/Compare/PWM Register 2 High Byte								51
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	51
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1(1)	PSSBD0(1)	51
PWM1CON	PRSEN	PDC6(1)	PDC5(1)	PDC4(1)	PDC3(1)	PDC2(1)	PDC1(1)	PDC0(1)	51

Esquema Elétrico



Serial UART (Código-fonte)

- LCD_8bits.h

  Arquivo cabeçalho com as definições dos pinos utilizados como via de dados, vias de controle e os protótipos das funções;

- LCD_8bits.c

  Arquivo que contém as funções de acesso ao LCD;

- Main_36.c

  Arquivo principal responsável por acionar uma ventoinha com um sinal PWM;

Display LCD / Funções

Função	Descrição
IniciaLCD	Inicializa LCD *controller*
TestPixelsLCD	Acende todos os *pixels* do LCD
EscInstLCD	Envia instrução para o LCD
EscDataLCD	Escreve um caractere na posição apontada pelo cursor
EscStringLCD	Escreve uma string lida na memória de dados a partir da posição apontada pelo cursor
EscStringLCD_ROM	Escreve uma string lida na memória de programa a partir da posição apontada pelo cursor
TesteBusyFlag	Verifica se o LCD *controller* está ocupado executando alguma instrução
Pulse	Aplica pulso de para leitura ou escrita no LCD
_Delay100us	Delay de 100us
_Delay5ms	Delay de 5ms
DelayFor20TCY	Delay de 20 ciclos de instrução do oscilador

LCD_8bits.h

Serial UART - RS232 (Código-fonte)

```
8  #ifndef __LCD_8BITS_H
9  #define __LCD_8BITS_H
10 //*************************************************************
11 //definições do port ligado no LCD
12 //*
13 #define PORT_LCD       PORTD          //LCD ligado no PORTD
14 #define PORT_CONT_LCD    PORTE          //PORT de controle do LCD
15 #define TRIS_PORT_LCD    TRISD          //direção dos pinos
16 #define TRIS_CONT_LCD    TRISE          //direção dos pinos
17 //*************************************************************
18 //definições dos pinos de controle
19 #define _RS PORTEbits.RE0              //pino dado/instrução
20 #define _EN PORTEbits.RE1              //pino enable
21 #define _RW PORTEbits.RE2              //pino escrita/leitura
22 //*************************************************************
23 //protótipos de funções
24 void IniciaLCD (unsigned char NL);
25 void Pulse(void);
26 void _Delay100us(void);
27 void _Delay5ms(void);
28 void TestPixelsLCD(void);
29 void DelayFor20TCY( void);
30 void DelayFor18TCY( void);
31 unsigned char TesteBusyFlag(void);
32 void EscDataLCD(char _data);
33 void EscInstLCD(unsigned char _inst);
34 void EscStringLCD(char *buffer);
35 void EscStringLCD_ROM(const rom char *buffer);
36 //*************************************************************
37 #endif
38
```

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

LCD_8bits.c

Serial UART (Código-fonte)– 1

```
9  /*********************************************************************
10 Esta biblioteca contém um conjunto de funções que permitem ao microcontrolador
11 se comunicar com o LCD controller HD44780.
12 *********************************************************************/
13 #include <p18cxxx.h>                  //diretiva de compilação
14 #include<delays.h>                     //diretiva de compilação
15 #include "LCD_8bits.h"                  //diretiva de compilação
16 /*********************************************************************
17 A função IniciaLCD() recebe como argumento um valor que irá inializar o LCD com:
18
19 valor = 1 -> inicializa o LCD com uma linha
20 valor != 1 -> inicializa o LCD com linha dupla
21
22 Quando o programa retorna ao ponto de chamada, o LCD mostra o cursor piscando
23 na primeira posição da primeira linha.
24 *********************************************************************/
```

LCD_8bits.c

Serial UART (Código-fonte) - 2

NL: Define o número de linhas que estarão ativas;

```
25  void IniciaLCD (unsigned char NL)
26  {
27     const unsigned char Seq_Inic[3] = {0x0F, 0x06, 0x01};    //declaração de vetor
28     unsigned char i;                                         //declaração de variável local
29     char x;                                                  //declaração de variável local
30     _EN = 0;                          //envia intrução
31     _RS = 0;                          //limpa pino enable
32     _RW = 0;                          //ativa ciclo de escrita
33     ADCON1 = 0x0F;                      //configura PORT de controle com digital
34     TRIS_CONT_LCD = 0;                   //configura PORT de controle como saída
35     TRIS_PORT_LCD = 0;                   //configura PORT de dados como saída
36  //************** envia para o LCD o comando 0x30 três vezes
37     for(i=0;i<3;i++)
38        {
39        PORT_LCD = 0x30;                 //comando 0x30
40        Pulse();                        //aplica pulso enable no LCD
41        _Delay5ms();                     //delay 5ms
42        }
43  //************* configura linha simples ou linha dupla
44     if(NL == 1)PORT_LCD = 0x30;          //se NL=1, ativa uma linha
45     else PORT_LCD = 0x38;                //se NL!=1, ativa duas linhas
46     Pulse();                        //aplica pulso enable no LCD
47     _Delay5ms();                     //delay 5ms
48  //*********************************
49     for(i=0;i<3;i++)
50        {
51        PORT_LCD = Seq_Inic[i];           //LCD recebe comando
52        Pulse();                        //aplica pulso enable no LCD
53        _Delay5ms();                     //delay 5ms
54        }
55     TRIS_PORT_LCD = 0xFF;               //configura PORT de dados como entrada
56  }                                  //final  da função IniciaLCD
```

LCD_8bits.c

Serial UART (Código-fonte) - 3

São utilizadas para gerar a base de tempo exigida pelo LCD

Precisam que o arquivo cabeçalho delay.h seja incluído no projeto.

Desenvolvida para freqüência de clock de 8Mhz.

```
58  //esta função escreve comando/dado no LCD
59  void Pulse(void)
60     {
61     DelayFor20TCY();                 //delay de 20 ciclos de clock
62     _EN = 1;                        //seta pino enable
63     DelayFor20TCY();                 //delay de 20 ciclos de clock
64     _EN = 0;                        //limpa pino enable
65     }
66  //***************************************************************************
67  //                          funções de delay
68  //***************************************************************************
69  //delay de 100us
70  void _Delay100us(void)
71  {
72     Delay100TCYx(2);                 //delay 100us
73  }
74  //***************************************************************************
75  //delay de 5ms
76  void _Delay5ms(void)
77  {
78     Delay10KTCYx(1);                 //delay 5ms
79  }
80  //***************************************************************************
81  //delay 20 ciclos do oscilador principal
82  void DelayFor20TCY( void )
83  {
84     Nop(); Nop();Nop(); Nop(); Nop();
85     Nop(); Nop();Nop(); Nop(); Nop();
86     Nop(); Nop();Nop(); Nop(); Nop();
87  }
```

LCD_8bits.c

Serial UART (Código-fonte) - 4

Verifica se o LCD está ocupado executando alguma instrução ou se ele está livre;

```
96  void DelayFor18TCY( void )
97  {
98      Nop(); Nop();Nop(); Nop(); Nop(); Nop(); Nop();
99      Nop(); Nop();Nop(); Nop(); Nop(); Nop(); Nop();
100 }
101 //*****************************************************************
102 //             funções de acesso ao LCD
103 //*****************************************************************
104 /*esta função fica aguardando o LCD controller terminar de executar
105 a instrução atual. ela retorna o  valor 0 quando a instrução terminar.*/
106 unsigned char TesteBusyFlag(void)
107 {
108 //   TRIS_PORT_LCD = 0xFF;            //configura PORT de dados como entrada
109
110     _RW = 1;                          //ativa ciclo de leitura
111     _RS = 0;                        //ciclo de intrução
112     DelayFor20TCY();                  //delay de 20 ciclos de clock
113     _EN = 1;                                //seta pino enable
114     DelayFor20TCY();                  //delay de 20 ciclos de clock
115     if(PORT_LCD&0x80)                       //leitura do bit busy flag
116     {                                         //se bit busy == 1, LCD ocupado
117         _EN = 0;                          //reseta pino enable
118         _RW = 0;                          //reseta linha de escrita
119         return 1;                          //LCD ocupado, retorna 1
120     }
121     else                                    //se busy flag == 0, LCD livre
122     {
123         _EN = 0;                          //reseta pino enable
124         _RW = 0;                        //ativa ciclo de escrita
125         return 0;                       //LCD livre, retorna 0
126     }
```

LCD_8bits.c

Serial UART (Código-fonte) - 5

Verifica se o LCD está ocupado executando alguma instrução ou se ele está livre;

Verifica se o LCD está ocupado executando alguma instrução ou se ele está livre;

```
128  //esta função escreve um caractere na pocição apontada pelo cursor.
129  void EscDataLCD(char _data)
130  {
131      TRIS_PORT_LCD = 0;              //configura PORT de dados como saída
132      PORT_LCD = _data;              //escreve dado
133      _RS = 1;                      //envia dado
134      _RW = 0;                      //ativa ciclo de escrita
135      Pulse();                      //aplica pulso enable no LCD
136      _RS = 0;                      //envia instrução
137       DelayFor20TCY();               //delay de 20 ciclos de clock
138      TRIS_PORT_LCD = 0xFF;           //configura PORT de dados como entrada
139  //*****************************************************************
140  }                                 //final da função EscDataLCD
141  //*****************************************************************
142  //esta função envia uma instrução para o LCD.
143  void EscInstLCD(unsigned char _inst)
144  {
145      TRIS_PORT_LCD = 0;              //configura PORT de dados como saída
146      PORT_LCD = _inst;              //escreve instrução
147      _RS = 0;                      //envia intrução
148      _RW = 0;                      //ativa ciclo de escrita
149      Pulse();                      //aplica pulso enable no LCD
150      _RS = 0;                      //envia dado
151       DelayFor20TCY();               //delay de 20 ciclos de clock
152      TRIS_PORT_LCD = 0xFF;           //configura PORT de dados como entrada
153  //*****************************************************************
154  }                                 //final da função EscInstLCD
```

LCD_8bits.c

Serial UART (Código-fonte) - 6

Envia para o LCD a *string* lida na memória de dados que será exibida no *display* a partir da posição apontado pelo cursor;

Envia para o LCD a *string* lida na memória de programa que será exibida no *display* a partir da posição apontado pelo cursor;

```
155  //*****************************************************************/
156  /*esta função escreve no LCD uma string lida da memória RAM a partir
157  da posição aponntada pelo cursor.*/
158  #pragma code My_codigo = 0x200
159  void EscStringLCD(char *buff)
160  {
161      while(*buff)                          //escreve caractere até econtrar null
162      {
163         while(TesteBusyFlag());           //espera LCD terminar de executar instruçã
164         EscDataLCD(*buff);               //escreve no LCD caractere apontado por buff
165           buff++;                          // Incrementa buffer
166      }
167  //*****************************************************************
168  }                                  //final da função EscStringLCD
169  #pragma code
170  //*****************************************************************
171  /*esta função escreve no LCD uma string lida da memória de programa
172  a partir da posição aponntada pelo cursor.*/
173  void EscStringLCD_ROM(const rom char *buff)
174  {
175      while(*buff)                          // Write data to LCD up to null
176      {
177         while(TesteBusyFlag());           //espera LCD controller terminar de executa
178           EscDataLCD(*buff);              //escreve no LCD caractere apontado por buf
179           buff++;                          // Incrementa buffer
180      }
181           return;
182  //*****************************************************************
183  }                                  //final da função EscStringLCD_ROM
```

LCD_8bits.c

Serial UART (Código-fonte) - 7

Função que acende todos os pixels do display do LCD;

Escreve cursor na primeira linha

Posiciona cursor na segunda linha

Caractere com todos os pixels acesos

```
185  //esta função testa o LCD acendendo todos os pixels do display.
186  void TestPixelsLCD(void)
187  {
188  unsigned char BffCheio[32];              //declaração de vetor
189  unsigned char i;                        //declaração de variável local
190     EscInstLCD(0x80);                     //posiciona cursor na primeira posição da primeira linha
191     while(TesteBusyFlag());                  //espera LCD controller terminar de executar instrução
192
193     for(i=0;i<32;i++)                     //laço de iteração
194     {
195        if(i<16)                          //i < 16?
196        {                                 //sim, executa bloco de código a seguir
197           EscDataLCD(0xFF);              //escreve caractere na posição pantada pelo  cursor
198           while(TesteBusyFlag());           //espera LCD controller terminar de executar instrução
199        }
200        else    if(i==16)                 //i==16?
201        {                                 //sim, executa bloco de código a seguir
202           EscInstLCD(0xC0);              //posiciona cursor na primeira posição da segunda linha
203           while(TesteBusyFlag());           //espera LCD controller terminar de executar instrução
204
205           EscDataLCD(0xFF);              //escreve caractere na posição pantada pelo  cursor
206           while(TesteBusyFlag());           //espera LCD controller terminar de executar instrução
207        }
208        else                              //se i !=16 executa bloco de c[odigo a seguir
209        {
210           EscDataLCD(0xFF);              //escreve caractere na posição pantada pelo  cursor
211           while(TesteBusyFlag());           //espera LCD controller terminar de executar instrução
212        }
213     }
214  //*********************************************************************
215  }
```

Main_36.c

Serial UART (Código-fonte) - 1

```
8   #include <p18f4520.h>       //diretiva de compilação
9   #include <delays.h>          //diretiva de compilação
10  #include <stdio.h>          //diretiva de compilação
11  #include "Lcd_8bits.h"       //diretiva de compilação
12  //***************************************************************
13  //protótipos de funções
14  void Inic_Regs (void);
15  void Atual_LCD (void);
16  void Confg_PWM (void);
17  void  Config_Duty_Cycle (int x);
18  //***************************************************************
19  //variáveis globais
20  char buf [17];
21  //char buf2 [17];
22  char z=0;
23  //***************************************************************
24  //definições
25  #define Botao1   PORTBbits.RB0
26  //***************************************************************
```

Main_36.c

Serial UART (Código-fonte) - 2

```
27  void main(void)                    //função main
28  {
29  float temp=0;                     //declaração  de variável local inicializada
30  int dly=0;                        //declaração  de variável local inicializada
31  char status_botao=0;               //declaração  de variável local inicializada
32  unsigned char count=0;              //declaração  de variável local inicializada
33  //*********************************
34     Inic_Regs ();                  //configurar SFRs
35     IniciaLCD (2);                  //inicializar LCD controller HD44780
36     TestPixelsLCD();               //teste no LCD - acende todos os pixels.
37
38     EscInstLCD(0x0C);               //desativa cursor
39     while(TesteBusyFlag());            //espera LCD controller terminar de executar
40  //*********************************
41  //delay de 3 segundos
42     for(dly=0;dly<600;dly++)         //comando de iteração
43     {
44        _Delay5ms();               //delay de 5ms
45     }
46  //*********************************
47     EscInstLCD(0x01);               //desativa cursor
48     while(TesteBusyFlag());            //espera LCD controller terminar de executar
49
50     sprintf(buf,"PWM desligado");      //converte valor em string e armazena no vet
51     EscStringLCD(buf);               //escreve string no LCD
52     while(TesteBusyFlag());            //espera LCD controller terminar de executar
53  //*********************************/
```

Main_36.c

Serial UART (Código-fonte) - 3

```
55    while (1)
56    {
57       if(!Botao1)                    //Botão 1 pressionado?
58       {                             //sim, executa bloco de código
59          if(!status_botao)             //Botão 1 foi tratado?
60          {                          //não, executa bloco de código
61             status_botao=1;             //sinaliza que Botão foi tratado
62              if(z==0)                //PWM está desligado?
63             {                          //sim, executa bloco de código
64                Confg_PWM ();          //ativa PWM com duty cycle de 50%
65                z=1;                //PWM ligado com duty cycle de 50%
66             }
67             else if(z==1)              //PWM está ligado com duty cycle de 50%?
68             {                          //sim, executa bloco de código
69                Config_Duty_Cycle (100);//atualiza duty cycle para 100%
70                z+=1;                //PWM ligado com duty cycle de 100%
71             }
72             else if(z==2)              //PWM ligado com duty cycle de 100%?
73             {
74                CCP1CON=0;             //sim, desliga PWM
75                z=0;                //PWM desligado
76             }
77          }
78       }
79       else status_botao=0;             //sinaliza que Botão foi tratado
80       Delay1KTCYx(100);                //delay de 20ms
81       count+=1;                       //incrementa count
82       if(count==50)
83       {
84          Atual_LCD();                 //atualiza LCD a cada segundo
85          count=0;                    //zera count
86       }
```

Main_36.c

Serial UART (Código-fonte) - 4

```
92   void Inic_Regs (void)
93   {
94      TRISA = 0x00;                    //PORTA saída
95      TRISB = 0x01;                    //RB0 como entrada e demais pinos do PORTB como
96      TRISC = 0x00;                    //PORTC saída
97      TRISD = 0x00;                    //PORTD saída
98      TRISE = 0x00;                    //PORTE saída
99      ADCON1 = 0x0F;                    //configura pinos dos PORTA e PORTE como digit
100     PORTA = 0;                       //limpa PORTA
101     PORTB = 0;                       //limpa PORTB
102     PORTC = 0;                       //limpa PORTC
103     PORTD = 0x00;                    //apaga displays
104     PORTE = 0;                       //limpa PORTE
105  }
106  //*********************************************************************
107  void Atual_LCD (void)
108  {
109     EscInstLCD(0x01);                  //limpa display e mostra cursor piscando na
110     while(TesteBusyFlag());              //espera LCD controller terminar de exec
111
```

Main_36.c

Serial UART (Código-fonte) - 5

```
113     if(z==0)                          //PWM desligado?
114     {
115        sprintf(buf,"PWM desligado");      //sim, converte valor em string e armazena no vetor
116        EscStringLCD(buf);               //escreve string no LCD
117        while(TesteBusyFlag());            //espera LCD controller terminar de executar instruç
118     }
119     else if(z==1)                       //PWM ligado com duty cycle de 50%?
120     {
121        sprintf(buf," PWM ligado");         //sim, converte valor em string e armazena no vetor
122        EscStringLCD(buf);               //escreve string no LCD
123        while(TesteBusyFlag());            //espera LCD controller terminar de executar instruç
124
125        EscInstLCD(0xC0);                //limpa display e mostra cursor piscando na primeira po
126        while(TesteBusyFlag());            //espera LCD controller terminar de executar instruç
127
128        sprintf(buf,"duty cycle 50%");      //converte valor em string e armazena no vetor buf
129        EscStringLCD(buf);               //escreve string no LCD
130        while(TesteBusyFlag());            //espera LCD controller terminar de executar instruç
131     }
132     else if(z==2)                       //PWM ligado com duty cycle de 100%?
133     {
134        sprintf(buf,"PWM ligado");         //sim, converte valor em string e armazena no vetor
135        EscStringLCD(buf);               //escreve string no LCD
136        while(TesteBusyFlag());            //espera LCD controller terminar de executar instruç
137
138        EscInstLCD(0xC0);                //limpa display e mostra cursor piscando na primeira po
139        while(TesteBusyFlag());            //espera LCD controller terminar de executar instruç
140
141        sprintf(buf,"duty cycle 100%");      //converte valor em string e armazena no vetor buf
142        EscStringLCD(buf);               //escreve string no LCD
143        while(TesteBusyFlag());            //espera LCD controller terminar de executar instruç
144     }
```

Main_36.c

Serial UART (Código-fonte) - 6

```
147  void Confg_PWM (void)
148  {
149     PR2 = 127;                          //período do sinal PWM = 15625Hz
150     CCPR1L = 0b01000000;
151     CCP1CONbits.DC1B1 = 0;
152     CCP1CONbits.DC1B0 = 0;                //bits de controle = 256
153     TRISCbits.RC2 = 0;                   //configura pino de saída do sinal PWM cc
154  //configura TMR2 para operar como temporizador e estourar a cada 4ms
155     T2CON = 0b00000100;                 //fator de postscaler de 1:1 <6:3>
156                                         //fator de prescaler de 1:1<1:0>
157                                         //liga TMR2<1>
158     TMR2 = 0;                           //inicializa TMR2
159     CCP1CON *= 0b00110000;
160     CCP1CON += 0b00001100;              //ativa modo PWM
161  //esta função atualiza o duty cycle do sinal PWM
162  void  Config_Duty_Cycle (int x)
163  {
164  int temp;                            //declaração de variável local
165     if(x>100)x=100;                     //ajusta valor de x
166     x = (x*100)/512;                   //obtém valor dos bits de controle
167     temp =  x >> 2;                     //obtém os seis bits mais significativos
168     CCPR1L = temp;                     //carrega bits de controle mais significativc
169  //*********************
170  //carrega segundo bit de controle menos significativo
171     temp = x & 2;
172     if (temp==1)CCP1CONbits.DC1B1 = 1;
173     else    CCP1CONbits.DC1B1 = 0;
174  //carrega bit de controle menos significativo
175     temp = x & 1;
176     if (temp==1)CCP1CONbits.DC1B0 = 1;
177     else    CCP1CONbits.DC1B0 = 0;
178  }//*****************************************************************
```

PWM (Código-fonte) - Ex.8.4

```
1  #include <p18f4520.h>
2  #include <stdio.h>
3  #include "pic_simb.h"
4
5  #pragma config OSC = XT, WDT = OFF, MCLRE = ON
6  #pragma config DEBUG = OFF, LVP = OFF, PWRT = ON, BOREN = OFF
7  #pragma config CCP2MX = PORTBE
8
9  #pragma code isr = 0x000008
10 #pragma interrupt tmr2_ISR
11 void tmr2_ISR(void)
12 {
13    static unsigned char dir;
14    PIR1bits.TMR2IF = 0;          // apaga flag de interrupção
15    if (!dir)
16    {
17       if (CCPR2L<255) CCPR2L++; else   // incrementa o ciclo ativo (CCPR2L)
18       {   // quando CCPR2L = 255
19          CCP2CON |= 0x30;   // força ciclo ativo máximo (DC2B1 e DC2B0 = 1)
20          dir=1;             // muda a direção
21       }
22    } else
23    {
24       if (CCPR2L) CCPR2L--; else   // decrementa ciclo ativo
25       {   // quando CCPR2L = 0
26          CCP2CON = bCCP_PWM;   // força ciclo mínimo (DC2B1 e DC2B0 = 0)
27          dir=0;             // muda a direção
28       }
29    }
30 }
31 #pragma code
```

PWM (Código-fonte) - Ex.8.4

```
33  void main(void)
34  {
35      ADCON1 = 0x0F;              // desliga entradas analógicas
36      TRISB = 0;              // RB0 a RB7 como saídas
37      T2CON = bTMR2ON | bT2CLK_PRE1 | bT2OUTPS_8;
38      CCP2CON = bCCP_PWM;
39      INTCON = bGIE | bPEIE;     // habilita GIE e PEIE
40      PIE1 = bTMR2IE;             // habilita TMR2IE
41      while(1);             // loop infinito
42  }
```

Próxima Aula

Aula 19

EEPROM, Flash e Módulos de Baixo Consumo