FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

Aula 16
Periféricos Analógicos

Microcontroladores PIC18 – Programação em C

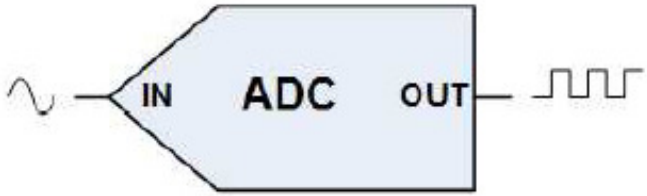
MICROCHIP

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

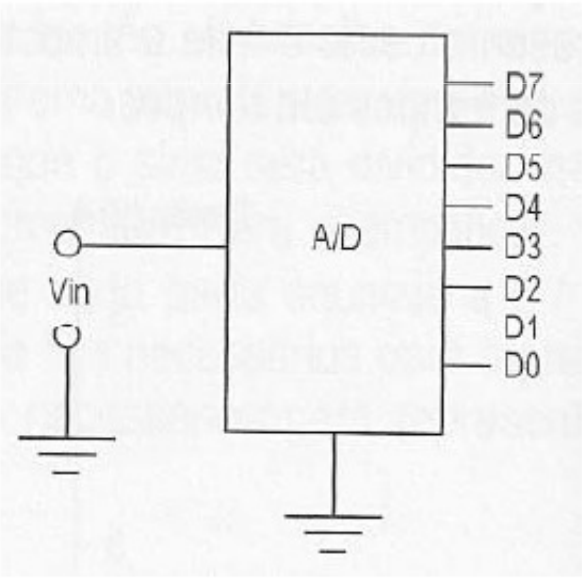
Jan / 2011

Conversor A/D

- Símbolo do
- Conversor A/D

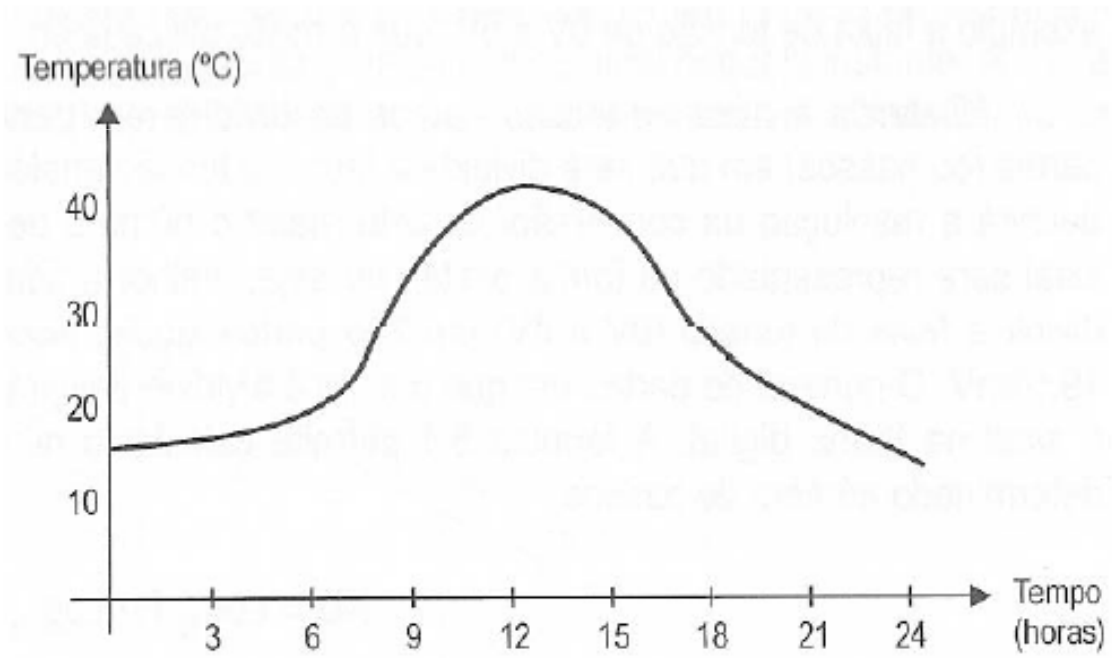


- Diagrama de blocos



Conversor A/D

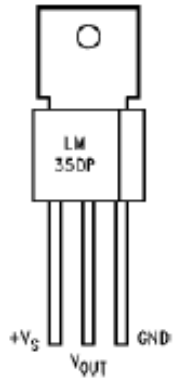
- Grandeza analógica
  - Ex.: temperatura ao longo do dia



Conversor A/D

- Transdutor
  - Converte uma grandeza não-elétrica em uma grandeza elétrica (tensão ou corrente)
  - Ex.: sensor de temperatura LM35D

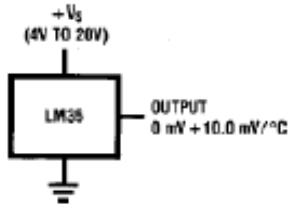
TO-202
Plastic Package



TL/H/5516–24

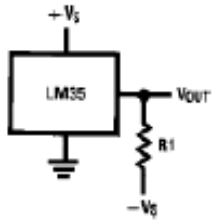
Order Number LM35DP
See NS Package Number P03A

Typical Applications



TL/H/5516–3

FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)



TL/H/5516–4

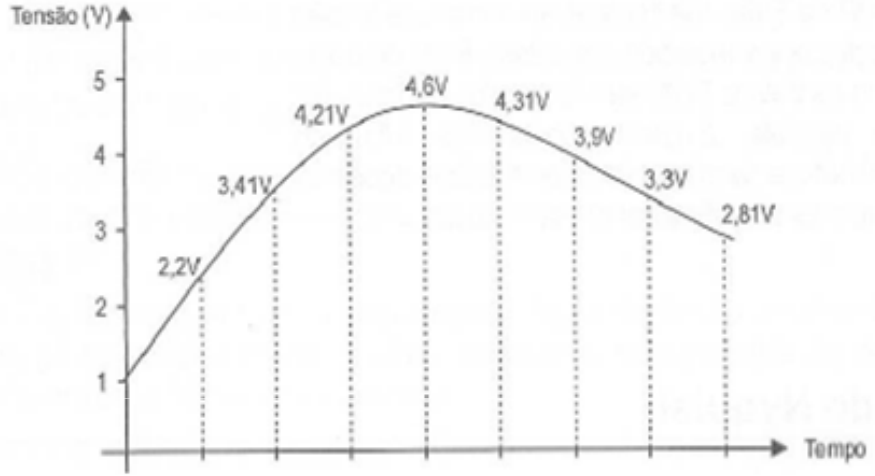
Choose $R_1 = -V_S/50\ \mu A$

$V_{OUT} = +1{,}500$ mV at +150°C
$= +250$ mV at +25°C
$= -550$ mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

Fonte: SeMICONDUCTOR, National. LM35/LM35A/LM35C/LM35CA/LM35D Precision Centigrade Temperature Sensors. Disponível em: http://www.national.com/ds/LM/LM35.pdf.

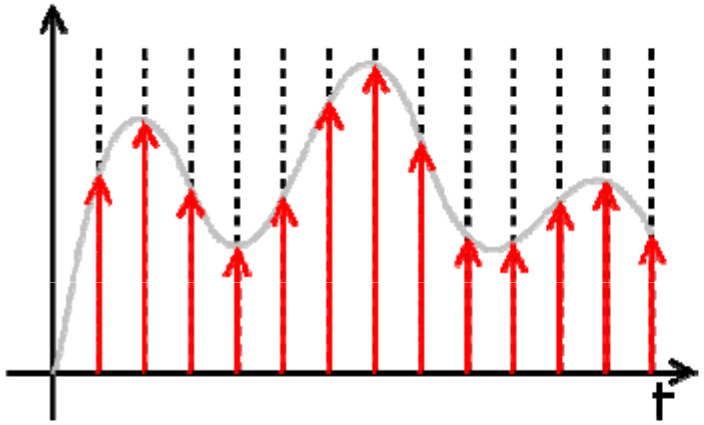
Conversor A/D



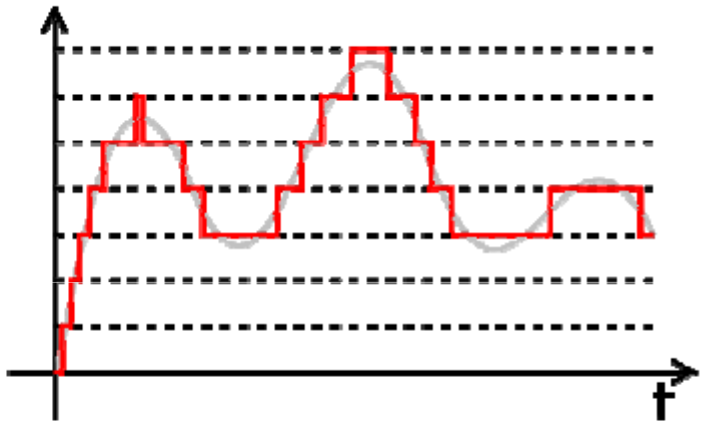
Partes	Tensão	Bits	Hexa
255	4,98V	11111111	FFh
254	4,96V	11111110	FEh
253	4,94V	11111101	FDh
.	.	.	.
.	.	.	.
5	97,65mV	00000101	05h
4	78,12mV	00000100	04h
3	58,59mV	00000011	03h
2	39,06mV	00000010	02h
1	19,53mV	00000001	01h
0	0V	00000000	00h

Conversor A/D

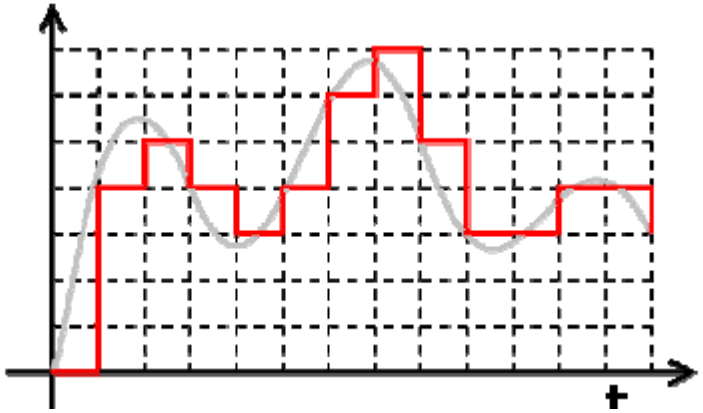
- Sinal amostrado



- Sinal quantizado



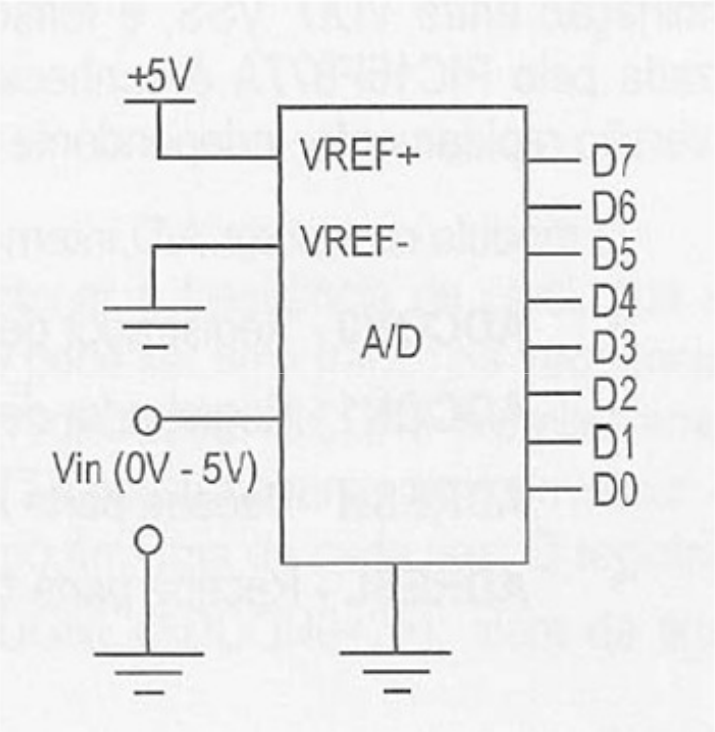
- Sinal digital



Tensão de Referência

- Tensão de referência VREF+
- Tensão de referência VREF-

A diferença entre as tensões aplicadas em VREF+ e VREF-, determina a faixa de tensão que será convertida em digital.



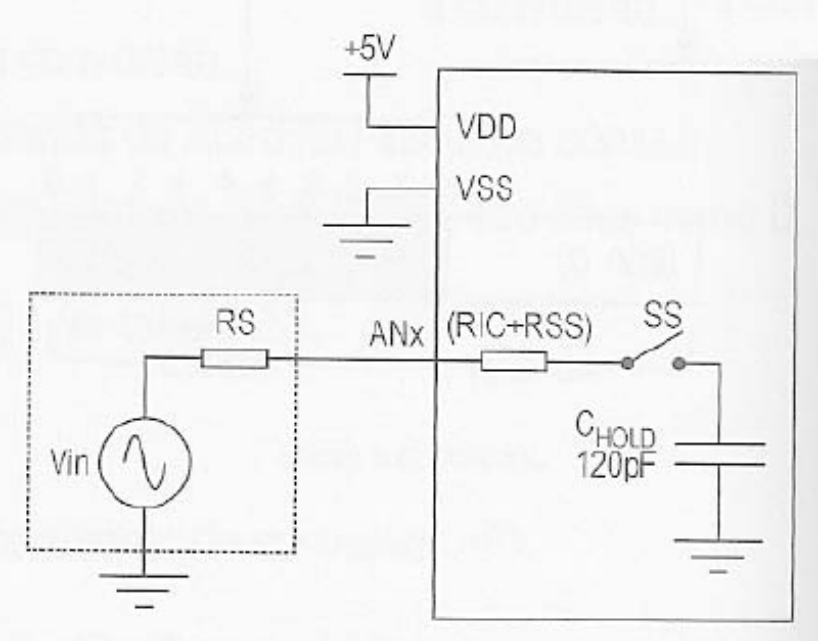
- Faixa de tensão analógica (VREF+) – (VREF-) a ser convertida em digital

Conversor A/D

- Resolução da conversão:
  - Ex.: (5V – 0V) / 256 = 19,53mV
- Número de bits para representar o sinal digital:
  - Ex.: 8 bits, pois 2^8 = 256

Tempo de Aquisição

- O PIC18F4520 no mínimo 2,4us para um Vin com RS=2,5kΩ (máximo recomendado pelo fabricante), à temperatura de 85º C;
- Por segurança, sugere-se um tempo de aquisição mínimo de 10us;



Tempo de Aquisição

- Para calcular o tempo de aquisição exato para uma determinada aplicação, deve-se consultar o datasheet do PIC18F4520, no qual encontra-se uma fórmula que permite efetuar com exatidão o cálculo do tempo de aquisição.

Conversor A/D do PIC18F4520

- O PIC18F4520 possui um módulo conversor A/D com 13 entradas analógicas que podem ser chaveadas para ter acesso, uma de cada vez, ao conversor;
- O conversor A/D do PIC18F4520 possui uma resolução de 10 bits;
- O conversor A/D pode operar mesmo com o PIC no modo SLEEP;
- Para isso, o sinal de clock do conversor A/D (CAD) deve ser derivado do oscilador RC interno do PIC;

Registradores FSR de Conversão A/D

- Existem três registradores no PIC18F4520 envolvidos com o recurso de conversão A/D:
  - ADCON0
    - Controla a operação do módulo conversor A/D;
  - ADCON1
    - Configura a função dos pinos de entrada do conversor;
  - ADCON2
    - Configura a origem do clock, a aquisição programada e a justificação;

Registradores SFR de Conversão A/D

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

Registradores – ADCON0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-2 **CHS<3:0>:** Analog Channel Select bits

0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)(1,2)
0110 = Channel 6 (AN6)(1,2)
0111 = Channel 7 (AN7)(1,2)
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Channel 11 (AN11)
1100 = Channel 12 (AN12)
1101 = Unimplemented)(2)
1110 = Unimplemented)(2)
1111 = Unimplemented)(2)

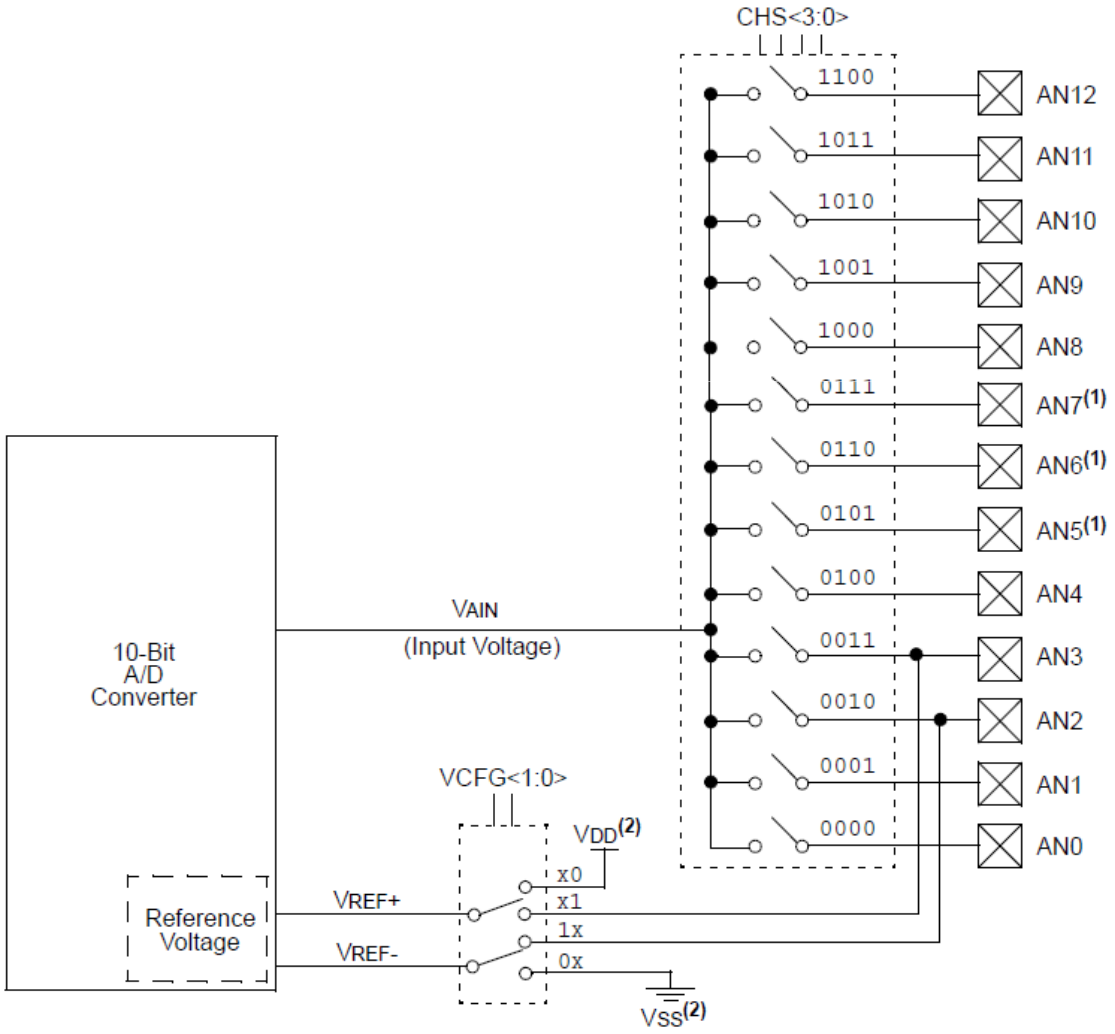
bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

When ADON = 1:
1 = A/D conversion in progress
0 = A/D Idle

bit 0 **ADON:** A/D On bit

1 = A/D Converter module is enabled
0 = A/D Converter module is disabled

Registradores – ADCON0



Registradores – ADCON1

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-q(1)	R/W-q(1)	R/W-q(1)
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **VCFG1:** Voltage Reference Configuration bit (V_{REF-} source)
1 = V_{REF-} (AN2)
0 = V_{SS}

bit 4 **VCFG0:** Voltage Reference Configuration bit (V_{REF+} source)
1 = V_{REF+} (AN3)
0 = V_{DD}

bit 3-0 **PCFG<3:0>:** A/D Port Configuration Control bits:

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111(1)	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

Registradores – ADCON2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified
0 = Left justified

bit 6 **Unimplemented:** Read as '0'

bit 5-3 **ACQT<2:0>:** A/D Acquisition Time Select bits
111 = 20 TAD
110 = 16 TAD
101 = 12 TAD
100 = 8 TAD
011 = 6 TAD
010 = 4 TAD
001 = 2 TAD
000 = 0 TAD(1)

bit 2-0 **ADCS<2:0>:** A/D Conversion Clock Select bits
111 = FRC (clock derived from A/D RC oscillator)(1)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC (clock derived from A/D RC oscillator)(1)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

Resultado de uma conversão

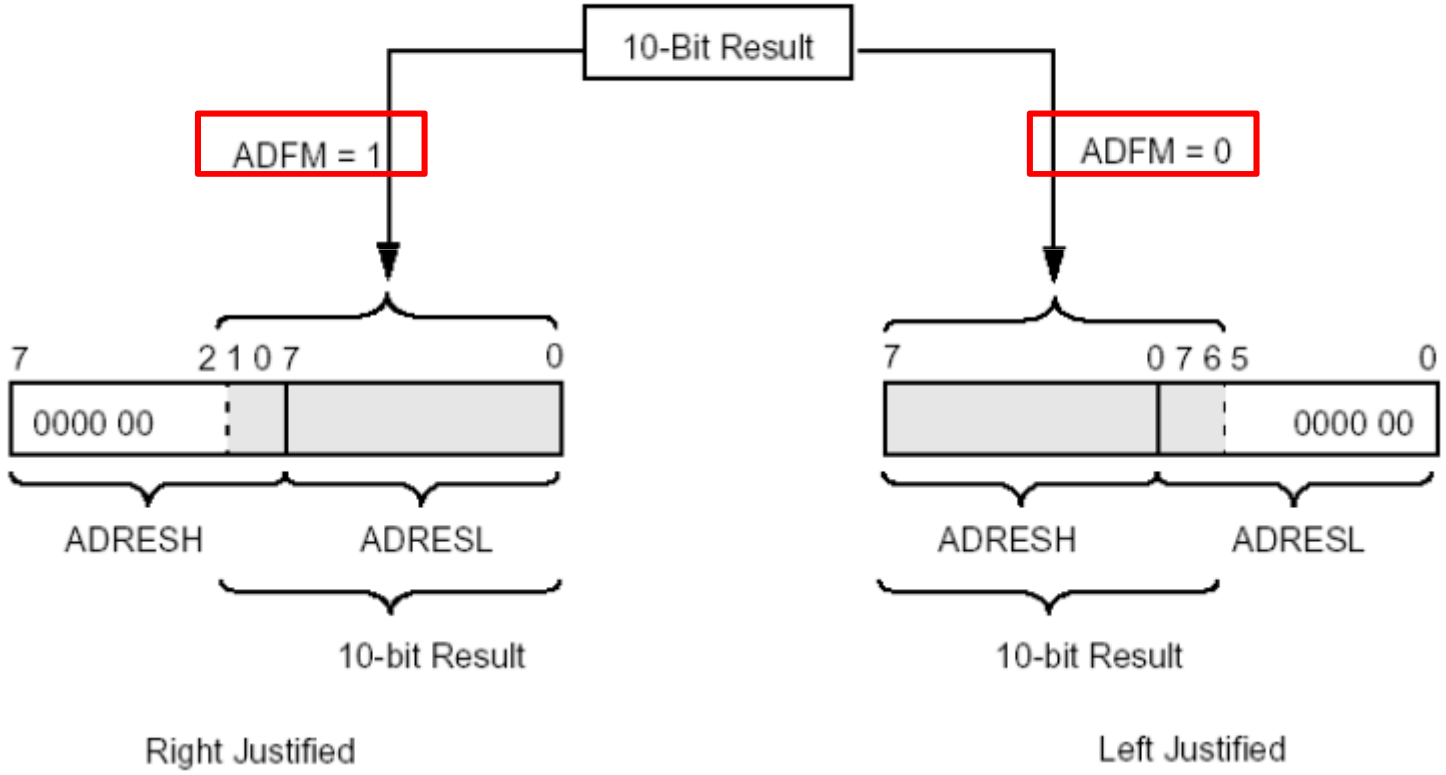
1. Para dar início a conversão, é necessário setar o bit G0/DONE (ADCON0<2>);
2. Quando a conversão for finalização, o bit G0/DONE é apagado automaticamente pelo hardware;
3. Sendo também setado automaticamente, o bit ADIF (PIR<6>);
4. A amostra resultante estará disponível nos registradores ADRESH:ADRESL;

Resultado de uma conversão

6. O resultado da conversão do PIC18F4520 é de 10 bits, mas estará disponível em 2 registradores de 8 bits;
7. A sobra de bits permitiu aos projetistas deslocar o resultado para esquerda ou para direita, procedimento denominado justificação do resultado;

Justificação do Resultado

- ADFM (ADCON2<7>)



Passos para a conversão A/D

1. Configuração do módulo A/D
   - Configurar
     - Pinos de entrada analógica
     - Tensões de referência
     - Pinos de I/O digital (ADCON1)
2. Selecionar o canal de entrada analógica (ADCON0)
3. Selecionar a freqüência do sinal de clock do conversor A/D (ADCON2)
4. Justificar o resultado da conversão (ADCON1);
5. Ligar o módulo A/D

Passos para a conversão A/D

5. Se necessário, configurar a interrupção do módulo A/D
   A. Reset o bit ADIF
   B. Sete o bit ADIE
   C. Sete o bit GIE
6. Aguardar o tempo de aquisição (10us);
7. Iniciar a conversão:
   - Setar o bit GO/DONE (ADCON0<2>)
8. Aguardar o tempo de conversão

Passos para a conversão A/D

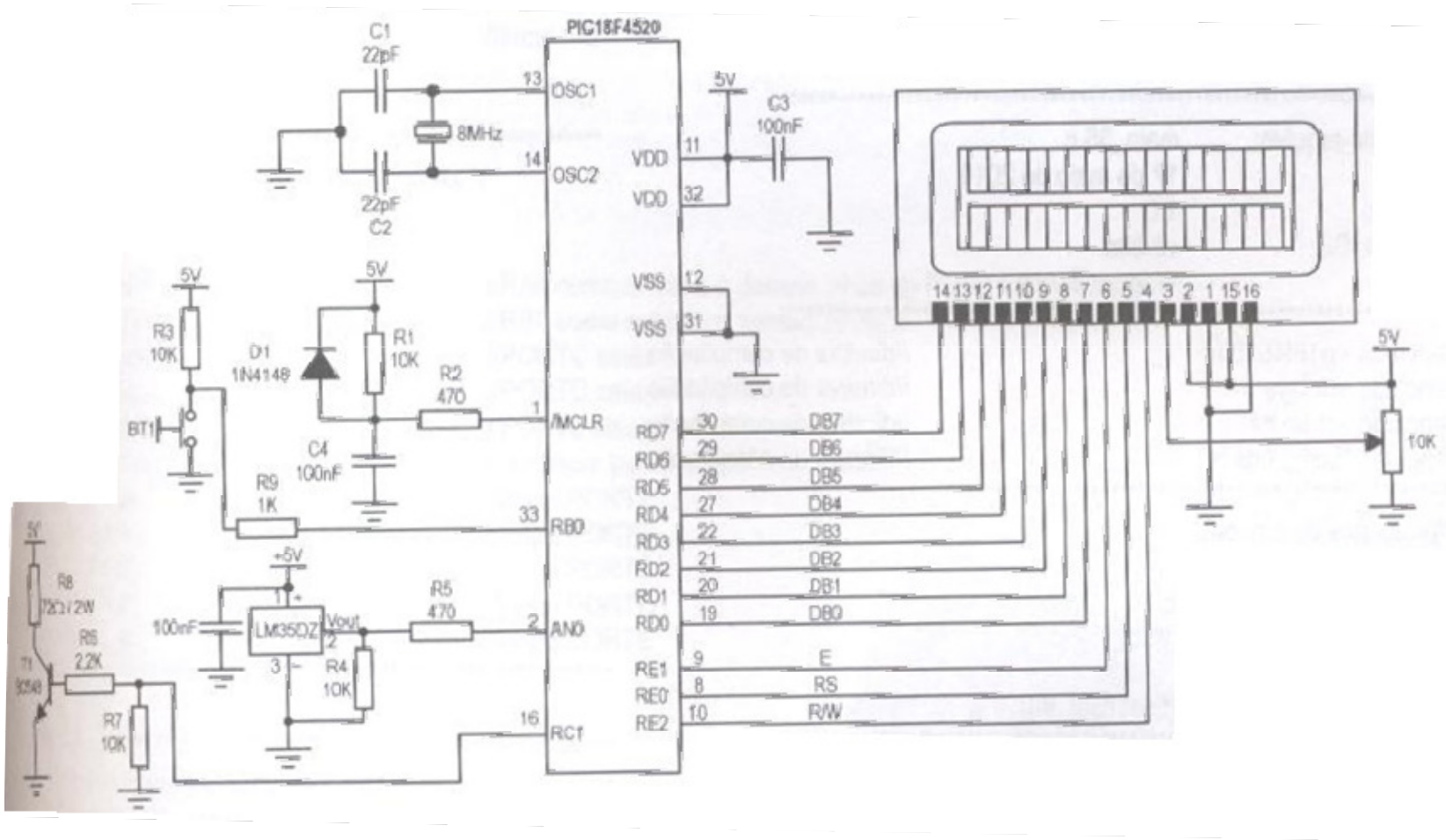
- verificando se o valor do bit GO/DONE é zero (sinalizando o fim de conversão A/D);
- ou aguardar pela interrupção gerada pelo módulo A/D ao fim da conversão;

9. Ler o par de registradores (ADRESH:ADRESL);
10. Aguardar no mínimo 2 x TAD para dar início a uma nova conversão;
11. Limpar o bit ADIF se a interrupção do conversor A/D estiver sendo utilizada;

Interrupção do Conversor A/D

- Para habilitar a interrupção do conversor A/D sem os níveis de prioridade:
  1. Ligar a chave geral de interrupção, setando-se o bit GIE (INTCON<7>);
  2. Habilita a interrupção dos periférico, setando-se o bit PEIE (INTCON<6>);
  3. Ligar a chave individual do módlo de conversor A/D, setando-se o bit ADIE (PIE1<6>);
  4. A flag de fim da conversão deve ser apagada manualmente dentro da função de tratamento da interrupção.

Esquema Elétrico



Conversor A/D (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_35.c

  Arquivo principal responsável por exibir a cada segundo no LCD o texto seguinte, em que o x representa o valor da tensão aplicada no pino AN0 convertida em graus Celsius;

  Temp = xx,x ℃

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

Conversor A/D (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

Conversor A/D (Código-fonte)– 1

```
/*****************************************************************
Esta biblioteca contém um conjunto de funções que permitem ao microcontrolador
se comunicar com o LCD controller HD44780.
//****************************************************************/
#include <p18cxxx.h>                  //diretiva de compilação
#include<delays.h>                     //diretiva de compilação
#include "LCD_8bits.h"                 //diretiva de compilação
/*****************************************************************
A função IniciaLCD() recebe como argumento um valor que irá inializar o LCD com:

valor = 1 -> inicializa o LCD com uma linha
valor != 1 -> inicializa o LCD com linha dupla

Quando o programa retorna ao ponto de chamada, o LCD mostra o cursor piscando
na primeira posição da primeira linha.
//****************************************************************/
```

LCD_8bits.c

Conversor A/D (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

Conversor A/D (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

Conversor A/D (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

Conversor A/D (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

Conversor A/D (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

Conversor A/D (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_35.c

Conversor A/D (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 Configura_AD   (void);
16 int Conv_AD    (void);
17 void Atual_LCD (void);
18 char *Conv_Float_String(float float_in);
19 //*****************************************************************
20 //variáveis globais
21 char buf [17];
22 //*****************************************************************
23 //definições
24 #define   AJUSTA_DECIMAL   10            //precisão de uma casa decimal
25 #define Botao1          PORTBbits.RB0
26 #define Resistencia       PORTCbits.RC1
27 //*****************************************************************
```

Main_35.c

Conversor A/D (Código-fonte) - 2

```
28  void main(void)                        //função main
29  {
30  float x=0;                            //declaração  de variável local inicializada
31  char y=0;                             //declaração  de variável local inicializada
32  char i=0;                             //declaração  de variável local inicializada
33  float temp=0;                         //declaração  de variável local inicializada
34  int dly=0;
35  //***********************************
36     Inic_Regs ();                      //configurar SFRs
37  //*
38     IniciaLCD (2);                      //inicializar LCD controller HD44780
39     TestPixelsLCD();                   //teste no LCD - acende todos os pixels.
40     EscInstLCD(0x0C);                          //desativa cursor
41     while(TesteBusyFlag());                       //espera LCD controller terminar
42  //***********************************
43  //delay de 3 segundos
44     for(dly=0;dly<600;dly++)           //comando de iteração
45     {
46        _Delay5ms();                    //delay de 5ms
47     }
48  //***********************************/
49  Configura_AD();
50  //***********************************
```

Main_35.c

Conversor A/D (Código-fonte) - 3

```
51  //rotina principal
52     while (1)
53     {
54        for(i=0;i<=49;i++)            //laço de iteração
55        {
56           if(!Botao1) Resistencia =1;   //liga a resistência se o botão estiver p
57           else Resistencia =0;       //senão, desliga resistência
58           x = (float)(Conv_AD()/2);   //chamada à função com retorno de valor
59           temp += x;                 //obtém temperatura medida
60           Delay1KTCYx(40);           //delay de 20ms
61        }
62        Conv_Float_String(temp/50);       //chamada à função
63        Atual_LCD ();                 //chamada à função
64        temp=0;                       //temp = 0
65     }
66  }
67  //*********************************************************************
68  /*Esta funcao inicializa os resgistradores SFRs.*/
69   void Inic_Regs (void)
70  {
71     TRISA = 0x01;                    //PORTA saída
72     TRISB = 0x01;                    //PORTB saída
73     TRISC = 0x00;                    //PORTC saída
74     TRISD = 0x00;                    //PORTD saída
75     TRISE = 0x00;                    //PORTE saída
76     ADCON1 = 0x0F;                    //configura pinos dos PORTA e PORTE como digi
77     PORTA = 0;                       //limpa PORTA
78     PORTB = 0;                       //limpa PORTB
79     PORTC = 0;                       //limpa PORTC
80     PORTD = 0x00;                    //apaga displays
81     PORTE = 0;                       //limpa PORTE
82  }//*********************************************************************
```

Main_35.c

Conversor A/D (Código-fonte) - 4

```
83  //esta função configura o conversor A/D
84  void Configura_AD   (void)
85  {
86     ADCON0=0b00000001;              /*canal AN0 selecionado<5:2>
87                                  Módulo conversor ligado<1>*/
88     ADCON1=0b00001110;              /*Vref- = VSS<5>
89                                  Vref+ = VDD<4>
90                                  pino RA0/AN0 analógico e demais pinos digitais<3:(
91     ADCON2=0b10101001;              /*resultado justificado à direita<7>
92                                  Tempo de aquisição de 12TAD<5:3>  TAD = 1µs<2:0>*/
93  }//*******************************************************************
94  //esta função efetua uma ocnversão A/D
95  int Conv_AD (void)
96  {
97  int Result_AD;                     //declaração de variável local
98     ADCON0bits.GO = 1;               //inicia conversão
99     while (ADCON0bits.GO);            //aguarda finalizar conversão
100    Result_AD = (((int)ADRESH)<<8)|(ADRESL);   //obtém valor da conversão
101    return Result_AD;                //retorna valor da conversão
102 }//*******************************************************************
103 /*******************************************************************/
104 /*Conversão de float para ASCII. Esta função converte valores float
105 na faixa de -65535.998 a +65535.998 em uma string.
106 //*******************************************************************/
107 char *Conv_Float_String(float float_in)
108 {
109 unsigned int parte_inteira;          //declaração de variável local
110 unsigned int parte_decimal;          //declaração de variável local
111 char sinal, *pt;                   //declaração de variável local
112 unsigned char x, y;                  //declaração de variável local
113 unsigned char z = 0xB2;
```

Main_35.c

Conversor A/D (Código-fonte) - 5

```
115  //*********************************************************************
116     if(float_in < 0)                //valor é menor que 0?
117     {
118        sinal = '-';                 //sim, sinal negativo
119        float_in = float_in * (-1);      //inverte o sinal de float_in
120     }
121     else sinal = ' ';                //não, sinal positivo
122  //*********************************************************************
123     parte_inteira = float_in;          //resgata parte inteira do valor
124  //*********************************************************************
125     //resgata parte fracionária do valor
126     parte_decimal =(unsigned int)((float_in - parte_inteira) * AJUSTA_DECIMAL);
127     //converte valor em string e armazena no vetor buf
128     sprintf(buf," Temp =%c%u.%01u*C",sinal,parte_inteira,parte_decimal);
129     pt = buf;                        //passa para o ponteiro pt o endereço de buf
130     return (pt);                     //retorna o endereço de buf.
131  }//*********************************************************************
132  //esta função atualiza o LCD
133  void Atual_LCD (void)
134  {
135     //limpa display e mostra cursor piscando na primeira posição da primeira linh
136     EscInstLCD(0x01)
137     while(TesteBusyFlag()); //espera LCD controller terminar de executar instruçã
138
139     EscStringLCD(buf);       //escreve string no LCD
140     while(TesteBusyFlag()); //espera LCD controller terminar de executar instruçã
141  }//*********************************************************************
```

Próxima Aula

Aula 17

Periférico de Comunicação