Aula 17
Periférico de Comunicação

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

Introdução

- Em várias aplicações utilizando uC pode ser necessário a comunicação entre o UC e um ou mais dispositivos externos;
- As técnicas de comunicação podem ser divididas em duas categorias: paralela e serial;

Comunicação Serial X Paralela

- Serial:
  - Transmissão de dados mais simples;
  - Utiliza apenas um canal de comunicação;
  - Menor velocidade de transmissão;
- Paralela:
  - Transmissão de dados mais custosa e complexa;
  - Requer mais de um canal de comunicação;
  - Maior velocidade de transmissão;

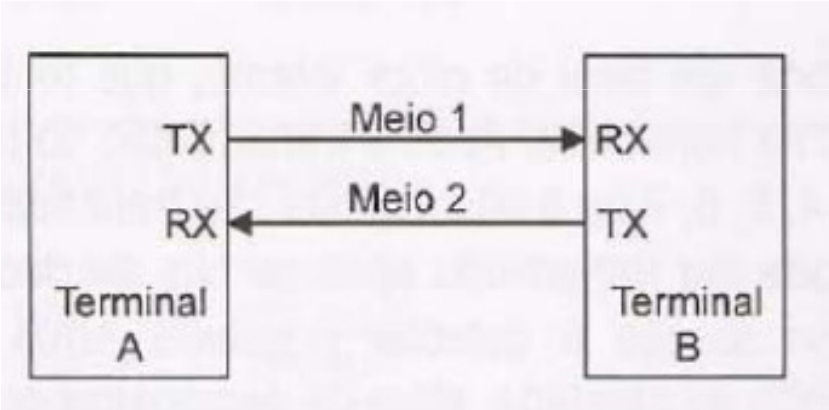
Comunicação Serial X Paralela

Característica	Serial	Paralelo
Velocidade	↓	↑
Custo	↓	↑
Imunidade a ruído	↑	↓
Distância	↑	↓

Transmissão Serial

- A Comunicação Serial pode ser classificada em duas categorias:

ASSÍNCRONO



SÍNCRONO

Transmissão Serial

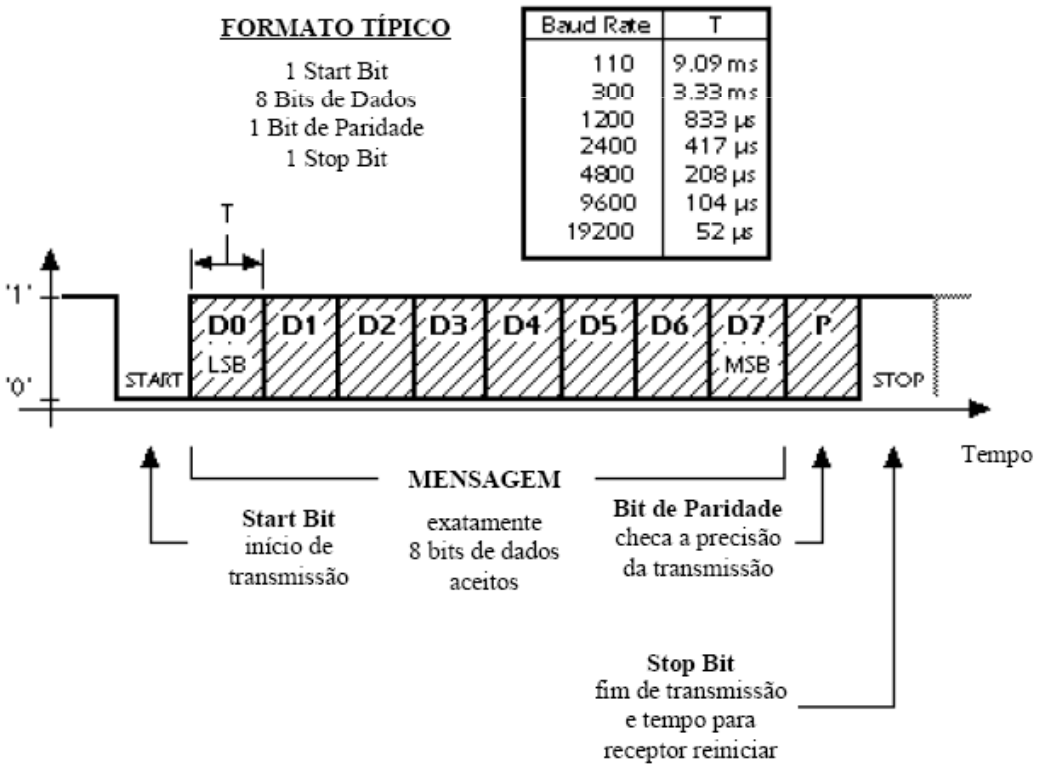
- Periférico que converte dados em paralelo para dados seriais.
  - Transforma bytes de informações em bits individuais passíveis de serem transmitidos.
- A operação inversa também é realizada:
  - Os bits recebidos são convertidos para bytes novamente.

Transmissão Serial

- Tipos de transferência:
  - **Simplex**: A informação é transmitida em um único sentido, ou seja, somente do transmissor para o receptor.
  - **Half-Duplex**: A informação é transmitida em ambos os sentidos, embora de forma alternada.
  - **Full-Duplex**: A informação é transmitida em ambos os sentidos de forma simultânea.
- Tipo de comunicação:
  - Broadcast;
  - Master/slave;
  - Ponto a ponto;

Sinais no tempo

- Bit de start;
- Palavra de dados;
- Bit de paridade: par ou ímpar;
- Bit de stop;



Interface Serial: UART

- UART
  - Universal Asynchronous Receiver Transmiter;
- È um protocolo de comunicação serial assíncrono desenvolvido em 1960 para permitir acomunicação ponto a ponto entre computadores e terminar remotos;
- A UART tornou-se tão popular que é amplamente utilizada até hoje em muitas aplicações;

Interface Serial: EUSART

- Praticamente todos os microcontroladores da série PIC18 vêm integrados com pelo menos um módulo de transmissão e recepção de dados serial;
- O PIC18F4520 possui um mpodulo de cmunicação serial chamado EUSART (Enhanced Universal Syncrhnous Receiver Transmiter);
- A principal evolução da EUSART é a capacidade de detectar automaticamente a taxa de transferência, um recurso chamado auto-baud detect;

Elementos de Configuração

- **Taxa de transmissão** (limitada a ~ 20Kb/s tx. de dados):
  - A velocidade de transmissão deve ser escolhida considerando-se a velocidade de recepção (medida em bauds).
- **Número de bits de dados:**
  - Sete bits se os dados a serem enviados estiverem no formato texto e oito se estiverem no formato binário.
- **Paridade:**
  - Indica quando não há uma correspondência exata entre a informação transmitida e recebida.
- **Start Bit e Stop Bit:**
  - Permite identificar o início e o fim da transmissão de um dado.

Padrão RS232

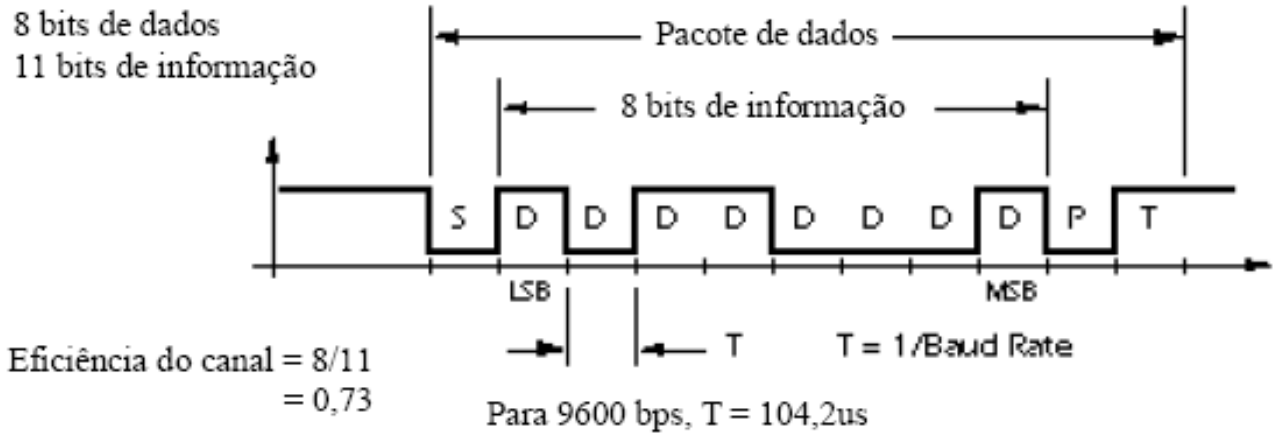
- Padrão de comunicação serial criado pela EIA (Electronics Industry Association)
  - RS: Recommended Standard
- Padrão elétrico e mecânico;
- Criado para conectar computador (DTE) em modem (DCE – Data Communication Equipment);
- Sinais de handshake (RTS, CTS, DTR, RI, ...);

Padrão RS232

- O padrão RS232 especifica:
  - Tensões
  - Temporização
  - Funções dos sinais
  - Um protocolo de comunicação;
  - Conectores mecânicos;
- Conhecido também como EIA232;

Taxa de Transferência (Baud Rate)

- A taxa de transferência refere-se à velocidade com que os dados são enviados através de um canal;
- É medido em transições elétricas por segundo.
- Uma taxa de 9600 bauds corresponde a uma transferência de 9600 dados por segundo, ou um período de aproximadamente, 104 ms (1/9600 s).



Taxa de Transferência (Baud Rate)

- Seleção da fórmula para cálculo do *baud rate;*

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-Bit/Asynchronous	Fosc/[64 (n + 1)]
0	0	1	8-Bit/Asynchronous	Fosc/[16 (n + 1)]
0	1	0	16-Bit/Asynchronous	
0	1	1	16-Bit/Asynchronous	Fosc/[4 (n + 1)]
1	0	x	8-Bit/Synchronous	
1	1	x	16-Bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGH:SPBRG register pair

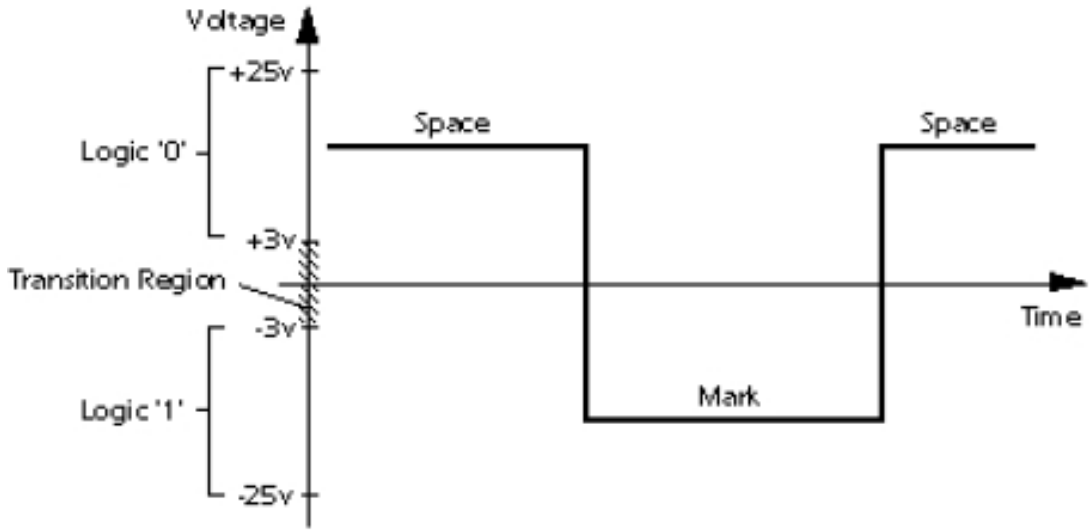
Taxa de Transferência (Baud Rate)

- Exemplo:

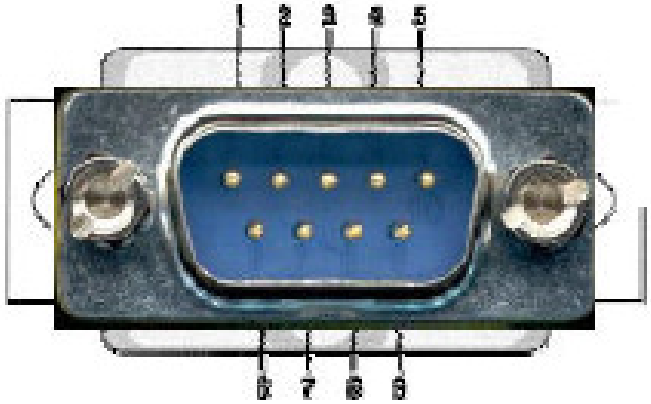
- Qual valor deve ser escrito no par de registradores SPBRGH:SPBRG para uma transmissão assíncrona, baud rate de 19200bps e uma Fosc de 8MHz?
- Deduzindo SPBRG na fórmula para 8 bits, temos:
- SPBRGH = Fosc / (M x baud rate) – 1
- Se BRGH = M = 64
  - SPBRGH = 5,51 aprox. = 6, logo Baud Rate = 17857,14
- Se BRGH = M = 16
  - SPBRGH = 25,04 aprox. = 25, logo Baud Rate = 19230,76

Tensões dos sinais

- "1" lógico: Tensões de -3V a -25V em relação ao pino de terra;
- "0" lógico Tensões de +3V a +25V em relação ao pino de terra;
- Para o intervalo de tensões entre -3V e +3V não é associado um estado do sinal, sendo uma região de transição;

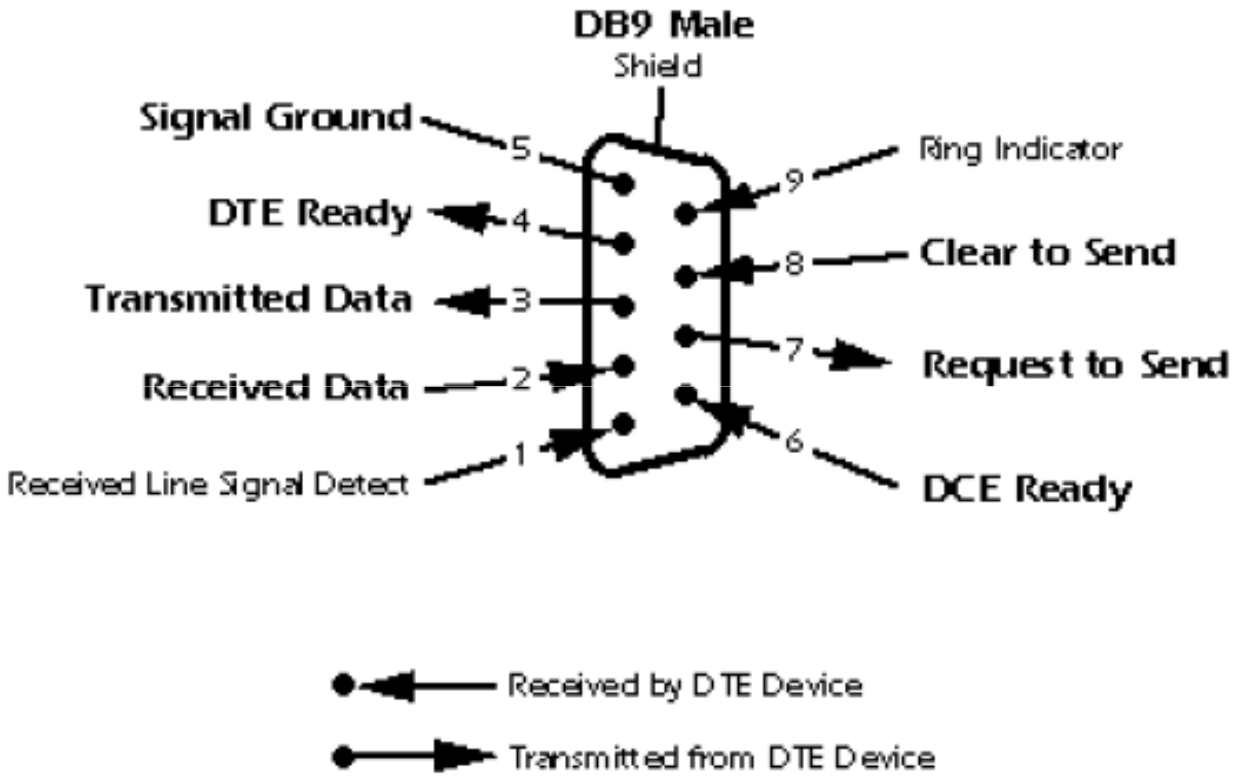


Pinos do conector DB9

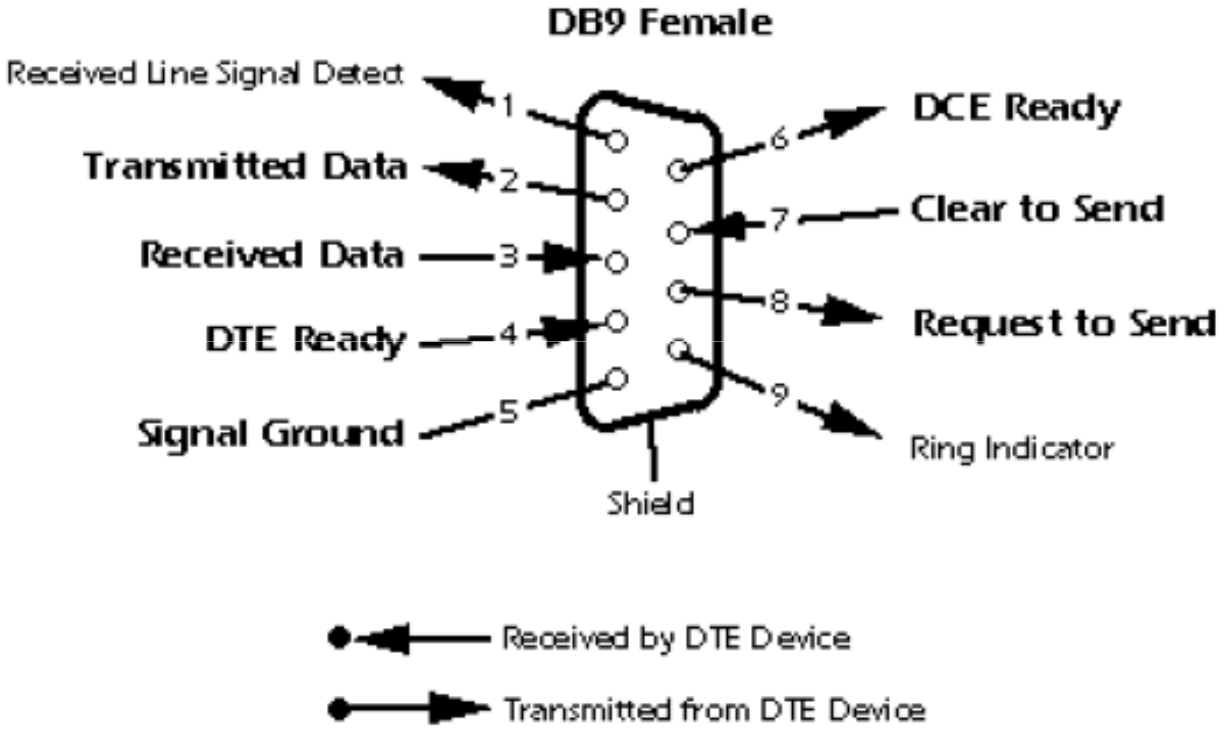


Pin	Name	Dir	Description
1	CD	←	Carrier Detect
2	RXD	←	Receive Data
3	TXD	→	Transmit Data
4	DTR	→	Data Terminal Ready
5	GND	—	System Ground
6	DSR	←	Data Set Ready
7	RTS	→	Request to Send
8	CTS	←	Clear to Send
9	RI	←	Ring Indicator

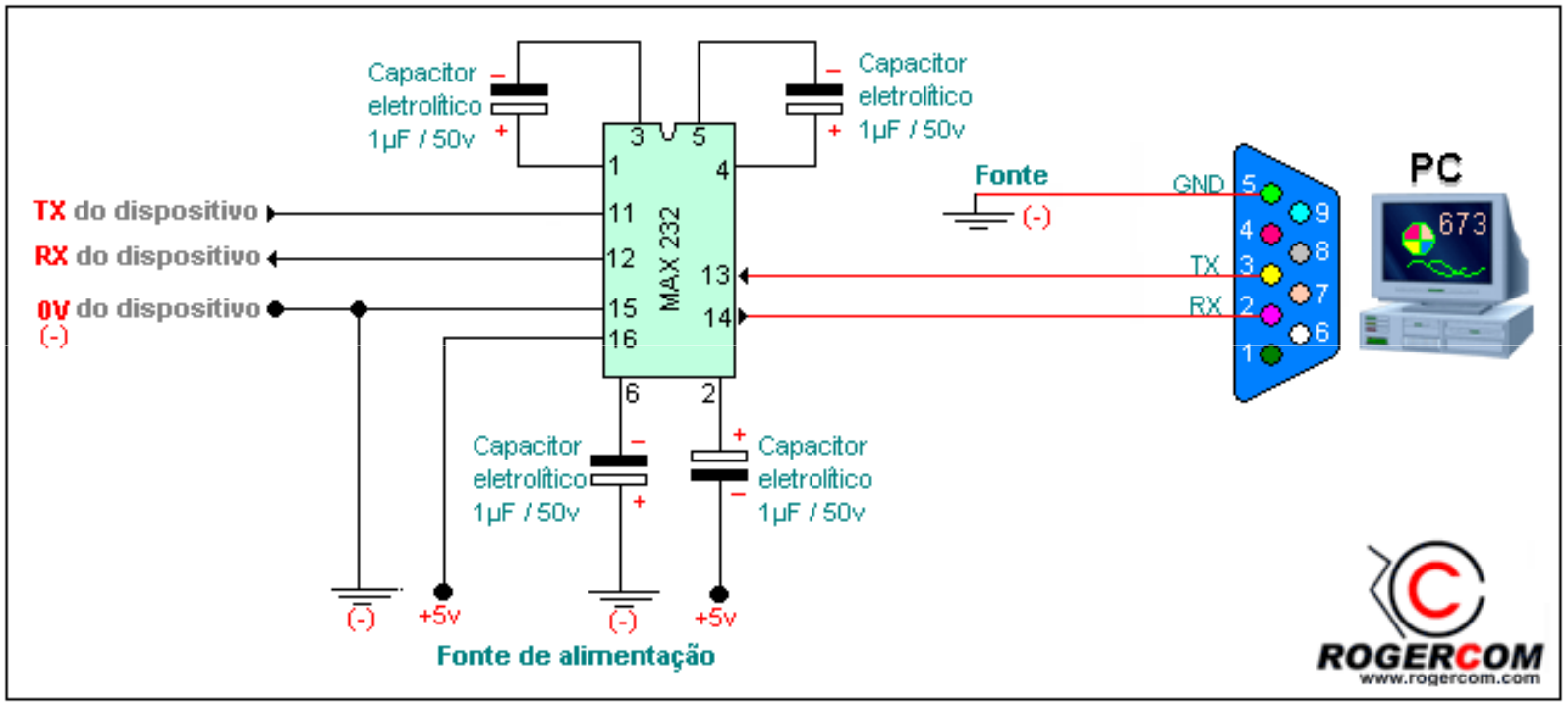
Pinos do conector DB9



Pinos do conector DB9

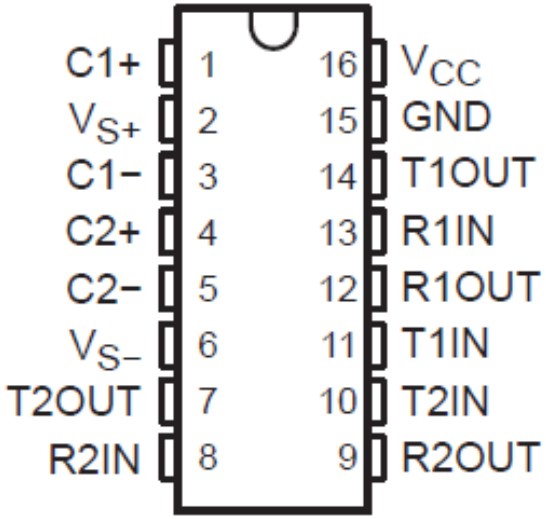


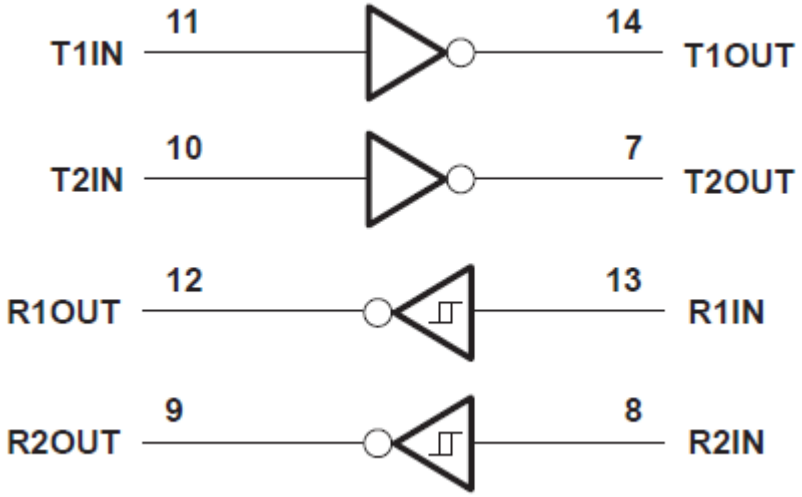
Conexão RS-232 entre *uC* e o PC



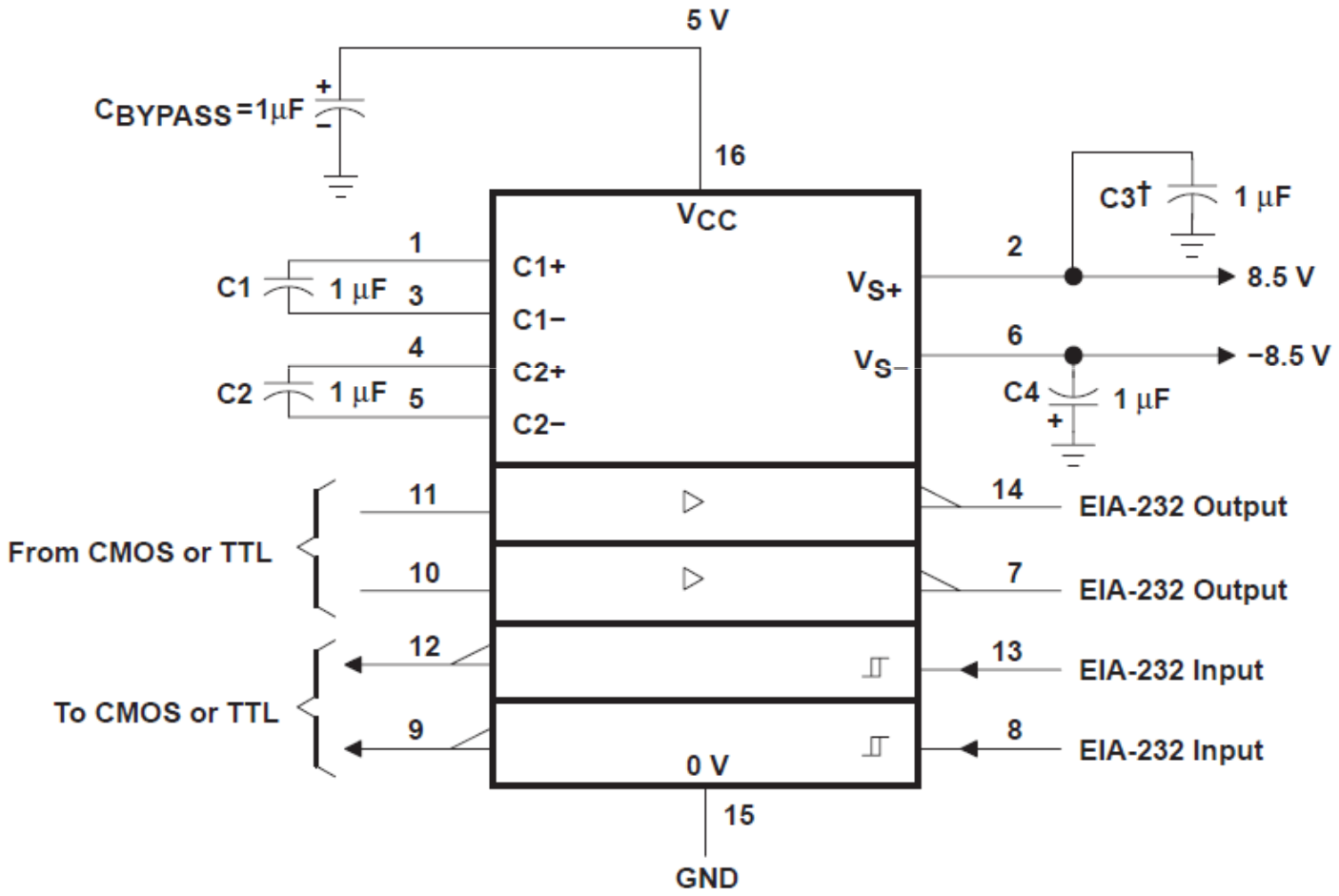
RS-232 driver/receiver (MAX232)

- CI para:
  - Ajuste dos níveis lógicos;
  - Inversão para a lógica correta;





RS-232 driver/receiver (MAX232)

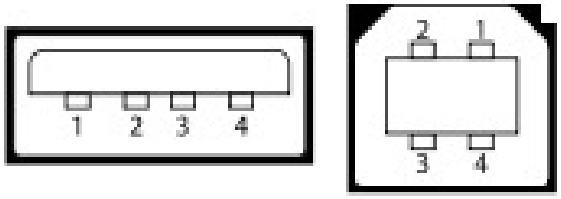


Comparação RS232, RS422 e RS485

Especificação	RS-232	RS-422	RS-485
Modo de operação	Ponto-a-ponto	Master/Slave	Master/Slave
Tipo de transmissão	Linha não balanceada	Linha balanceada	Linha balanceada
Controle tristate da linha	não	opcional	sim
Comprimento máximo de cabo	15 m	1.200 m	1.200 m
Taxa de transmissão máxima	20 kbps	10 Mbps	10 Mbps
Driver/Receiver numa linha	1/1	1/10	1/32 (1/256)

USB (Universal Serial Bus)

- USB (Universal Serial Bus): taxas de transmissão da ordem de 12 Mbits/s;
- Padrão de Barramento Serial para Interfaces com Dispositivos de E/S
- Conectores:



Pino	Função
1	VBUS (4,75 – 5,25)
2	D+
3	D-
4	GND

Serial Síncrona: SPI e I2C

- SPI
  - Desenvolvido pela *Motorola*, é útil em comunicaçao full-duplex e adequado em aplicações nas quais se deseja transferir fluxos de dados, tais como na interface com chips periféricos, memórias e até mesmo cartões de memória;
- I2C
  - Desenvolvido na década de 80 pela Philips, tornou-se um padrão conhecido e amplamente utilizado em projetos de circuitos eletrõnicos. Sua principal finalidade é permitir a comunicação entre dispositivos *onboard*;

Comunicação serial no PIC18F4520

- Parâmetros mínimos e máximos da UART

	PIC18F4520	Porta COM do PC
Baud Rate	110 a 1.250.000 bps	110 a 921.600 bps
No de Bits	8 a 9	4 a 8
Bit de paridade	Não suporta	Par, impar, marca ou espaço
Stop bit	1	1, 1.5 ou 2
Controle de Fluxo	Não suporta	Hardware, XON/XOFF ou nenhum

Passos para implementação de uma recepção Serial Assíncrona

1. Inicializar o registrador SPBRG para o baud rate desejado;
2. Configurar os bits RC7:RC6 como entrada;
3. Habilitar a porta, SYNC = 0 e SPEN = 1;
4. Habilitar a interrupção se for desejado;
5. RX9 = 1 se a recepção for de 9 bits;
6. Habilitar a recepção, CREN = 1;
7. Se ocorrer uma interrupção, o flag de sinalização de interrupção, RCIF = 1;
8. Ler o registrador RCSTA para obter o nono bit (RX9D) ou para verificar se houve erro durante a recepção;
9. Ler os 8 bits de dados que se encontram em RCREG;
10. Se ocorreu algum erro, apagar e setar o bit CREN para que novos dados possam ser recebidos;

Passos para implementação de uma transmissão Serial Assíncrona

1. Inicializar o registrador SPBRG para o baud rate desejado;
2. Configurar os bits RC7:RC6 como entrada;
3. Habilitar a porta, SYNC = 0 e SPEN = 1;
4. Habilitar a interrupção se for desejado;
5. RX9 = 1 se a recepção for de 9 bits;
6. Habilitar o transmissor, TXEN = 1;
7. Se a transmissão for de 9 bits, escrever o nono em TXD9;
8. Carregar o dado a ser transmitido em TXREG (a transmissão terá início);
9. A transmissão estará finalizada quando o bit TRMT for setado pelo hardware ;ww

Registradores FSR da EUART

- Existem três registradores no PIC18F4520 envolvidos com o recurso de serial UART:
  - TXSTA
    - É um registrador de controle do módulo transmissor da EUSART;
  - RCSTA
    - É um registrador de controle do módulo receptor da EUSART;
  - BAUDCON
    - É um registrador de controle de operação;

Registradores SFR da EUART

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 – TXSTA

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

bit 3 **SENDB:** Send Break Character bit
Asynchronous mode:
1 = Send Sync Break on next transmission
0 = Sync Break transmission completed
Synchronous mode:
Don't care.

bit 2 **BRGH:** High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode.

bit 1 **TRMT:** Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th Bit of Transmit Data
Can be address/data bit or a parity bit.

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care.
Synchronous mode:
1 = Master mode (clock generated internally
0 = Slave mode (clock from external source)

bit 6 **TX9:** 9-Bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit(1)
1 = Transmit enabled
0 = Transmit disabled

bit 4 **SYNC:** EUSART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

Registradores – RCSTA

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 3 **ADDEN:** Address Detect Enable bit

Asynchronous mode 9-Bit (RX9 = 1):

1 = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set

0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit

Asynchronous mode 9-Bit (RX9 = 0):

Don't care.

bit 2 **FERR:** Framing Error bit

1 = Framing error (can be cleared by reading RCREG register and receiving next valid byte)

0 = No framing error

bit 1 **OERR:** Overrun Error bit

1 = Overrun error (can be cleared by clearing bit, CREN)

0 = No overrun error

bit 0 **RX9D:** 9th Bit of Received Data

This can be address/data bit or a parity bit and must be calculated by user firmware.

Registradores – RCSTA

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit

1 = Serial port enabled (configures RX/DT and TX/CK pins
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-Bit Receive Enable bit

1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit

Asynchronous mode:
Don't care.

Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.

Synchronous mode – Slave:
Don't care.

bit 4 **CREN:** Continuous Receive Enable bit

Asynchronous mode:
1 = Enables receiver
0 = Disables receiver

Synchronous mode:
1 = Enables continuous receive until enable bit, CREN, is
0 = Disables continuous receive

Registradores – BAUDCON

R/W-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

bit 3 **BRG16:** 16-Bit Baud Rate Register Enable bit

1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG
0 = 8-bit Baud Rate Generator – SPBRG only (Compatible mode), SPBRGH value ignored

bit 2 **Unimplemented:** Read as '0'

bit 1 **WUE:** Wake-up Enable bit

Asynchronous mode:
1 = EUSART will continue to sample the RX pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge
0 = RX pin not monitored or rising edge detected

Synchronous mode:
Unused in this mode.

bit 0 **ABDEN:** Auto-Baud Detect Enable bit

Asynchronous mode:
1 = Enable baud rate measurement on the next character. Requires reception of a Sync field (55h); cleared in hardware upon completion.
0 = Baud rate measurement disabled or completed

Synchronous mode:
Unused in this mode.

Registradores – BAUDCON

R/W-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

bit 7 **ABDOVF:** Auto-Baud Acquisition Rollover Status bit

1 = A BRG rollover has occurred during Auto-Baud Rate Detect mode (must be cleared in software)
0 = No BRG rollover has occurred

bit 6 **RCIDL:** Receive Operation Idle Status bit

1 = Receive operation is Idle
0 = Receive operation is active

bit 5 **RXDTP:** Data/Receive Polarity Select bit

Asynchronous mode:
1 = Receive data (RX) is inverted (active-low)
0 = Receive data (RX) is not inverted (active-high)

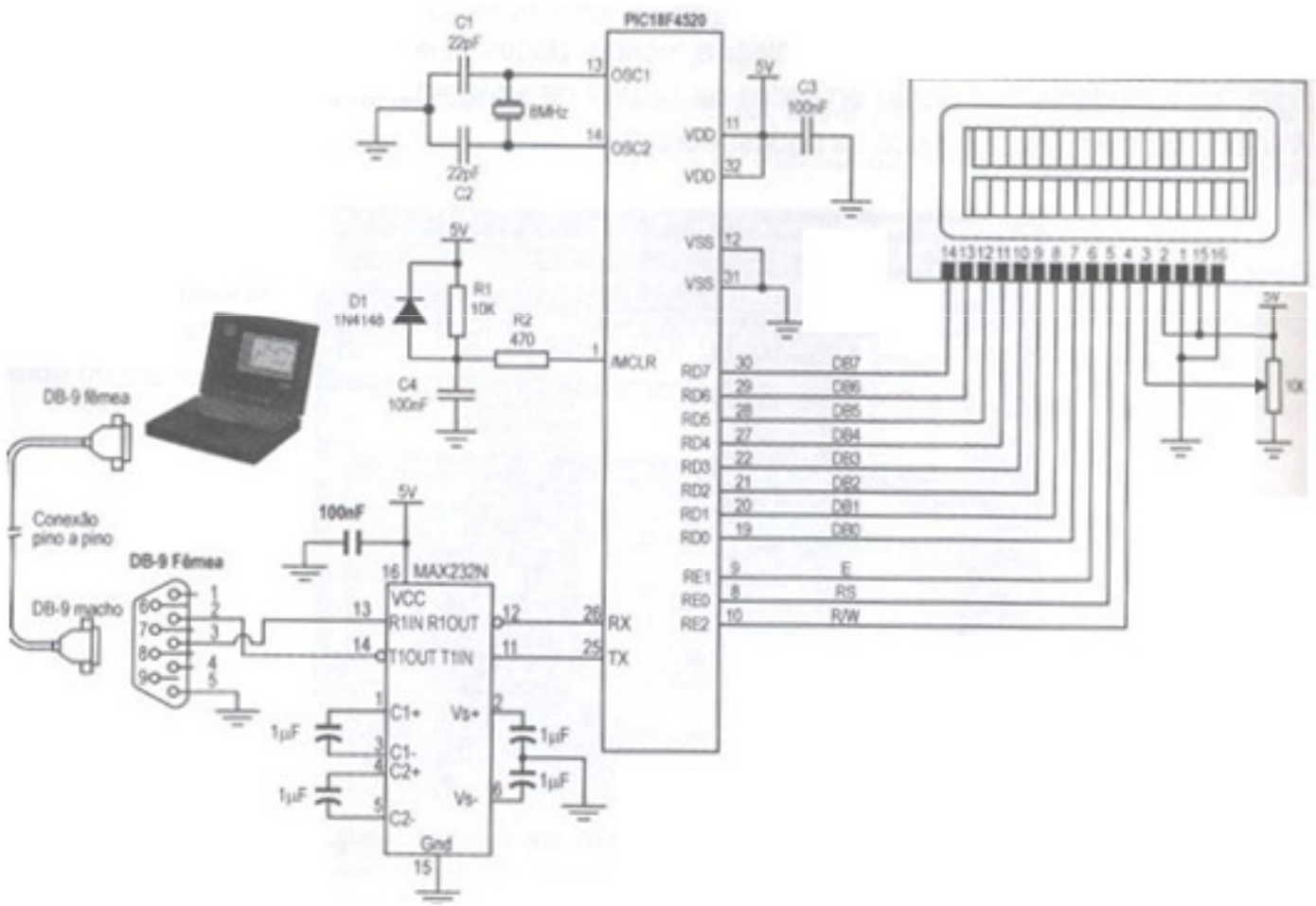
Synchronous mode:
1 = Data (DT) is inverted (active-low)
0 = Data (DT) is not inverted (active-high)

bit 4 **TXCKP:** Clock and Data Polarity Select bit

Asynchronous mode:
1 = Idle state for transmit (TX) is a low level
0 = Idle state for transmit (TX) is a high level

Synchronous mode:
1 = Idle state for clock (CK) is a high level
0 = Idle state for clock (CK) is a low level

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_39.c

  Arquivo principal responsável por receber uma um dado pela UART, retransmitir e, na seqüência, enviar para o LCD;

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 EUART (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 EUART (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 EUART (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 EUART (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 EUART (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 EUART (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 EUART (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 EUART (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_39.c

Serial EUART (Código-fonte) - 1

```
8  #include <p18f4520.h>      //diretiva de compilação
9  #include "Lcd_8bits.h"      //diretiva de compilação
10 //*********************************************************************
11 //protótipos de funções
12 void Inic_Regs (void);
13 void Atual_LCD (char dado);
14 void Configura_UART (void);
15 void Transmite_UART (char dado);
16 //*********************************************************************
17 //variáveis globais
18 char count=0;
19 //*********************************************************************
20 void main(void)                    //função main
21 {
22 float x=0;                        //declaração  de variável local inicializada
23 int dly=0;                        //declaração  de variável local inicializada
24 char DADO;                        //declaração  de variável local inicializada
25 //***********************************
26    Inic_Regs ();                  //configurar SFRs
27 //***********************************
28    IniciaLCD (2);                  //inicializar LCD controller HD44780
29    TestPixelsLCD();               //teste no LCD - acende todos os pixels.
30
31    EscInstLCD(0x0C);               //desativa cursor
32    while(TesteBusyFlag());            //espera LCD controller terminar de executa
33 //***********************************
```

Main_39.c

Serial EUART (Código-fonte) - 2

```
31      EscInstLCD(0x0C);                 //desativa cursor
32      while(TesteBusyFlag());             //espera LCD controller terminar de executar
33    //*********************************
34    //delay de 3 segundos
35      for(dly=0;dly<600;dly++)          //comando de iteração
36      {
37          _Delay5ms();                //delay de 5ms
38      }
39    //*********************************
40      EscInstLCD(0x01);                 //desativa cursor
41      while(TesteBusyFlag());             //espera LCD controller terminar de executar
42    //*********************************/
43      Configura_UART();                 //configura UART com taxa de 2400bps
44    //*********************************/
45    //rotina principal
46      while(1)
47      {
48          while(!PIR1bits.RCIF);            //aguarda chegar um novo byte
49          if (RCSTAbits.FERR)               //houve erro de transmissão?
50          {
51              RCSTAbits.CREN = 0;           //sim, desabilita recepção
52              RCSTAbits.CREN = 1;           //habilita recepção
53          }
54          else                            //não houve erro de transmissão, transmite dado
55          {
56              DADO = RCREG;               //obtém caractere
57              Transmite_UART (DADO);          //transmite DADO
58              Atual_LCD(DADO);            //atualiza LCD
59          }
60      }
```

Main_39.c

Serial EUART (Código-fonte) - 3

```
65  void Inic_Regs (void)
66  {
67     TRISA = 0x00;                    //PORTA saída
68     TRISB = 0x00;                    //PORTB saída
69     TRISC = 0x00;                    //PORTC saída
70     TRISD = 0x00;                    //PORTD saída
71     TRISE = 0x00;                    //PORTE saída
72     ADCON1 = 0x0F;                    //configura pinos dos PORTA e PORTE
73     PORTA = 0;                       //limpa PORTA
74     PORTB = 0;                       //limpa PORTB
75     PORTC = 0;                       //limpa PORTC
76     PORTD = 0x00;                    //apaga displays
77     PORTE = 0;                       //limpa PORTE
78  }//*****************************************************************
79  //inicializa USART
80  void Configura_UART (void)
81  {
82     TRISCbits.TRISC7 = 1;          //configura pino RX como entrada
83     TRISCbits.TRISC6 = 1;          //configura pino TX como entrada
84     TXSTA = 0b00100100;             //transmissão habilitada<5>
85                               //transmissão assíncrona<4>
86                               //transmissão em alta velocidade<2>
87     RCSTA = 0b10010000;               //porta serial habilitada<7>
88                               //recepção contínua habilitada<4>
89     BAUDCON = 0b00000000;             //TX ocioso em nível alto<4>
90                               //gerador de baud rate de 8 bits<3>
91     SPBRG = 207;                  //2400bps
92  }//*********************************
```

Main_39.c

Serial EUART (Código-fonte) - 4

```
93  //transmite o caractere
94  void Transmite_UART (char dado)
95  {
96     TXREG = dado;                //inicia a transmissão
97     while(TXSTAbits.TRMT);          //aguarda transmissão terminar
98  }//***************************************************************************
99  //função  que  atualiza o LCD.
100 void Atual_LCD (char dado)
101 {
102    if(count==16)                 //se cursor chegou no final da primeira linha
103    {
104       EscInstLCD(0xC0);         //move cursor para a primeira posição da segunda linha
105       while(TesteBusyFlag());  //espera LCD controller terminar de executar instrução
106    }
107    else if(count==32)            //se cursor chegou no final da segunda linha
108    {
109       EscInstLCD(0x01);  //limpa display e mostra cursor piscando na primeira posição da
110       while(TesteBusyFlag());   //espera LCD controller terminar de executar instrução
111    }
112    EscDataLCD(dado);       //escreve caractere no LCD na posição apontada pelo cursor
113    while(TesteBusyFlag());  //espera LCD controller terminar de executar instrução
114    count+=1;                 //incrementa count
115 //***************************************************************
116 }
```

Serial EUART (Código-fonte) - Ex.10.2

- Transmissão de uma string:

```
1  #include <p18f4520.h>
2  #include "pic_simb.h"
3
4  #pragma config OSC = XT, WDT = OFF, MCLRE = ON
5  #pragma config DEBUG = OFF, LVP = OFF, PWRT = ON, BOREN = OFF
6
7  void MCU_init(void)
8  {
9     TRISCbits.TRISC6 = 0;       // configura RC6(TX) como saída
10    TXSTA = bTXEN | bBRGH;    // TXSTA = 0x24 -> TXEN=1 e BRGH=1
11    RCSTA = bSPEN;             // RCSTA = 0x80 -> SPEN=1
12    SPBRG = 25;   // configura a velocidade da EUSART (9600bps para Fosc = 4MHz)
13 }
14
15 void EUSART_envia_string(char *string)
16 {
17    while (*string)
18    {
19       while (!PIR1bits.TXIF);    // aguarda espaço no buffer de TX
20       TXREG = *string;          // envia um caractere
21       string++;              // avança para o próximo caractere
22    }
23 }
24
25 void main(void)
26 {
27    char str[11]="Ola Mundo!";
28    MCU_init();
29    EUSART_envia_string(str);
30    while(1);
31 }
```

Serial EUART (Código-fonte) - Ex.10.3

- Recepção de uma string:

```
1  #include <p18f4520.h>
2  #include "pic_simb.h"
3
4  #pragma config OSC = XT, WDT = OFF, MCLRE = ON
5  #pragma config DEBUG = ON, LVP = OFF, PWRT = ON, BOREN = OFF
6
7  #define L1      LATBbits.LATB0
8
9  #pragma interrupt EUSART_RX_ISR
10 void EUSART_RX_ISR(void)
11 {
12    TXREG = RCREG+1;   // retransmite o dado recebido (+1)
13    L1 = !L1;   // muda o estado do led L1
14 }
15
16 #pragma code isr_baixa = 0x0008
17 void ISR_baixa_prioridade(void)
18 {
19    if (PIR1bits.RCIF) _asm BRA EUSART_RX_ISR _endasm
20 }
21 #pragma code
22
23 void MCU_init(void)
24 {
25    TRISCbits.TRISC6 = 0;   // pino RC6(TX) como saída
26    TRISBbits.TRISB0 = 0;   // pino RB0 (led) como saída
27    TXSTA = bTXEN | bBRGH;    // TXSTA = 0x24 -> TXEN=1 e BRGH=1
28    RCSTA = bSPEN | bCREN;    // RCSTA = 0x80 -> SPEN=1 e CREN=1
29    SPBRG = 25;          // configura a velocidade da EUSART (9600 bps)
30    PIE1bits.RCIE = 1;       // habilita a interrupção de recepção da EUSART
31    INTCON = bGIE | bPEIE;    // habilita interrupções
32 }
```

Próxima Aula

Aula 18

Modulação PWM