NOTAS DE AULAS: MICROCONTROLADORES
Comunicação Serial com o PIC

Prof. João Perea Martins
Dep. De Computação, FC–UNESP
E-mail: joao.perea@unesp.br

1. Comunicação Serial

A interface de comunicação serial se comunica com a CPU de forma paralela, porém, ela se comunica com a serial de outro computador, utilizando um padrão serial, conforme exemplifica a figura 1.

Neste caso, a serial envia bit a bit da informação a ser transmitida. Na serial receptora, a informação é recebida bit a bit e quando receber o número de bits relativos a palavra de transmissão, que usualmente é 8 bits, então ela compõe uma informação (byte) e disponibiliza esta informação para a CPU da máquina receptora.

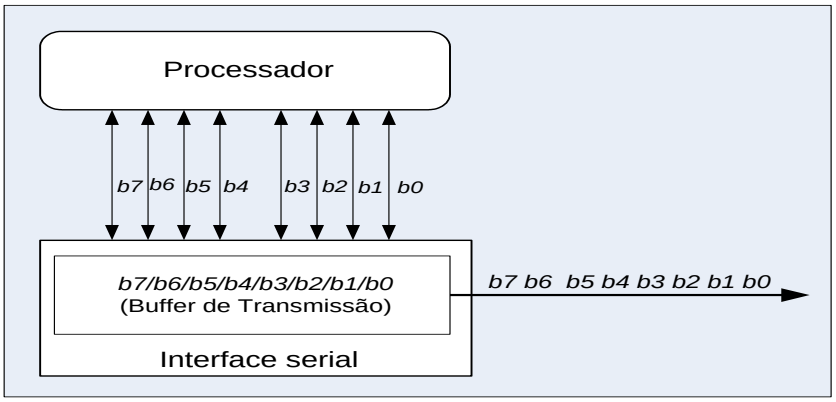

Figura 1. Comunicação paralela entre a CPU e a serial, e a transmissão bit a bit (serial) na saída da interface serial.

Na prática a interface serial é capaz de transmitir e receber dados simultaneamente, o que é chamado de comunicação “full-duplex”. Isto é possível, pois ela possui um buffer específico para transmissão e outro específico para recepção. A interface serial do PC utiliza um padrão chamado de RS232, O computador pode ter mais de uma interface serial RS232, assim, as seriais são referenciadas como “COM1”, “COM2”, “COM3”,...., até “COM16”. Na prática, atualmente, a USB pode conectar um hardware específico e funcionar como serial

A figura abaixo mostra a ideia de conexão de duas interfaces serias. O pino RX faz a recepção serial dos dados (bit a bit). O pino TX faz a transmissão serial dos bits. O pino GND é o chamado “terra” e a sua função é exclusiva no âmbito elétrico. Este padrão serial pode operar com apenas três fios, o que facilita a montagem dos cabos.

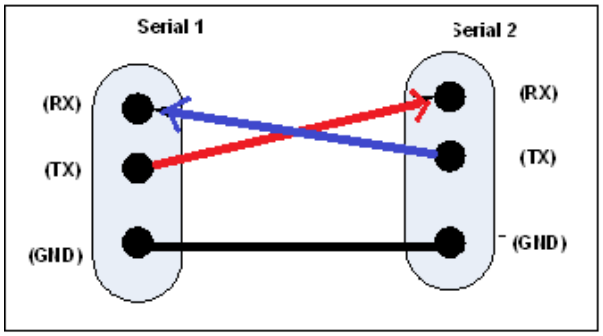

Figura 2. Conexão serial TX/RX

2. BPS

O termo BPS significa *Bits por Segundo* e está associado à ideia de "velocidade". Lembre-se que a velocidade do sinal elétrico em um cabo metálico é fixada na ordem de 0,7C, onde C é a velocidade da luz no vácuo. Portanto é mais correto associar o conceito de BPS a "densidade", ou volume de informações por segundo.

A duração física (T) de cada bit é mostrada na figura 5. Este valor é medido em segundos e pode ser calculada em função da BPS, como:

T = 1 / BPS

Quanto maior for a BPS, menor será T e, portanto, o sistema de comunicação pode se tornar mais susceptível a falhas.

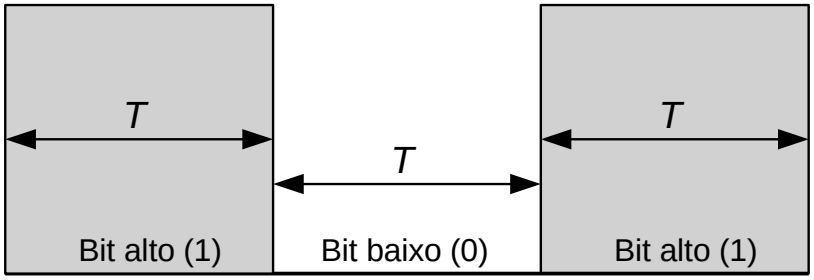

Figura 3. Exemplo da duração de cada bit, calculado em função da BPS.

3. **Comunicação Síncrona e Assíncrona**

Tanto a forma de comunicação síncrona quanto a síncrona são seriais. Na comunicação *Síncrona* é necessário que haja um elemento adicional de hardware que gere um "clock" (sinal) de referência, o qual determina a cadência da comunicação entre os computadores. A sua vantagem é a possibilidade de se tingir maiores taxas de transmissão, porém a sua desvantagem é a de exigir mais hardware, ou um hardware mais específico.

Na comunicação *Assíncrona* não é necessário nenhum hardware de referência para coordenar a sincronia de transmissão. Neste caso, todo byte é acrescido de dois outros bytes adicionais, chamados de "start bit" e "stop bit".

O start-bit é sempre 0 e o stop-bit é sempre 1. Quando não há comunicação, a linha de comunicação fica sempre em nível lógico alto (1), assim, se o receptor estiver ocioso e de repente ele receber um nível lógico 0, este será considerado um "aviso" de que imediatamente após será recebido um byte. Após o byte o receptor recebe obrigatoriamente um stop-bit, que um bit em nível lógico 1, o que servirá como referencial para saber quando começara uma nova recepção.

4. Bit de Paridade

É uma técnica de verificação de erros na transmissão, onde um bit adicional (9° bit) é adicionado aos 8 bits do byte a ser transmitido.

Se a paridade escolhida for par, então o número de '1' nos 9 bits deve ser par

Se a paridade escolhida for ímpar, então o número de '1' nos 9 bits deve ser par

Por exemplo, suponha que a paridade escolhida seja ímpar, então nono bit poderá ser um ou zero, para tornar o número de dígitos '1' par ou ímpar, conforme exemplo abaixo.

Exemplo do 9° bit, com paridade ímpar

Byte a ser transmitido	**(9° bit) Bit de paridade a ser acrescentado no final do byte a ser transmitido**
1 1 0 0 1 0 0 0	**0**
1 1 0 0 1 0 0 1	**1**

Registradores utilizados na programação da comunicação serial do PIC.

1) Registradores Principais:

TXSTA	Programa Transmissão serial
RCSTA	Programa Recepção Serial
SPBRG	Definição da Baud Rate

2) Buffers

TXREG	Buffer de transmissão
RCREG	Buffer de recepção (2 bytes)

3) Registradores Auxiliares na programação:

PIE1	Peripheral Interrupts Enable
PIR1	Individual flag bits for the peripheral interrupt

1) TXSTA: Transmit Status and Control Register

TXSTA

7	6	5	4	3	2	1	0
CSRC	TX9	TXEN	SYNC	X	BRGH	TRMT	TX9D

CSRC: Fonte do clock
Modo assíncrono: não importa.

TX9: habilita transmissão de 9º bits
1 = Seleciona a transmissão de 9 bits
0 = Seleciona a transmissão de 8 bits

TXEN: Habilita transmissão
1 = transmissão ativada
0 = Transmitir desativado

SYNC: Seleção do Modo de Operação
1 = modo síncrono
0 = modo assíncrono

BRGH: Seleção de taxa para cálculo da baud rate (a escolha vai depender do cálculo).
Só é usado no modo Assíncrono
1 = alta velocidade
0 = velocidade baixa

TRMT: Status do registrador de transmissão.
1 = TSR vazio
0 = TSR cheio

TX9D: 9º bit de dados de transmissão.
Pode ser bit de paridade

2) RCSTA: Receive Status and Control Register

RCSTA

7	6	5	4	3	2	1	0
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

SPEN: Habilitação da porta serial
1 = Porta serial ativada
(Os pinos RB1/RX e RB2/TX para serial. TRISB <2:1> deve ser setado).
0 = porta serial desativada

RX9: habilitação de recebimento de 9 bits
1 = Seleciona a recepção de 9 bits
0 = Seleciona a recepção de 8 bits

SREN: Bit de habilitação de recebimento único
Modo assíncrono: não importa.

CREN: Habilita recebimento contínuo
Modo assíncrono: 1 = Permite recebimento contínuo
0 = desativa o recebimento contínuo

ADDEN: Habilitação de detecção de endereço
Modo síncrono: Não é usado
Modo assíncrono: 1 = Habilita a detecção de endereços
0 = Desativa a detecção de endereço

FERR: bit de erro de enquadramento. Indica se se houve problema na detecção do Stop-Bit
1 = Erro de enquadramento (Framig)
0 = sem erro de enquadramento

OERR: bit de erro de Overrum (O buffer já está cheio e chegam novos bytes)
1 = Erro de overrun
0 = sem erro

RX9D: 9º bit de dados
9º bit de dados recebidos (pode ser o valor do bit de paridade recebido)

3) SPBRG: USART Baud Rate Generator

O SPBRG determina a taxa de transmissão e depende do bit: TXSTA,BRGH

Se BRGH=0:

$$SPBRGH = \left(\frac{Fosc}{64*bps}\right) - 1 \qquad (1)$$

Se BRGH=1:

$$SPBRGH = \left(\frac{Fosc}{16*bps}\right) - 1 \qquad (2)$$

Valores usuais de bps: 1200, 2400, 4800, 9600, 19200, 38400, 57600 e 115200 bps

4) Buffers de comunicação: TXREG e RCREG

TXREG: Funciona como um Buffer de Transmissão
RCREG: Funciona como um Buffer de Recepção

6) PIE1: Peripheral Interrupts Enable

PIE1

7	6	5	4	3	2	1	0
		RCIE	**TXIE**				

PIE,RCIE = 0 Desabilita interrupção de Recepção
PIE,RCIE = 1 Habilita interrupção de Recepção

PIE,TXIE = 0 Desabilita interrupção de Transmissão
PIE,TXIE = 1 Habilita interrupção de Transmissão

Após reset o PIE1 é totalmente zerado automaticamente.

7) PIR1: individual flag bits for the peripheral interrupt

PIR1:

7	6	5	4	3	2	1	0
		RCIF	**TXIF**				

PIE,RCIF = 0 Não houve interrupção de Recepção
PIE,RCIE = 1 Houve interrupção de Recepção

PIE,TXIF = 0 Não houve interrupção de Transmissão
PIE,TXIF = 1 Houve interrupção de Transmissão

ALGORITMOS BÁSICOS:

1) Programação:

```
TRISB  =  b 'xxxx   x11x'
TXSTA = b '0010   x110'
RCSTA = b'1001  0000 '
SPBRS  = valor-calculado
```

2) Lógica de recepção de um byte.

```
LOOP:  BTFSS  PIR1,RCIF      ; Chegou algum byte pela serial ?
       GOTO   LOOP           ; Não
       MOVF   RCREG,W        ; Sim, então tiro o byte do buffer de recepção
```

O comando RCREG,W zera automaticamente o bit PIR1,RCIF

3) Lógica de transmissão de um byte.

```
WAIT:  BTFSS  TXSTA,TRMT  ; O buffer de transmissão está vazio ?
       GOTO   WAIT        ; Não
       MOWF  TXREG        ; Sim, então coloco dado no buffer para ser enviado
```