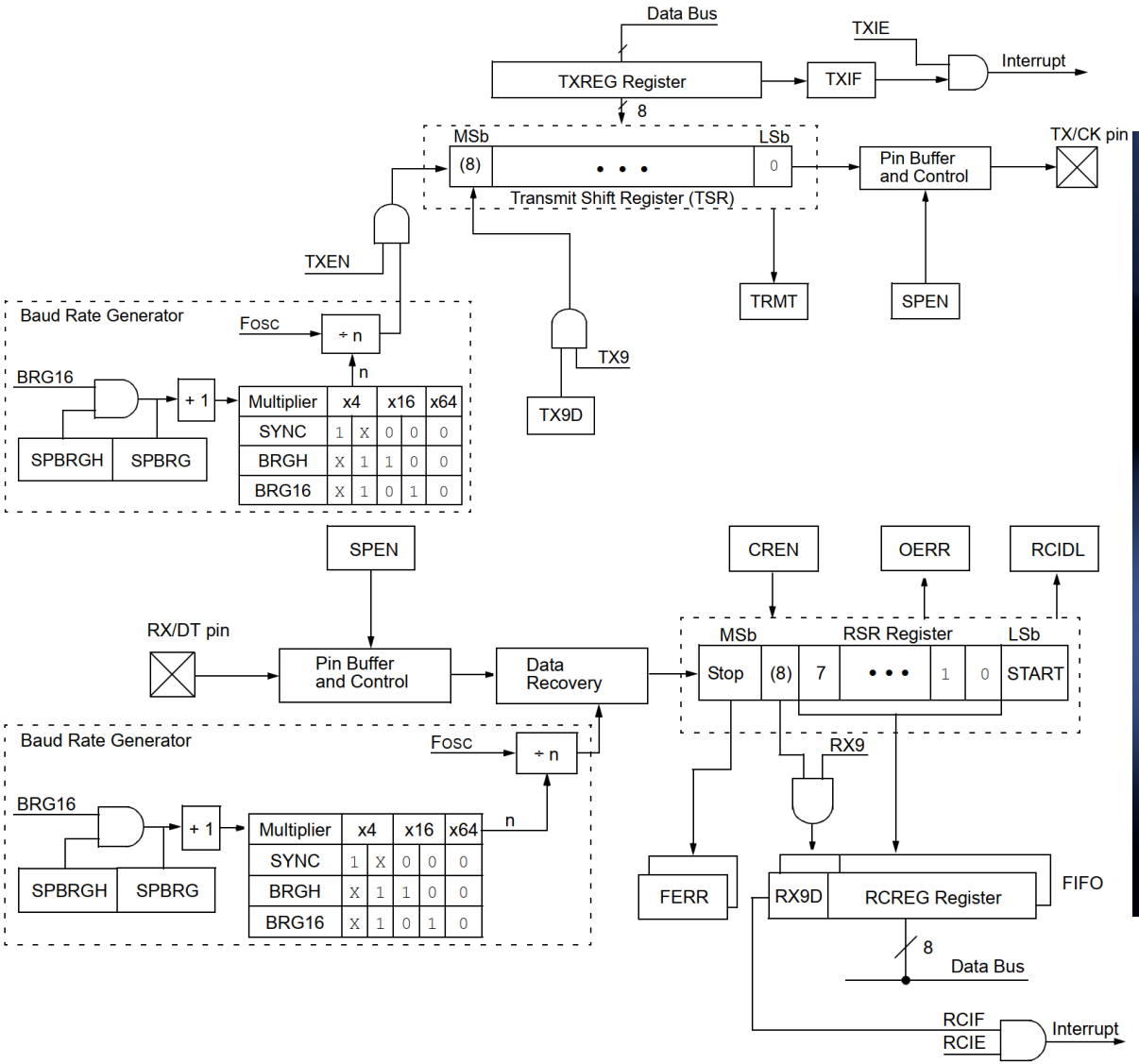


PMR 3406 – Microprocessadores em Automação e Robótica

Aula 06 – Comum. Serial

Prof. Dr. Jun Okamoto Jr.

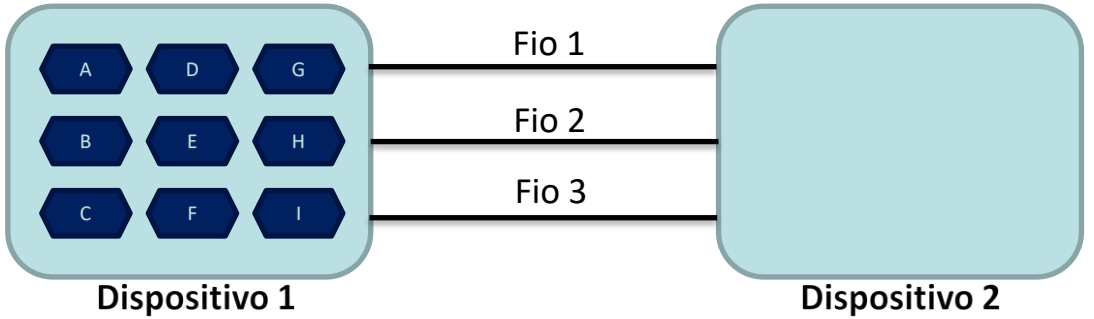
Prof. Dr. Rafael Traldi Moura

Comunicação

- É o processo de transmitir dados entre dois dispositivos;
- A comunicação pode ser do tipo paralela ou em série;
- Diversas características são analisadas em um sistema de transmissão:
  - Custo dos materiais;
  - Velocidade da transmissão;
  - Ruídos do processo;
  - Distância entre os dispositivos;

Tipos de comunicação

Comunicação paralela



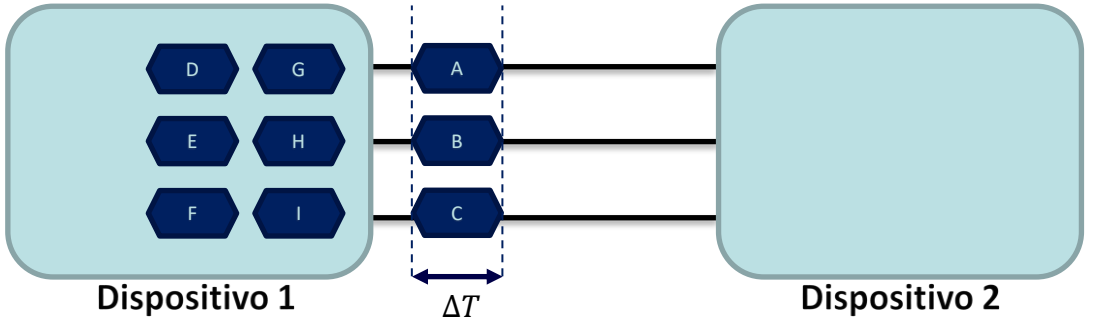
- n bits são transmitidos por n fios de dados simultaneamente.

Comunicação serial

- 1 bit é transmitido de cada vez pelo único fio de dado existente.

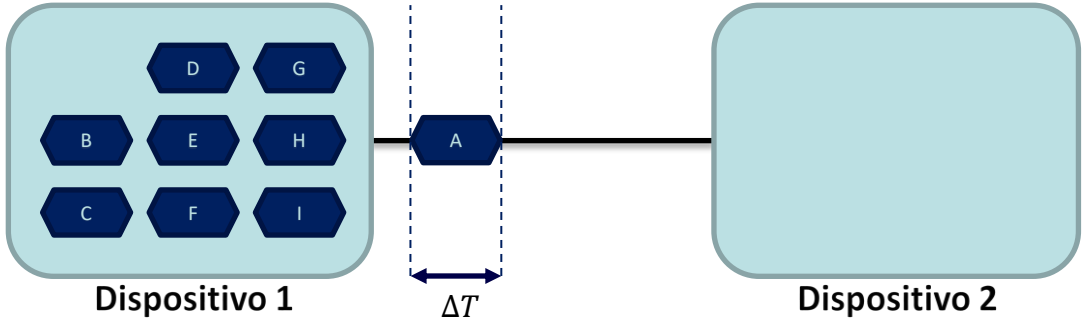
Tipos de comunicação

Comunicação paralela



Taxa de dados $\frac{n\ bits}{\Delta T}$

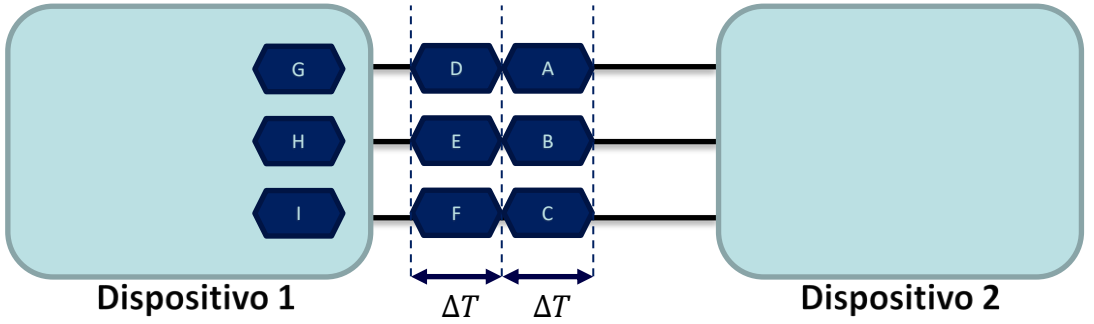
Comunicação serial



Taxa de dados $\frac{1\ bits}{\Delta T}$

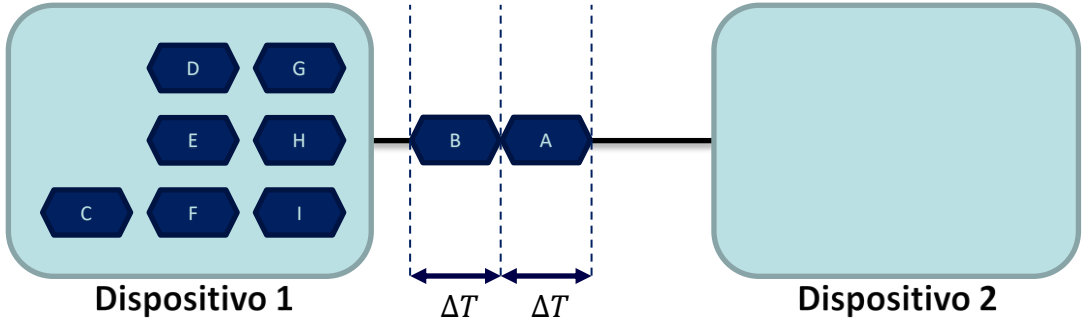
Tipos de comunicação

Comunicação paralela



Taxa de dados $\frac{n\ bits}{\Delta T}$

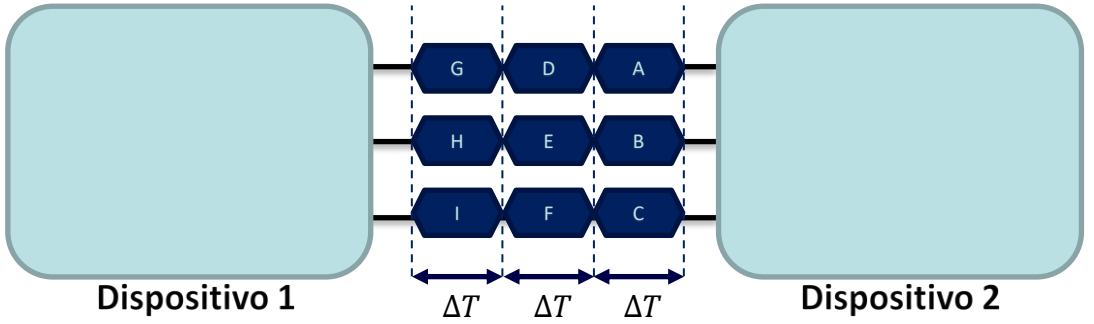
Comunicação serial



Taxa de dados $\frac{1\ bits}{\Delta T}$

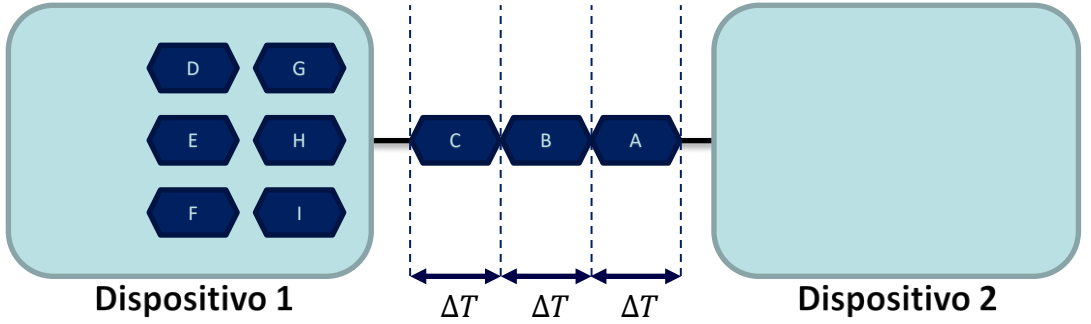
Tipos de comunicação

Comunicação paralela



Taxa de dados $\frac{n\ bits}{\Delta T}$

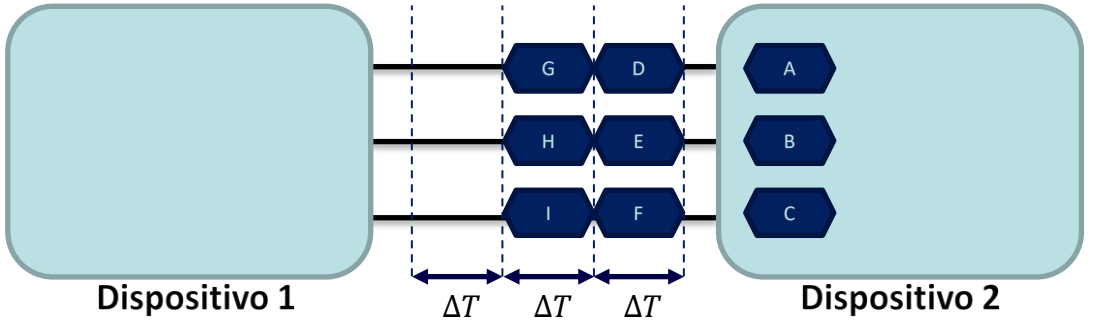
Comunicação serial



Taxa de dados $\frac{1\ bits}{\Delta T}$

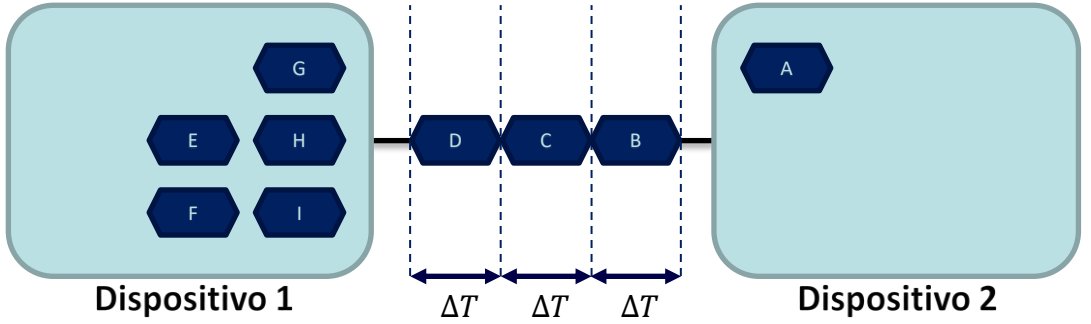
Tipos de comunicação

Comunicação paralela



Taxa de dados $\frac{n\ bits}{\Delta T}$

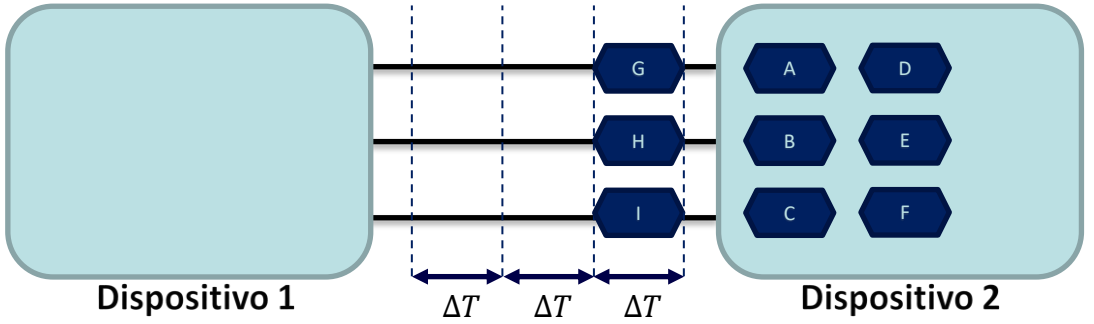
Comunicação serial



Taxa de dados $\frac{1\ bits}{\Delta T}$

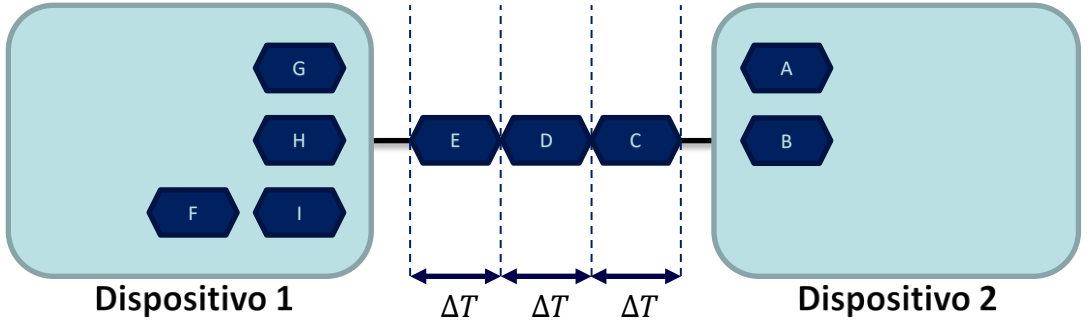
Tipos de comunicação

Comunicação paralela



Taxa de dados $\frac{n\ bits}{\Delta T}$

Comunicação serial



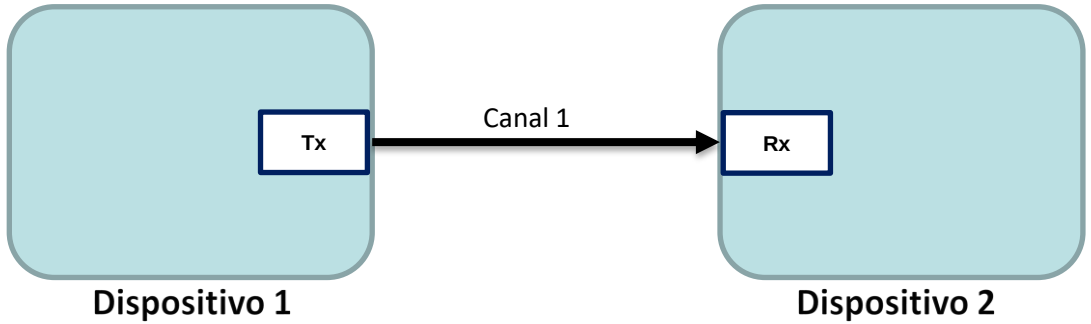
Taxa de dados $\frac{1\ bits}{\Delta T}$

Comunicação Serial

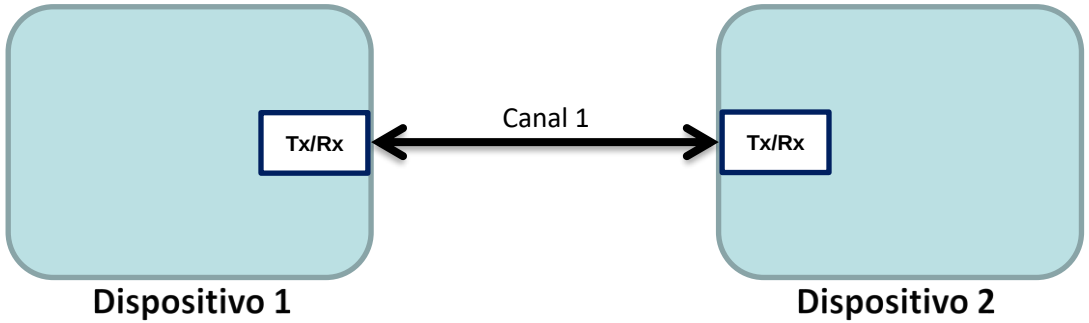
- É n vezes mais lenta que a comunicação paralela;
- É utilizada quando não é possível (custo, ruído) ou não precisa (quando não precisa de velocidade alta de transmissão) aplicar a paralela;
- Cenários que se usa comunicação serial:
  - Transmissão em longas distâncias;
  - Menor custo de transmissão;
  - Quando a comunicação precisa ter menos ruído;
  - Se o dispositivo receptor é mais lento que o transmissor;
- Pode ser do tipo: Simplex, Half-duplex, Full-duplex
- Exemplo de uso:
  - Cabo **U**niversal **S**erial **B**us USB

Tipos de transmissão

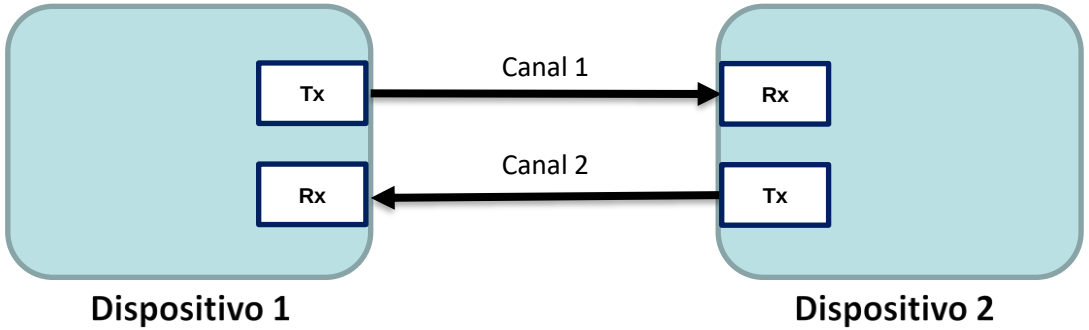
Simplex
Transmissão de dados em uma única direção



Half-duplex
Transmissão de dados nas duas direções, uma de cada vez

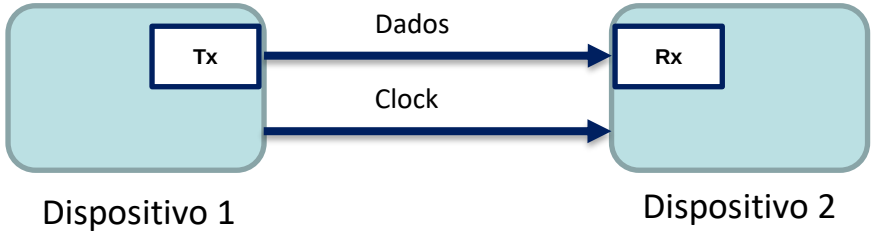


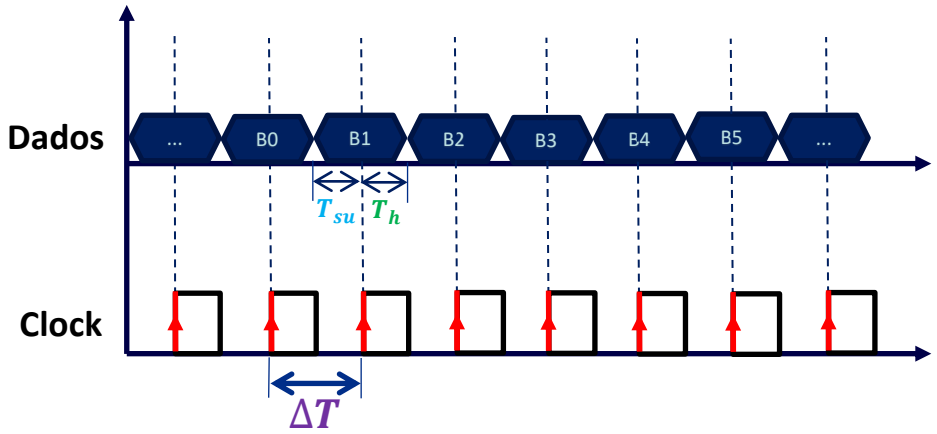
Full-duplex
Transmissão de dados nas duas direções, simultaneamente



Tipos de transmissão

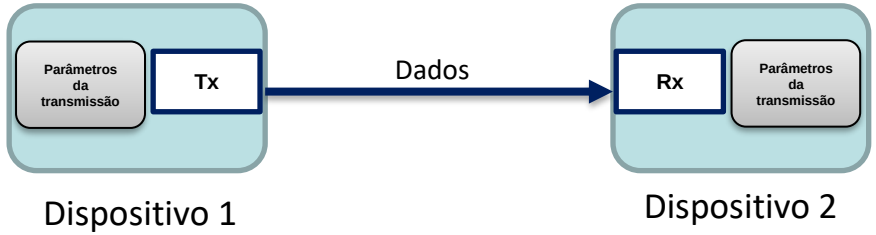
Síncrona

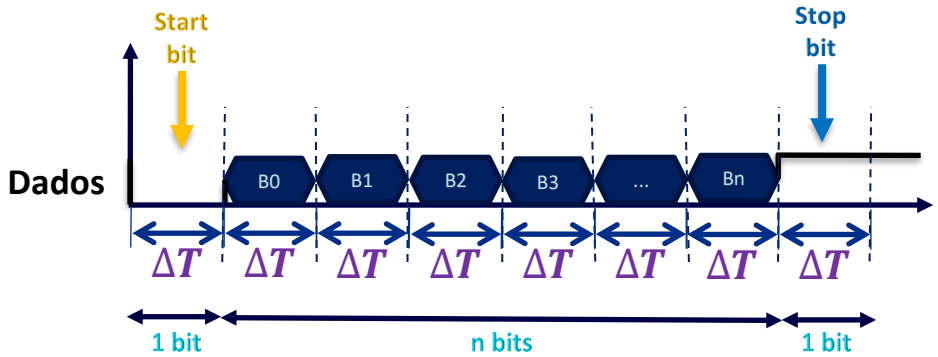




- O **Tempo de setup** (T_{su}) e **Tempo de hold** (T_h) do receptor precisam ser respeitados;
- A **borda de subida do clock** indica que o bit está estável para transmissão;
- ΔT corresponde a **taxa de transmissão** em bits/segundo (bps), e é dada pela frequência do clock;

Assíncrona





- Os **parâmetros da transmissão (n bits, frequência, etc)** são definidos previamente;
- Um **start bit** (low) indica o início da transmissão;
- Um **stop bit** (high) indica o fim da transmissão;
- ΔT corresponde a **taxa de transmissão** em bits/segundo (bps);
- São necessários **n+2 bits** para transmitir os dados;

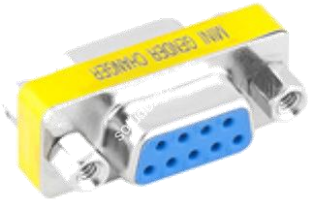
Padrão RS232 (Recomended Standart 232)

- O padrão define características elétricas, mecânicas e lógicas;
- Possui como padrão **elétrico** ter níveis de tensão de 5 a $20V$, sendo que:
  - $\boldsymbol{V_{HIGH}} < 0$, com intervalos de $[-5; -20]V$
  - $\boldsymbol{V_{LOW}} > 0$, com intervalos de $[+5; +20]V$

  Este intervalo é utilizado para remediar perdas de tensão, por exemplo, em transmissões de longa distância, pois como a faixa de tensão é grande, apesar das perdas, lê-se o estado correto; e a justificativa para o deslocamento dos valores iniciais distantes do 0V ($\boldsymbol{V_{LOW}}$ iniciando em +5V e $\boldsymbol{V_{HIGH}}$ iniciando em -5V) é para ter uma clara separação entre os estados HIGH e LOW;
- O padrão **mecânico** consiste no uso do conector DB-25 (ou DB-9);
- Já o padrão **lógico** é a utilização de comunicação serial assíncrona full-duplex;
- Surgiu para realizar transmissões a longas distâncias usando Comunicação DTE-DCE (Data Communication Equipament e Data Terminal Equipament), sempre com um modem no meio;

No PIC do laboratório:
- $\boldsymbol{V_{HIGH}} \rightarrow 5V$;
- $\boldsymbol{V_{LOW}} \rightarrow 0V$;
- Não usa conector DB-25;
- Comunicação serial assíncrona full-duplex;

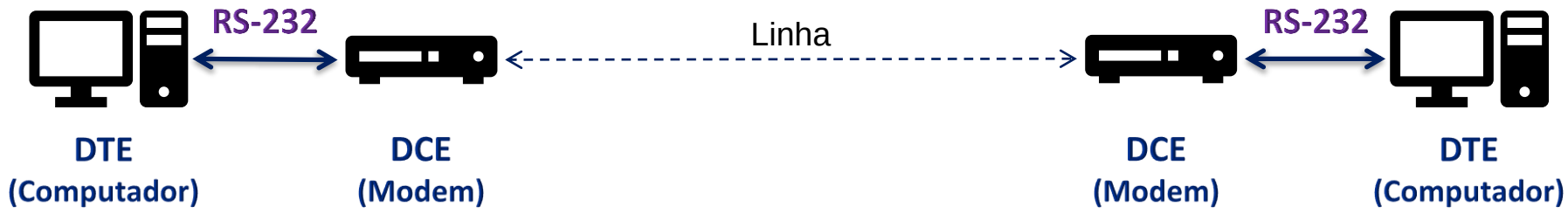
Conector DB-25



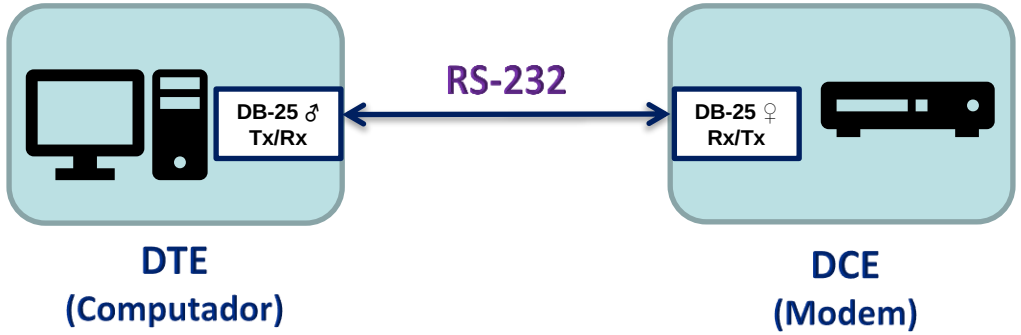
Conector DB-9

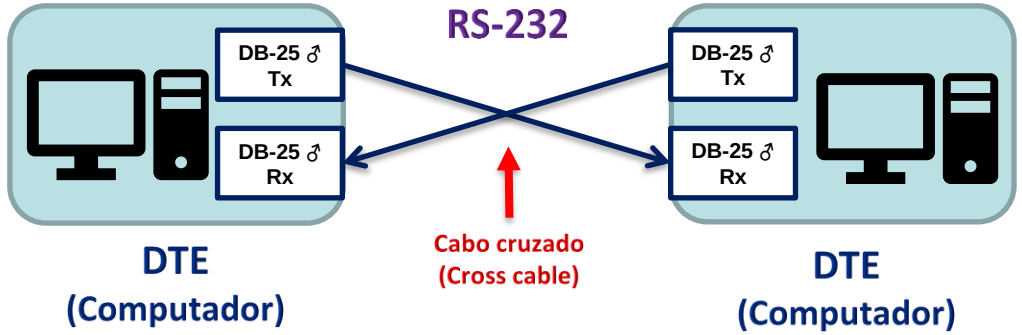
Comunicação DTE-DCE

- DTE – Data Terminal Equipment;
- DCE – Data Communication Equipment;



- Realizando uma análise ampliada da comunicação usando **RS-232**:





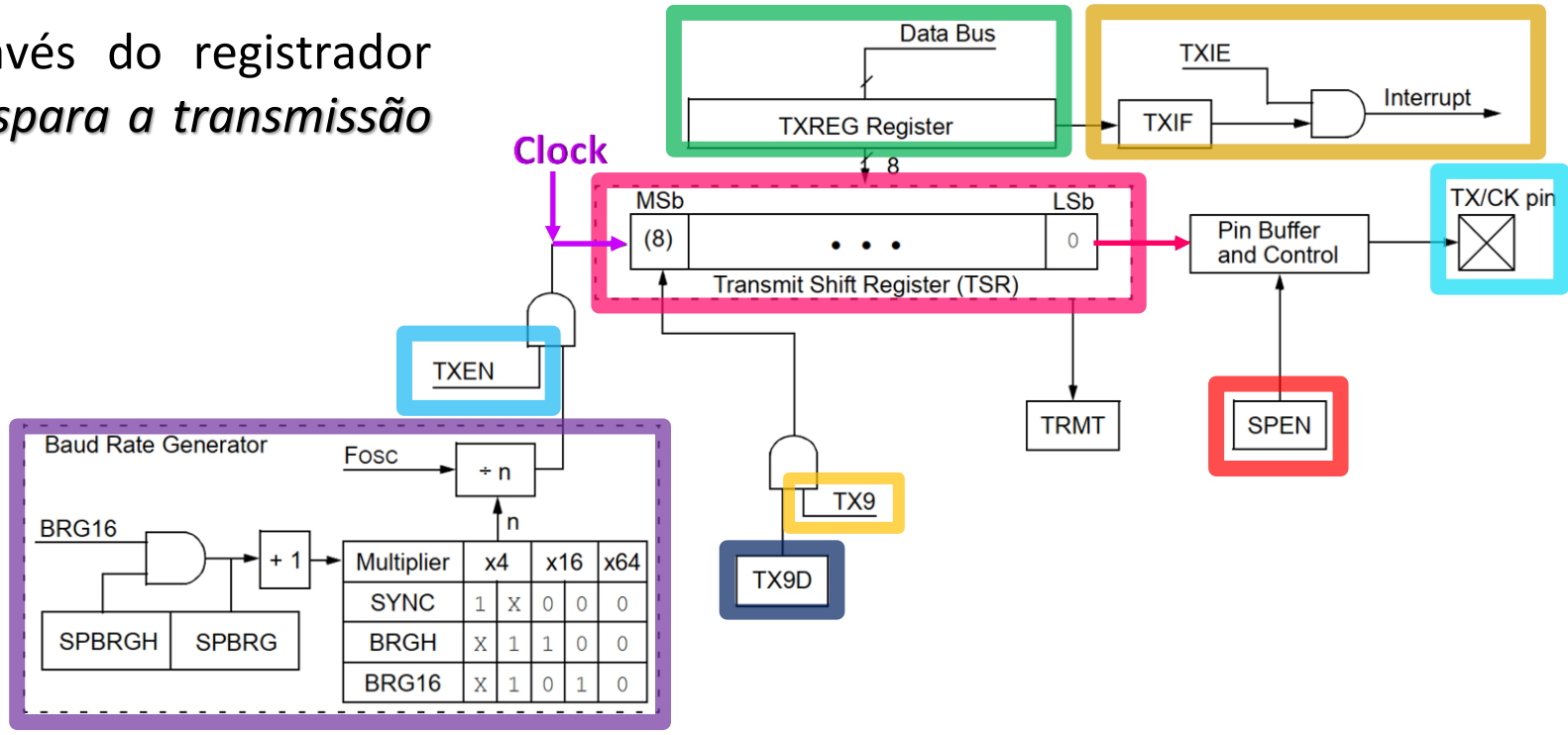
EUSART no PIC16F886

- **E**nhanced **U**niversal **S**ynchronous **A**synchronous **R**eceiver **T**ransmisser é uma Interface de Comunicação Serial (SCI);
- Pode ser configurada como **full-duplex assíncrona** ou **half-duplex síncrona**;
- O modo de full-duplex assíncrono é útil para comunicações com sistemas periféricos como computadores;
- Já o modo de Half-duplex síncrono é mais utilizado para dispositivos periféricos como conversores A/D ou D/A, serial EEPROMs, microcontroladores, etc.
- No módulo de EUSART do PIC16F886 há:
  - Transmissão e recepção assíncrona full-duplex;
  - Buffer de saída com 1 caractere;
  - Buffer de entrada com 2 caracteres;
  - Programação dos dados de transmissão de comprimento 8-9 bits;

Módulo EUSART do PIC - Transmissão

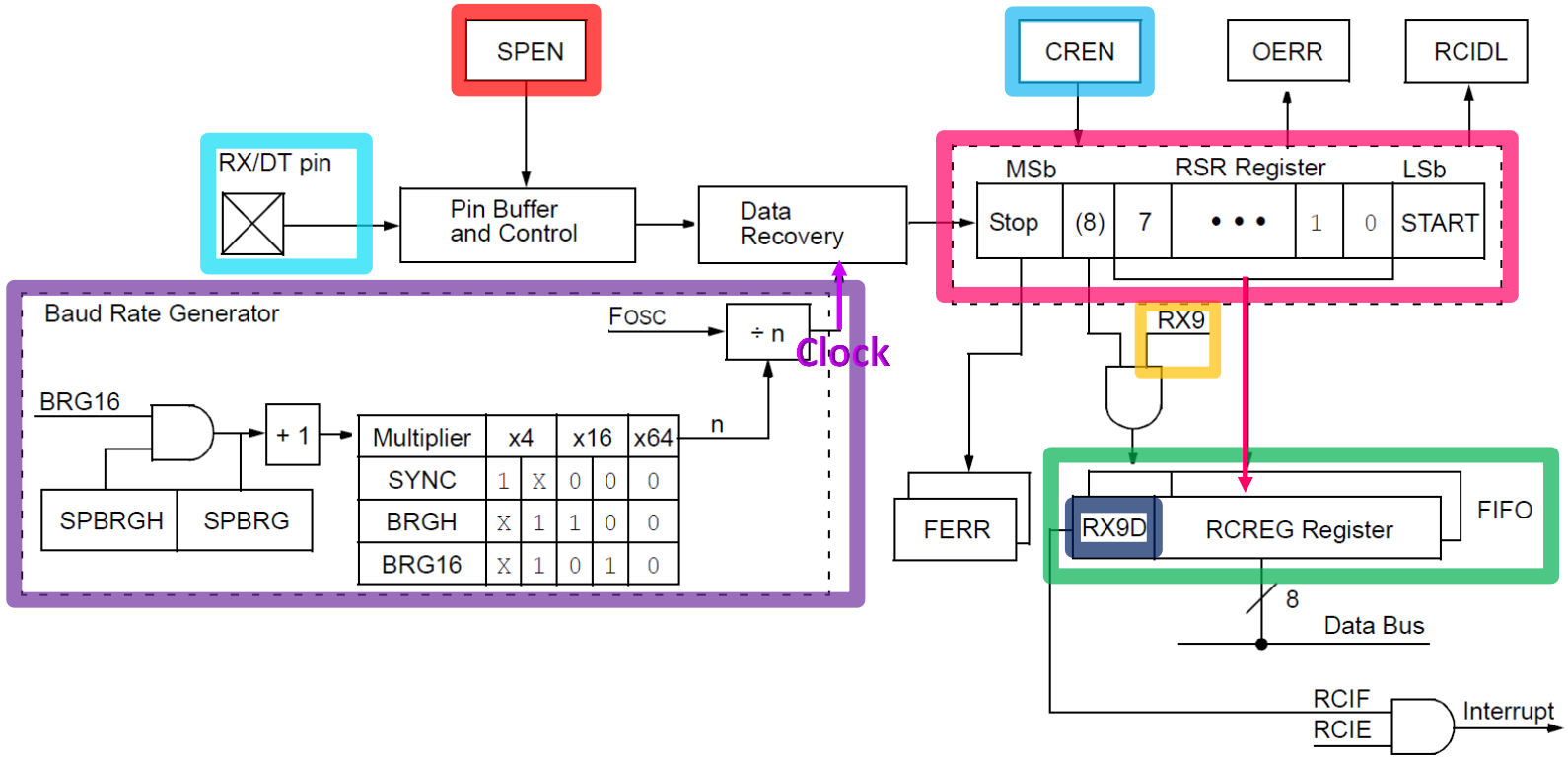
- Os dados transmitidos saem pelo pino **TX/CK**;
- Ao receber um sinal de **Clock**, o **Transmit Shift Register** move o bit menos significativo para o pino de saída **TX/CK**, enquanto realiza o deslocamento dos bits;
- Os dados entram pelo Data Bus através do registrador **TXREG**. *A escrita de dados no **TXREG** dispara a transmissão serial*;
- Para habilitar o bit 9, precisa setar **TX9**. Caso o bit 9 esteja habilitado seu dado é inserido pelo registrador **TX9D**;
- O **Baud Rate Generator** emite o **Clock** que ativa a transmissão de 1 bit, regulando a **frequência de transmissão**;
- Precisa habilitar **SPEN** para usar como serial. Para habilitar o transmissor, é necessário setar **TXEN**;
- Pode ser controlado também usando interrupção. Quando o buffer fica vazio, a flag **TXIF** é setada, gerando a interrupção;

EUSART TRANSMIT BLOCK DIAGRAM
FIGURE 12-1



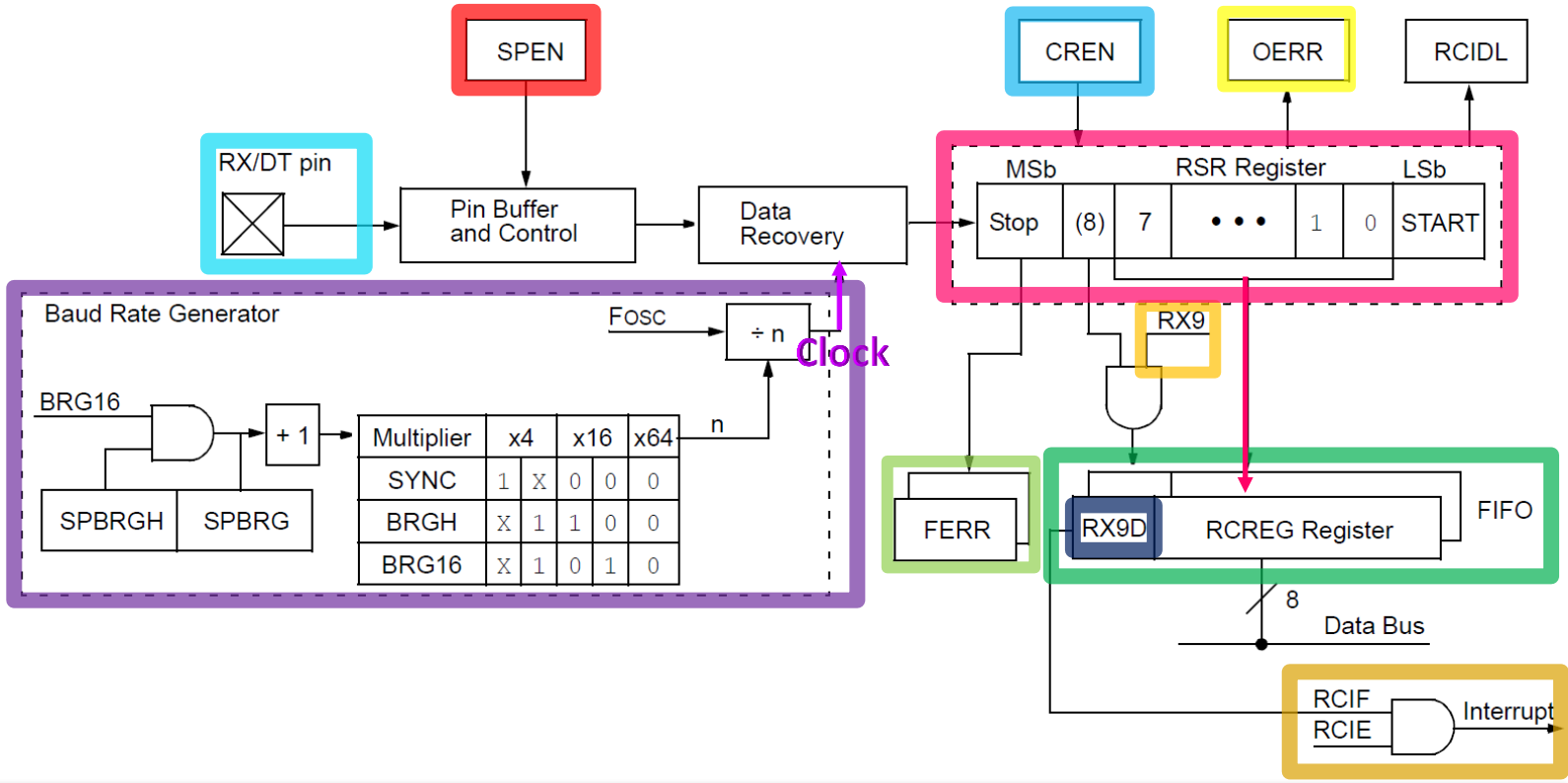
Módulo EUSART do PIC - Recepção

- Os dados chegam no pino **RX/DT**;
- Os bits passam pelo buffer, chegando no **RSR Receive Shift Register**.
- Uma vez recebidos os 8/9 bits, o caractere é passado para o buffer de 2 bytes FIFO (First In, First Out), nos registradores **RCREG** e **RX9D** (se for 9 bits e **RX9** estiver setado). Se eu receber um segundo caractere antes de retirar o primeiro de **RCREG** (e **RX9D**), eu não perco este segundo caractere.
- O **Baud Rate Generator** emite o **Clock** que ativa a recepção de 1 bit, regulando a **frequência de recepção**;
- Precisa habilitar **SPEN** para usar como serial. Para habilitar a recepção, é necessário setar **CREN** (Continuous Receive Enable bit);



Módulo EUSART do PIC - Recepção

- A flag de erro **FERR** (frame error) é setada se não chegar o *stop bit*;
- A flag de erro **OERR** (overrun error) é setada quando chega um caractere antes de esvaziar o buffer de 2 caracteres;
- A flag de interrupção **RCIF** é de apenas leitura e não pode ser setada ou limpa por programação;
- A flag de interrupção **RCIF** é setada quando há um caractere não lido no buffer FIFO, independente do estados dos bits de enable de interrupções;
- Para usar a interrupção na recepção, os seguintes bits devem estar setados:
  - RCIE no registrador PIE1;
  - PEIE no registrador INTCON;
  - GIE no registrador INTCON.



Calculando velocidade de transmissão

REGISTER 12-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

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

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

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

BRG16: 16-bit Baud Rate Generator bit
1 = 16-bit Baud Rate Generator is used
0 = 8-bit Baud Rate Generator is used

REGISTER 12-3: BAUDCTL: BAUD RATE CONTROL REGISTER

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

Calculando velocidade de transmissão

TABLE 12-4: REGISTERS ASSOCIATED WITH THE BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SPBRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0
SPBRGH	BRG15	BRG14	BRG13	BRG12	BRG11	BRG10	BRG9	BRG8

TABLE 12-3: BAUD RATE FORMULAS

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

Calculando velocidade de transmissão

For a device with Fosc of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

$$Desired\ Baud\ Rate = \frac{Fosc}{64([SPBRGH:SPBRG] + 1)}$$

Solving for SPBRGH:SPBRG:

$$X = \frac{\frac{Fosc}{Desired\ Baud\ Rate}}{64} - 1$$

$$= \frac{\frac{16000000}{9600}}{64} - 1$$

$$= [25.042] = 25$$

$$Calculated\ Baud\ Rate = \frac{16000000}{64(25 + 1)}$$

$$= 9615$$

$$Error = \frac{Calc.\ Baud\ Rate - Desired\ Baud\ Rate}{Desired\ Baud\ Rate}$$

$$= \frac{(9615 - 9600)}{9600} = 0.16\%$$

Calculando velocidade de transmissão

Exemplo para Fosc = 20 MHz e Baud rate = 19200

1. Calculando *n* que é (SPBRGH:SPBRG):

- `SYNC = 0; //assíncrona`
- `BRGH = 1; //High Frequency`
- `BRG16 = 0; //usar somente 8 bits`

$$n = \frac{20 \cdot 10^6}{16 \cdot 19200} - 1 = 64{,}14$$

- `SPBRGH = 0; // BR de 19200bps`
- `SPBRG = 64; // BR de 19200bps`

2. Calculando erro:

$$baud\ rate = \frac{20 \cdot 10^6}{16 \cdot (64 + 1)} = 19230{,}77$$

$$erro = \frac{19230{,}77\ - 19200}{19200} = 0{,}16\%$$

TABLE 12-3: BAUD RATE FORMULAS

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

Procedimento para transmissão

INICIALIZAÇÃO

1. Iniciar SPBRGH, SPBRG, BRGH e BRG16 :

```
• SPBRGH = 0; // BR de 19200bps
• SPBRG = 64; // BR de 19200bps
• BRGH = 1; // BR de 19200bps
• BRG16 = 0; // BR de 19200bps
```

2. Assíncrona em SYNC e habilita em SPEN:

```
• SYNC = 0; //assíncrona
• SPEN = 1; //habilita Serial
```

3. 8 bits:

```
• TX9 = 0; //8 bits
```

4. Habilita TX que causa set de TXIF:

```
• TXEN = 1; //habilita tx
```

5. Sem interrupções;

12.1.1.6 Asynchronous Transmission Setup:

1. Initialize the SPBRGH, SPBRG register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 12.3 "EUSART Baud Rate Generator (BRG)"**).
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If 9-bit transmission is desired, set the TX9 control bit. A set ninth data bit will indicate that the 8 Least Significant data bits are an address when the receiver is set for address detection.
4. Enable the transmission by setting the TXEN control bit. This will cause the TXIF interrupt bit to be set.
5. If interrupts are desired, set the TXIE interrupt enable bit of the PIE1 register. An interrupt will occur immediately provided that the GIE and PEIE bits of the INTCON register are also set.
6. If 9-bit transmission is selected, the ninth bit should be loaded into the TX9D data bit.
7. Load 8-bit data into the TXREG register. This will start the transmission.

Procedimento para transmissão

TRANSMISSÃO (exemplo transmitindo `'A'`)

6. 8 bits;
7. Coloca dados a serem transmitidos em TXREG:

- `TXREG = 'A'; //carrega no registrador`

12.1.1.6 Asynchronous Transmission Setup:

1. Initialize the SPBRGH, SPBRG register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 12.3 "EUSART Baud Rate Generator (BRG)"**).
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If 9-bit transmission is desired, set the TX9 control bit. A set ninth data bit will indicate that the 8 Least Significant data bits are an address when the receiver is set for address detection.
4. Enable the transmission by setting the TXEN control bit. This will cause the TXIF interrupt bit to be set.
5. If interrupts are desired, set the TXIE interrupt enable bit of the PIE1 register. An interrupt will occur immediately provided that the GIE and PEIE bits of the INTCON register are also set.
6. If 9-bit transmission is selected, the ninth bit should be loaded into the TX9D data bit.
7. Load 8-bit data into the TXREG register. This will start the transmission.

Procedimento para recepção

INICIALIZAÇÃO

1. Iniciar SPBRGH, SPBRG, BRGH e BRG16 :

```
• SPBRGH = 0; // BR de 19200bps
• SPBRG = 64; // BR de 19200bps
• BRGH = 1; // BR de 19200bps
• BRG16 = 0; // BR de 19200bps
```

2. Assíncrona em SYNC e habilita em SPEN:

```
• SYNC = 0; //assíncrona
• SPEN = 1; //habilita Serial
```

3. Sem interrupções;
4. 8 bits:

```
• RX9 = 0; //8 bits
```

5. Habilita recepção com CREN:

```
• CREN = 1; //habilita rx
```

6. Sem interrupções;

12.1.2.8 Asynchronous Reception Setup:

1. Initialize the SPBRGH, SPBRG register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 12.3 "EUSART Baud Rate Generator (BRG)"**).
2. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
3. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
4. If 9-bit reception is desired, set the RX9 bit.
5. Enable reception by setting the CREN bit.
6. The RCIF interrupt flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
7. Read the RCSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
8. Get the received 8 Least Significant data bits from the receive buffer by reading the RCREG register.
9. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

Procedimento para recepção

RECEPÇÃO

7. 8 bits;
8. Coloca dados recebidos de RCREG em uma variável :

```
• char a = RCREG; //carrega do registrador
```

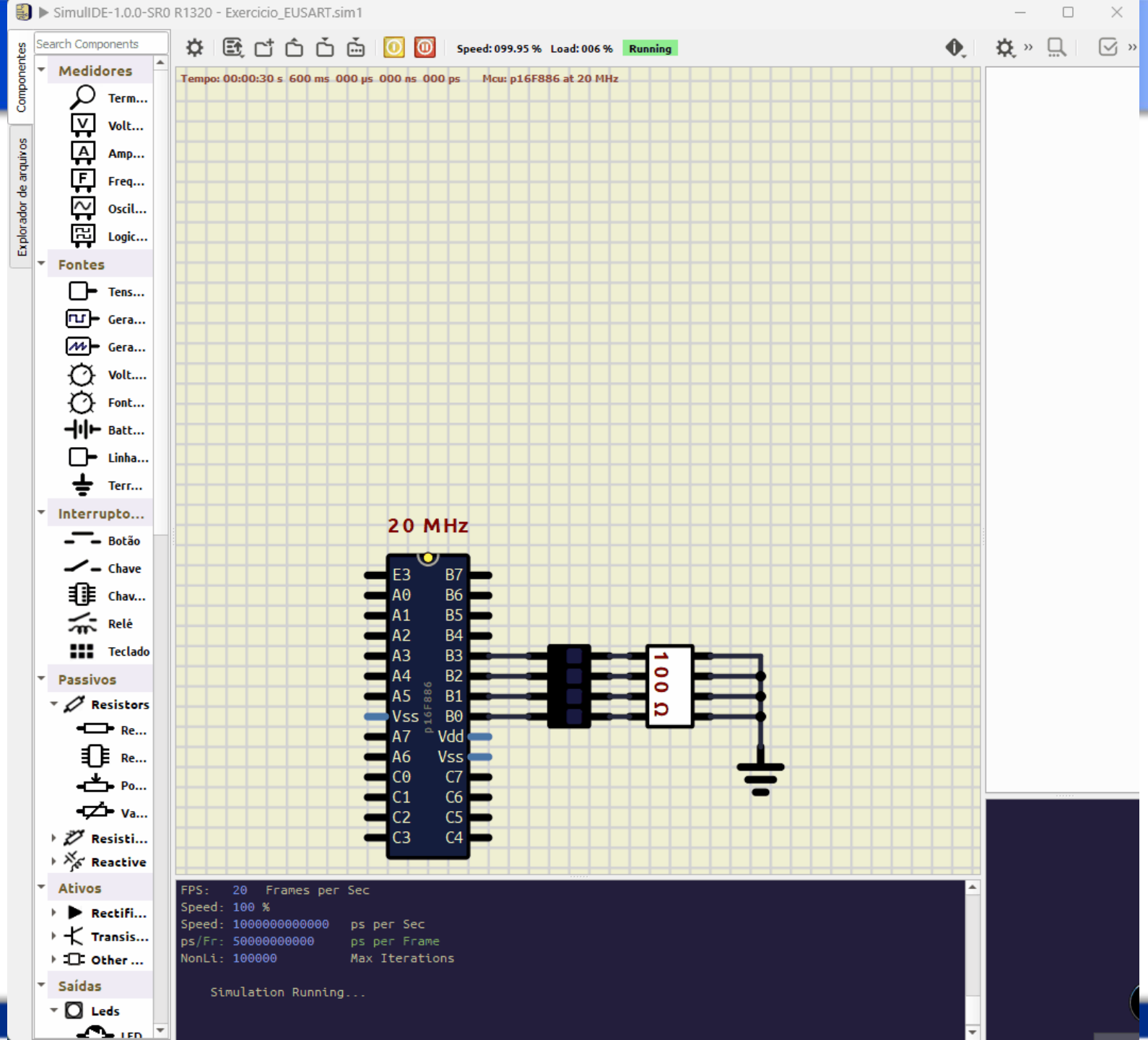
9. Limpa a flag OERR ao resetar CREN :

```
• if(OERR){
      faz algo;
      mais algo;
      CREN = 0; //limpa flag OERR
  }
  CREN = 1; //habilita recepção
```

12.1.2.8 Asynchronous Reception Setup:

1. Initialize the SPBRGH, SPBRG register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 12.3 "EUSART Baud Rate Generator (BRG)"**).
2. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
3. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
4. If 9-bit reception is desired, set the RX9 bit.
5. Enable reception by setting the CREN bit.
6. The RCIF interrupt flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
7. Read the RCSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
8. Get the received 8 Least Significant data bits from the receive buffer by reading the RCREG register.
9. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

Exemplo


SimulIDE-1.0.0-SR0 R1320 - Exercicio_EUSART.sim1
Speed: 099.95 %
Load: 006 %
Running
Tempo: 00:00:30 s 600 ms 000 µs 000 ns 000 ps
Mcu: p16F886 at 20 MHz
20 MHz
E3
A0
A1
A2
A3
A4
A5
Vss
A7
A6
C0
C1
C2
C3
B7
B6
B5
B4
B3
B2
B1
B0
Vdd
Vss
C7
C6
C5
C4
p16F886
100 Ω
FPS: 20 Frames per Sec
Speed: 100 %
Speed: 1000000000000 ps per Sec
ps/Fr: 50000000000 ps per Frame
NonLi: 100000 Max Iterations
Simulation Running...
Componentes
Explorador de arquivos
Search Components
Medidores
Fontes
Interrupto...
Botão
Chave
Relé
Teclado
Passivos
Resistors
Ativos
Rectifi...
Transis...
Other ...
Saídas
Leds

Exemplo - fusíveis

```
/*
 * File:   main.c
 * Author: Rafael Traldi Moura
 *
 * Created on 15 de Maio de 2023, 11:27
 */

// PIC16F886 Configuration Bit Settings
// 'C' source line config statements
// CONFIG1
#pragma config FOSC = EC        // Oscillator Selection bits (EC: I/O function
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled and
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = ON       // RE3/MCLR pin function select bit (RE3/MCLR p
#pragma config CP = OFF         // Code Protection bit (Program memory code pro
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code p
#pragma config BOREN = ON       // Brown Out Reset Selection bits (BOR enabled)
#pragma config IESO = ON        // Internal External Switchover bit (Internal/E
#pragma config FCMEN = ON       // Fail-Safe Clock Monitor Enabled bit (Fail-Sa
#pragma config LVP = OFF        // Low Voltage Programming Enable bit (RB3 pin
// CONFIG2
#pragma config BOR4V = BOR40V   // Brown-out Reset Selection bit (Brown-out Res
#pragma config WRT = OFF        // Flash Program Memory Self Write Enable bits
// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.
```

Exemplo – include e inicialização da serial

```c
#include<xc.h>
#include<stdio.h>
#define _XTAL_FREQ 20000000 // define a frequencia do clock em MHz
// Produz atraso em ms, time < 50.463.240
//__delay_ms(time)
// Produz atraso em us, time < 50.463.240
//__delay_us(time)

// Inicialização do Serial
void serial_init(void) {
    SYNC = 0; //assíncrona
    BRGH = 1; // BR de 19200bps
    BRG16 = 0; // BR de 19200bps
    SPBRGH = 0; // BR de 19200bps
    SPBRG = 64; // BR de 19200bps

    TX9 = 0; //8 bits
    RX9 = 0; //8 bits

    TXIE = 0; //Desabilita Interrupção TX
    RCIE = 0; //Desabilita Interrupção RX

    SPEN = 1; //habilita Serial
    TXEN = 1; //habilita Tx
    CREN = 1; //habilita Rx
}
```

12.1.1.6 Asynchronous Transmission Setup:

1. Initialize the SPBRGH, SPBRG register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 12.3 "EUSART Baud Rate Generator (BRG)"**).
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If 9-bit transmission is desired, set the TX9 control bit. A set ninth data bit will indicate that the 8 Least Significant data bits are an address when the receiver is set for address detection.
4. Enable the transmission by setting the TXEN control bit. This will cause the TXIF interrupt bit to be set.

12.1.2.8 Asynchronous Reception Setup:

1. Initialize the SPBRGH, SPBRG register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 12.3 "EUSART Baud Rate Generator (BRG)"**).
2. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
3. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
4. If 9-bit reception is desired, set the RX9 bit.
5. Enable reception by setting the CREN bit.

Exemplo – recebendo e enviando

```
//Essa função verifica se existe um caractere no buffer de recepção.
//Se existir, retorna o caractere, senão retorna o valor 255.
unsigned char chkchr(void){
    unsigned char recebido;
    if(RCIF){
        recebido = RCREG; //carrega do registrador
    }
    else {
        recebido = 255;
    }
    if(OERR){
        CREN = 0; //limpa flag OERR
        __delay_ms(10);
        CREN = 1; //habilita recepção
    }
    return recebido;
}
//Recebe um caractere ASCII pelo canal serial. Essa função não
//retorna enquanto não for recebido um caractere.
unsigned char getch(void){
    while(!RCIF);
    unsigned char recebido = RCREG;
    if(OERR){
        CREN = 0; //limpa flag OERR
        __delay_ms(10);
        CREN = 1; //habilita recepção
    }
    return recebido;
}
void putch(unsigned char c){
//Envia um caractere ASCII pelo canal serial.
    TXREG = c; //carrega no registrador
}
```

12.1.1.6 Asynchronous Transmission Setup:

6. If 9-bit transmission is selected, the ninth bit should be loaded into the TX9D data bit.
7. Load 8-bit data into the TXREG register. This will start the transmission.

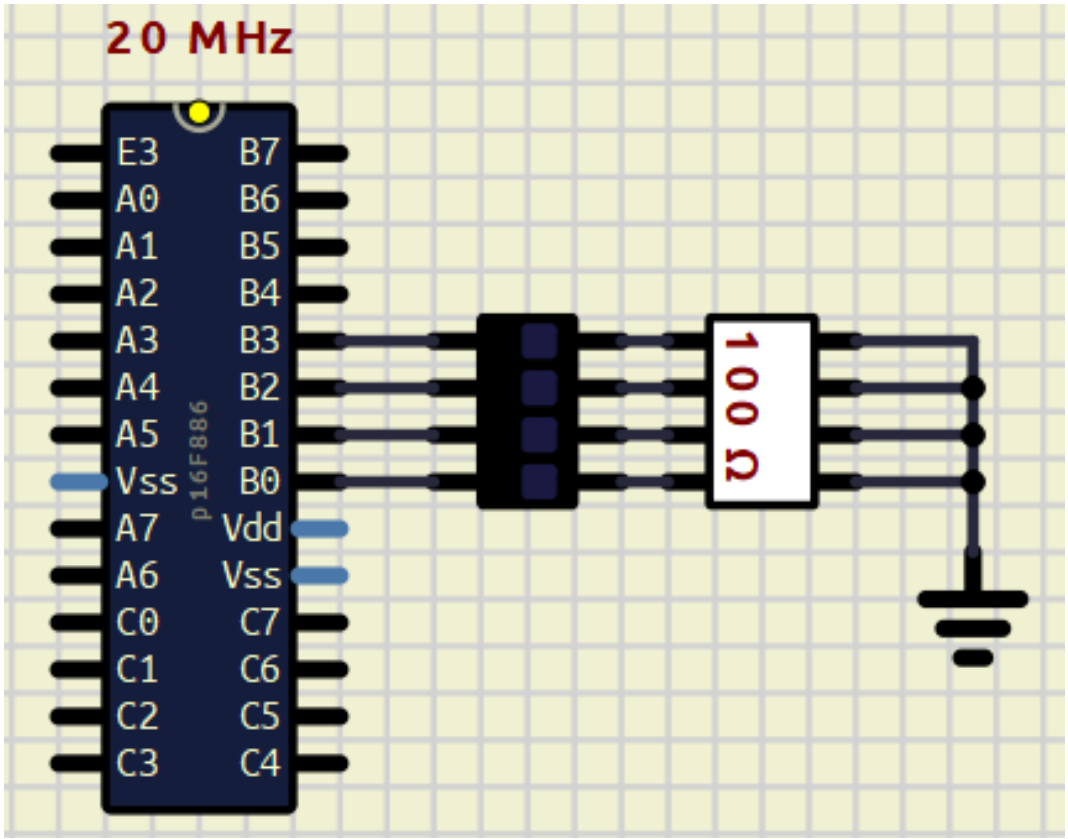
12.1.2.8 Asynchronous Reception Setup:

6. The RCIF interrupt flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
7. Read the RCSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
8. Get the received 8 Least Significant data bits from the receive buffer by reading the RCREG register.
9. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

Exemplo – main

```
void main(void){
    serial_init();
    TRISB &= 0b11110000;
    unsigned char comando;

    while(1){
        comando = chkchr();
        if (comando != 255){
            putch(comando);
            if(comando=='0'){PORTB = 0b0000000;}
            if(comando=='1'){PORTB = 0b0000001;}
            if(comando=='2'){PORTB = 0b0000011;}
            if(comando=='3'){PORTB = 0b0000111;}
            if(comando=='4'){PORTB = 0b0001111;}
        }
    }
    return;
}
```



Exercício

Considerando um **PIC16F886** com clock de **8MHz** (FOSC), determine os valores de **BRG16** e **BRGH**, com **SYNC = 0**, que permitam obter o menor erro para taxa de baud de

- 9600 bps;
- 57600 bps.

Mostre os cálculos para determinar **SPBRG** e **SPBRGH** e os erros para todas as combinações possíveis de BRG16 e BRGH.