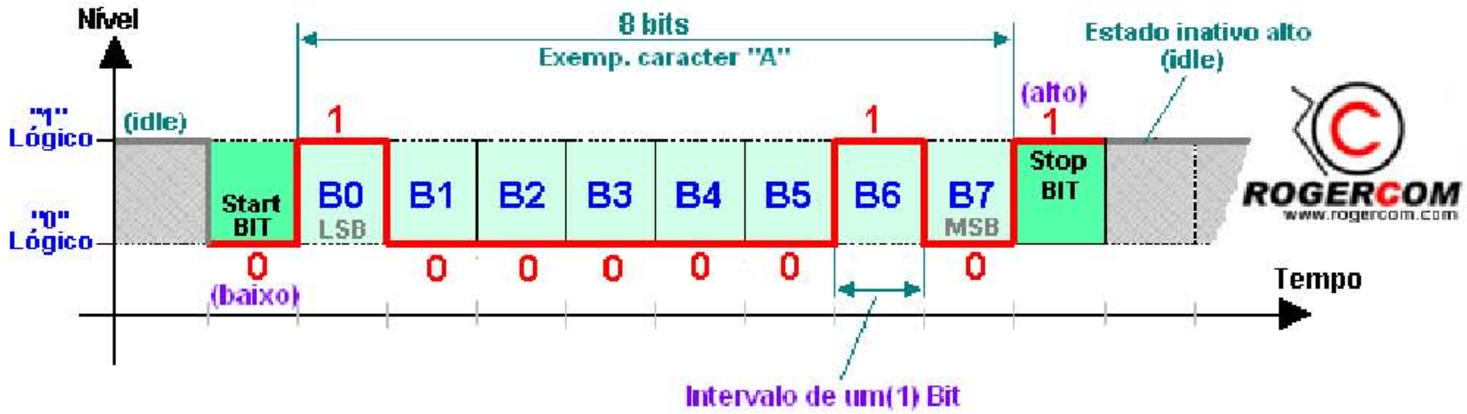
COMUNICACAO SERIAL 18F



Baseado no material da Universidade Tuiuti do Paraná
Engenharia da Computação - 2003

© Prof. Engº esp Luiz Antonio Vargas Pinto
www.vargasp.com

1. Alcance limitado pela distância
2. Sujeito a distorções por ruídos ambientais
3. A Transmissão é diretamente proporcional a potência do sinal e inversamente proporcional ao ruído. Razão para uso de protocolo.

© Prof. Vargas

Canais de comunicação

- Caminho para o trânsito da informação:
  - ☒ Fio elétrico
  - ☒ Ondas de rádio (Wireless)
  - ☒ LASER
  - ☒ Futuramente – comunicação quântica

© Prof. Vargas

Comunicação Serial

Paralela x Serial

- Taxa de transferência paralela é mais rápida
- Muitas linhas $\Rightarrow$ **R$** $\uparrow$

Velocidade de envio (Baud Rate)

- Taxa em Kbps, Mbps (Bit Per Second)
- Exemplo:

Taxa de 9600 bauds $\Rightarrow$ 9600 bits por segundo – Crédito a Èmile Baudot inventor do telégrafo

© Prof. Vargas

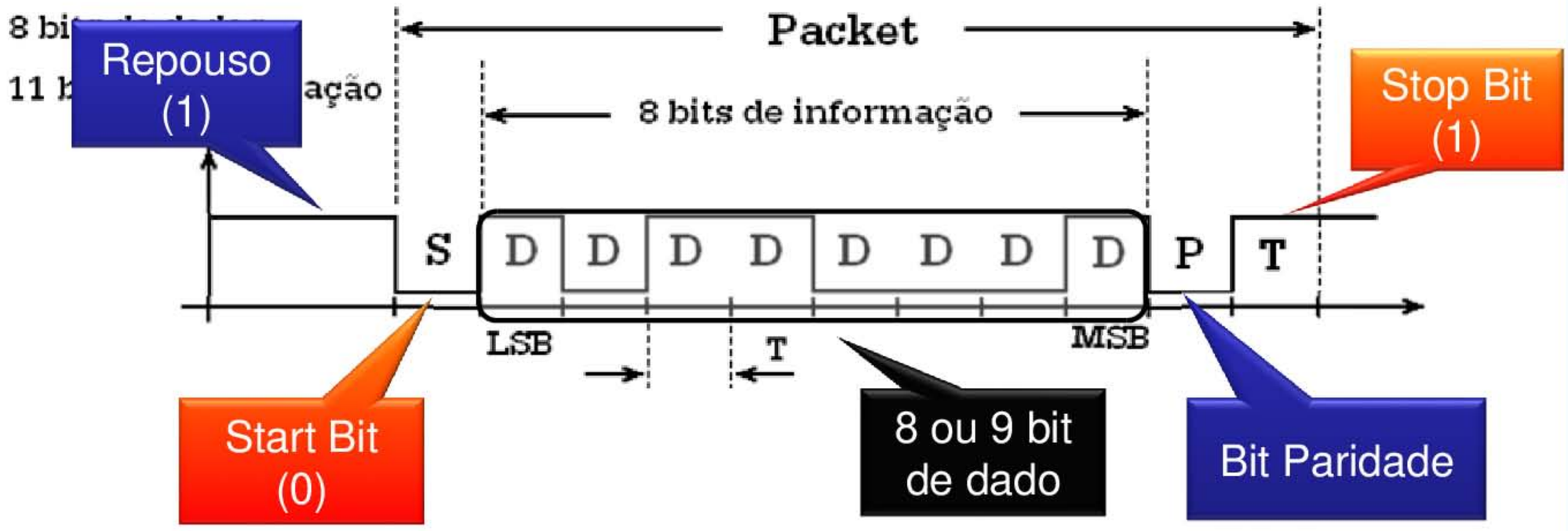
Comunicação Síncrona

- Dados e clock em canais separados
- Sincronizado pelo clock
  - receptor armazena o valor do bit coletado naquele momento.
- Aguarda o próximo clock
  - Opcionalmente clock e dados podem usar o mesmo canal – caso dos MODEM

© Prof. Vargas

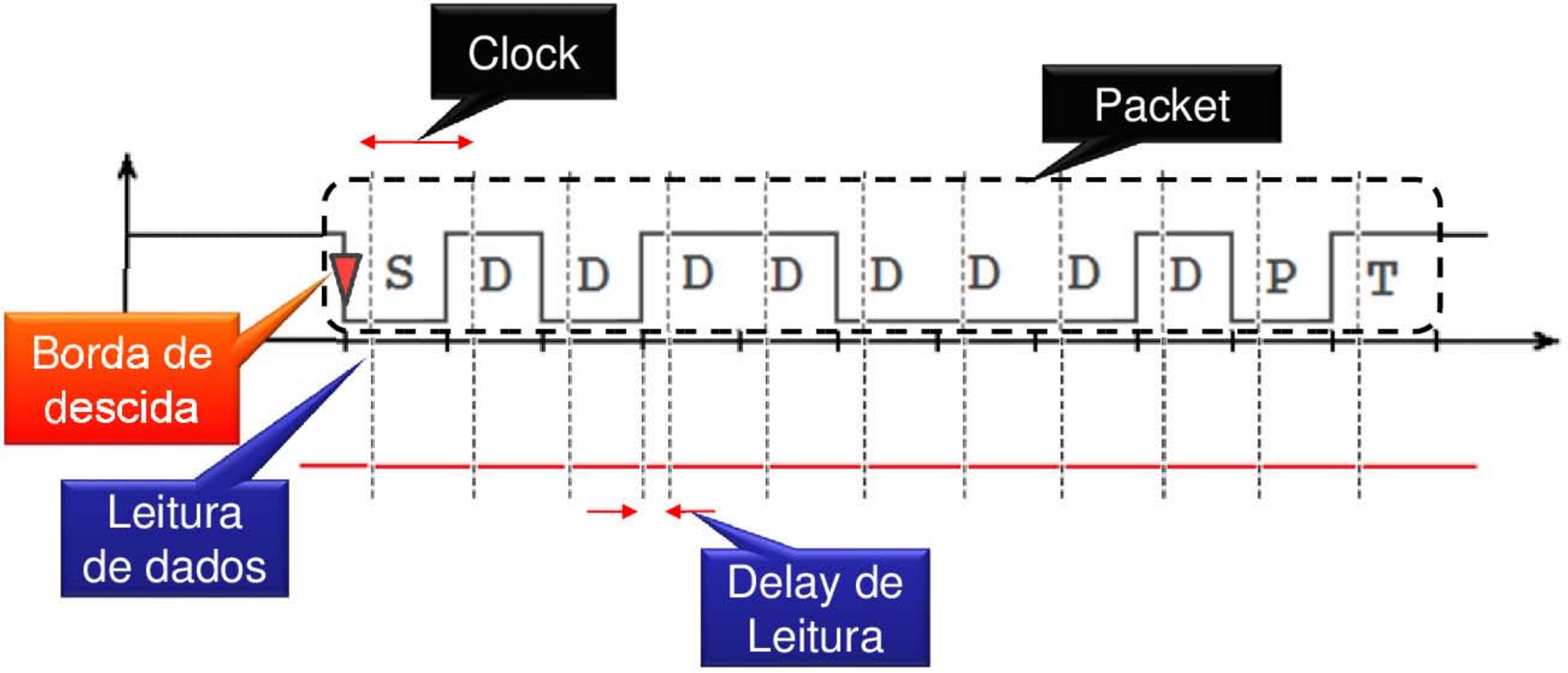
Comunicação Assíncrona

- Dado e clock trafegam em um único canal
- Clock de precisão em ambos Tx e Rx
- Packet mais comum: dados em pacotes de 10 ou 11 bits - 8 constituem a mensagem.



© Prof. Vargas

Comunicação Assíncrona



- O packet é concluído com bits paridade e stop bit
- O comprimento do packet deve ser pequeno para minimizar variações dos osciladores Tx e Rx
- 1 ou 2 bits de erros podem ser corrigidos pelo receptor

© Prof. Vargas

Paridade

- O bit paridade é incluído para detecção de erro
- Em Rx a paridade é recalculada

Dado	Bit de Paridade
10110010	0
10001010	1

© Prof. Vargas

Paridade

- Mas se um número par de bits for trocado, a paridade é mantida e o erro será validado
- Análises estatísticas de erros de comunicação de dados tem mostrado que um erro com bit simples é muito mais provável que erros em múltiplos bits na presença de ruído randômico. Portanto, a paridade é um método confiável para a detecção de erro.

(CANZIAN, E. **Comunicação Serial RS232**: conceitos básicos sobre comunicação serial. http://www.cnz.com.br. CNZ Engenharia e Informática Ltda)

© Prof. Vargas

Checksum

- **CHECKSUM** é uma técnica que insere na comunicação um byte de verificação
  - ☒ Σ (bytes) da mensagem completa.
- Se a Σ (bytes) + checksum ≠ **0** ⇒ Erro
  - ☒ Na ocorrência de erro é improvável (mas não impossível) que corrupção de dados resultem Σ (bytes) + checksum = **0**

© Prof. Vargas

Checksum

```
1011.0001 ..... 0xB1
1000.0110 ..... 0x87
0100.1100 ..... 0x4C
1111.1111 ..... 0xFF
1010.0000 ..... 0xA0
```

Bytes da mensagem

```
0011.0010.0010   Soma Aritmética
   0010.0010     Soma Truncada de 8 bits
 +  1101.1110    Checksum (complemento de 2)
-------------------
   0000.0000     Soma + Checksum = 0
```

A correção de erros em uma transmissão, contudo, abaixa a eficiência do canal, e o resultado é uma queda na transmissão.

© Prof. Vargas

A INTERFACE SERIAL RS-232C

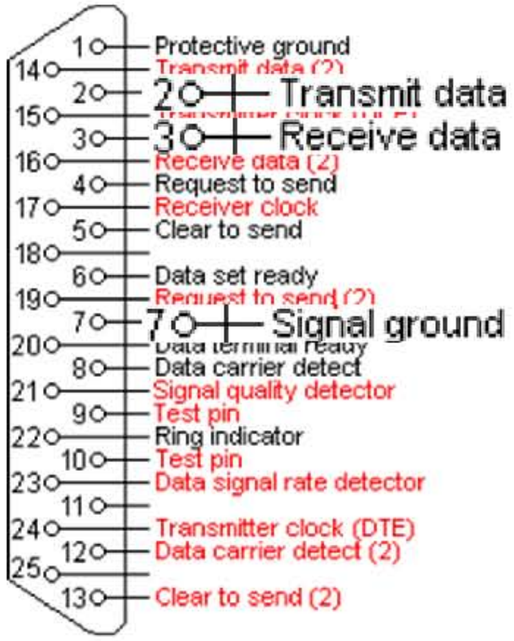
© Prof. Vargas

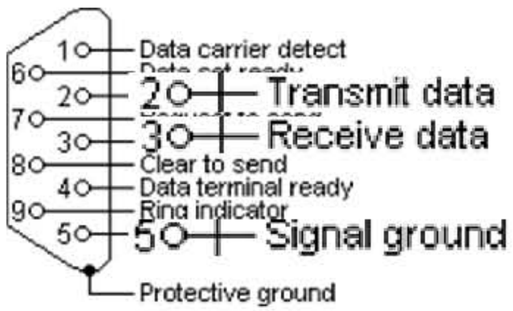
Interface Serial RS-232C

- O termo RS232 (Recommended Standard 232) será utilizado para fazer referência à interface de comunicação.
- O termo EIA232 será utilizado para fazer referência à norma estabelecida pela EIA (Eletronics Industries Association).

© Prof. Vargas

Conectores

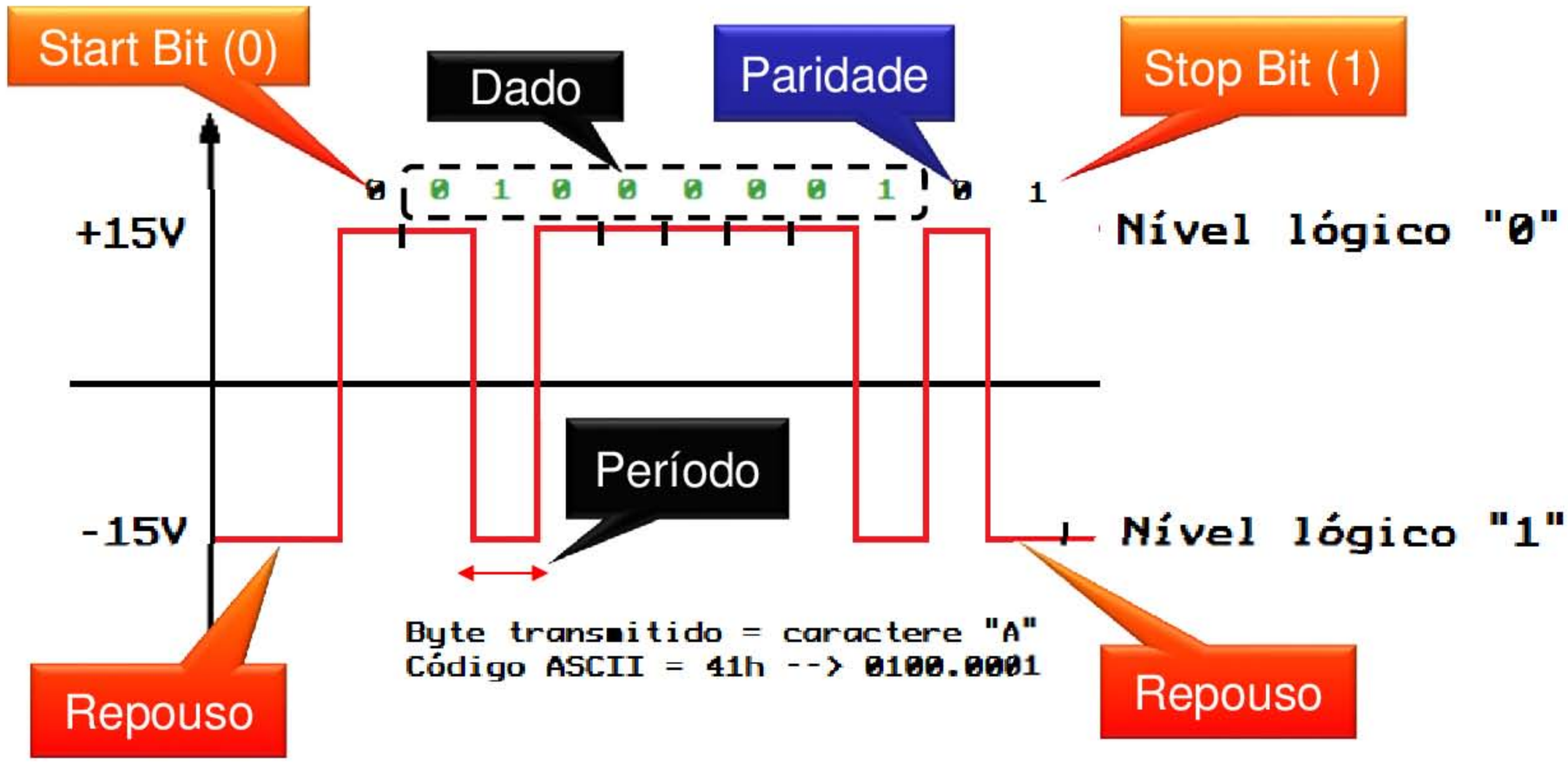




© Prof. Vargas

Níveis de tensão

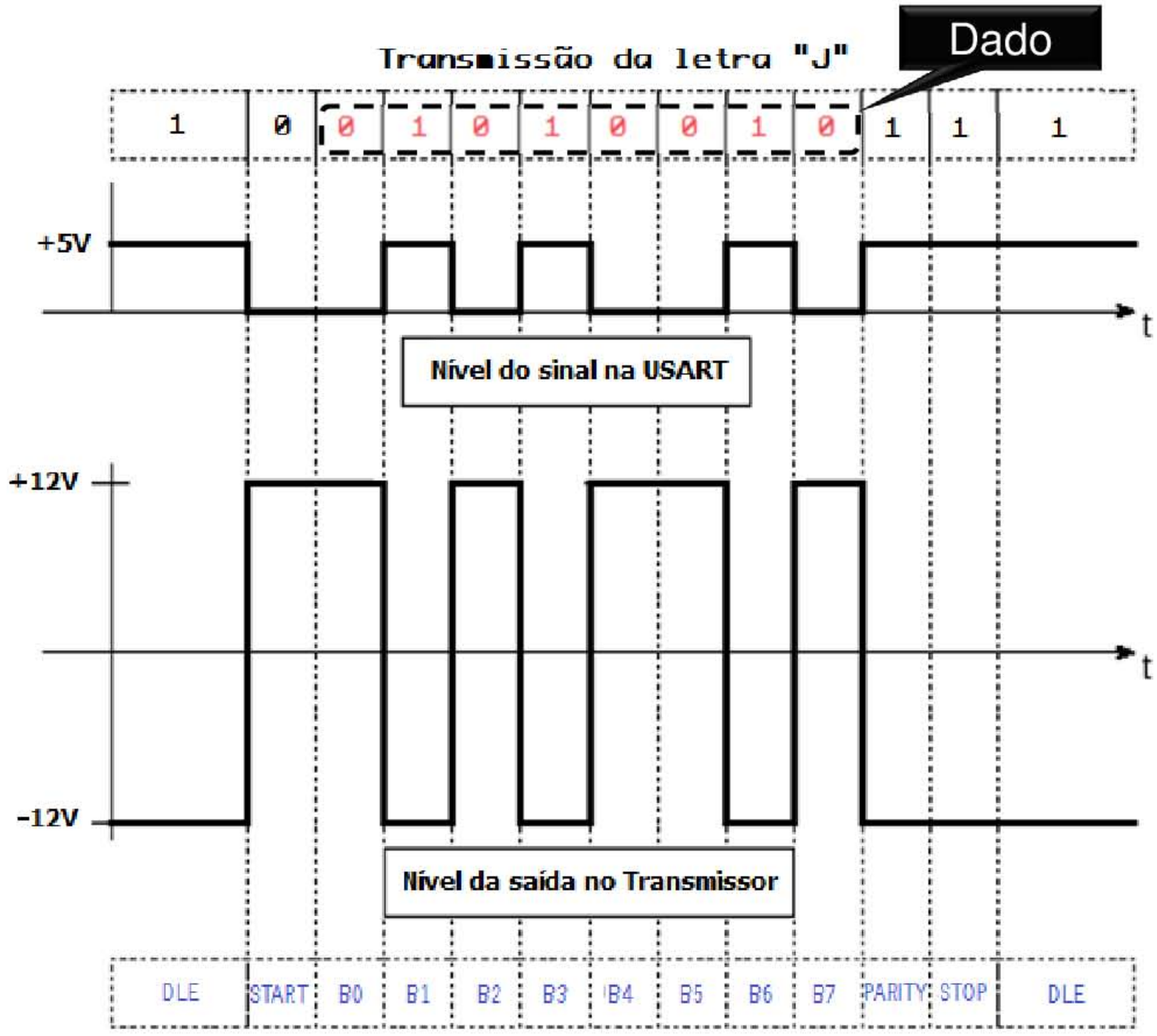
- Converte os níveis de tensão da USART (0 - 5 V) TTL para os níveis utilizados na RS232 (-15 a +15 V)



© Prof. Vargas

Níveis de tensão

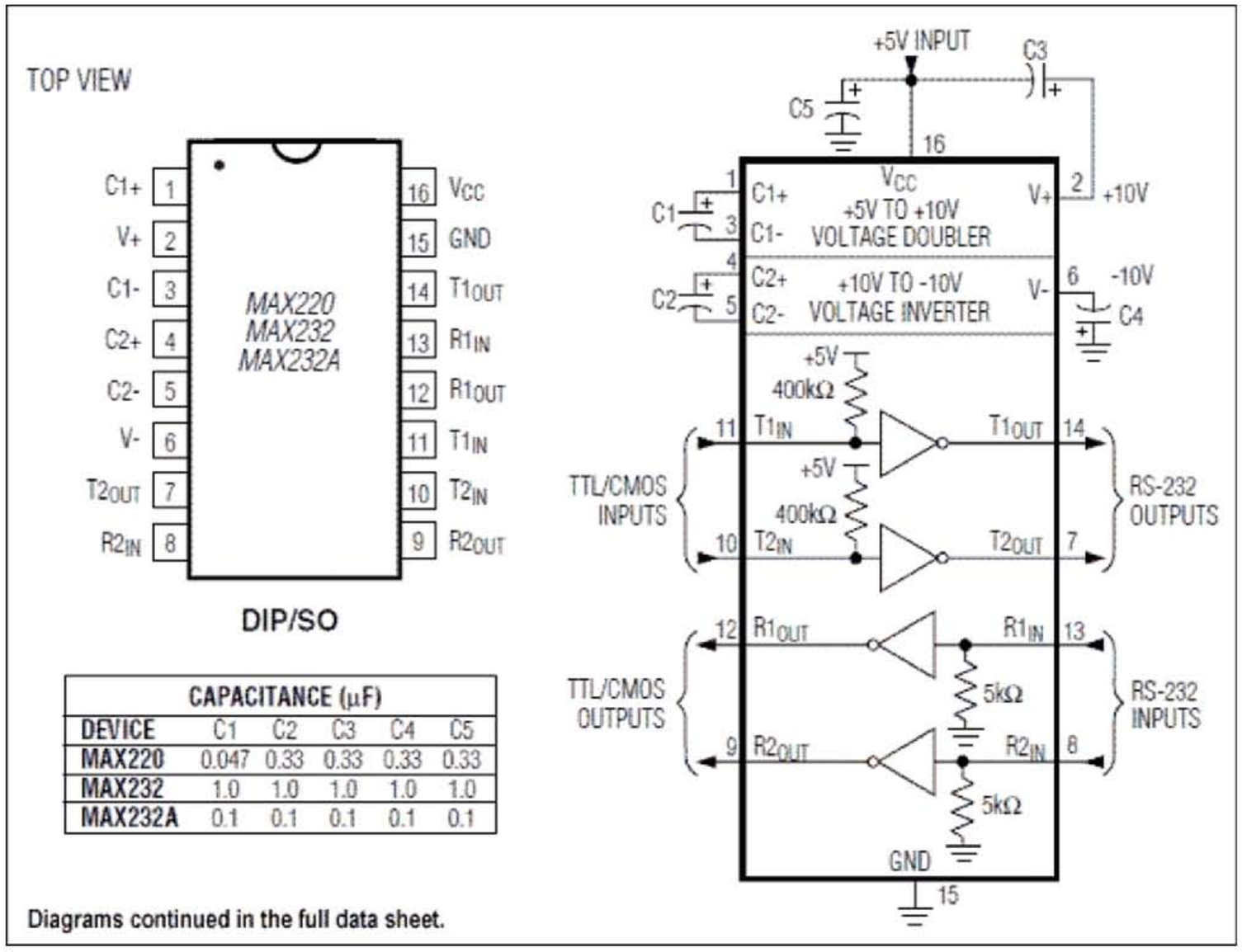
- Atualmente o padrão aceita de -12V a +12V



© Prof. Vargas

MAX-232

- A adequação de níveis de tensão podem ser obtidos com o uso do CI MAX232



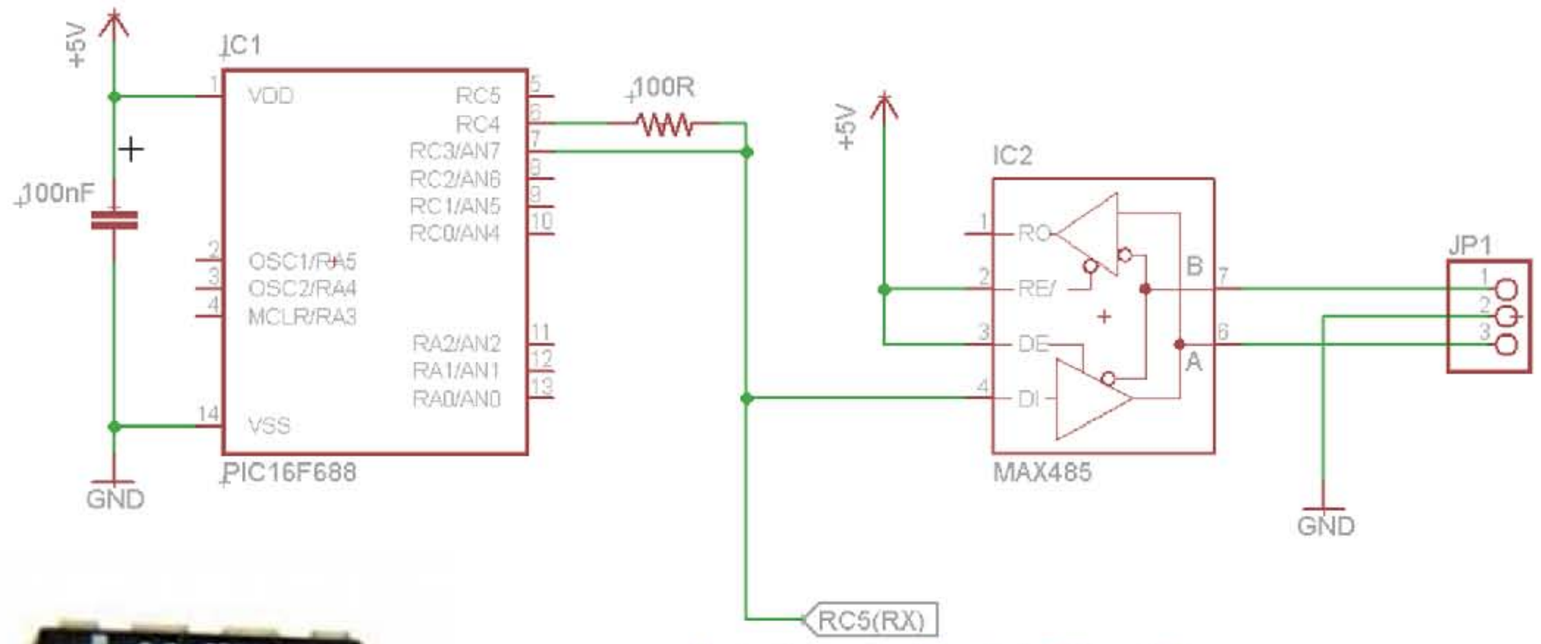
CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Diagrams continued in the full data sheet.

© Prof. Vargas

RS-485

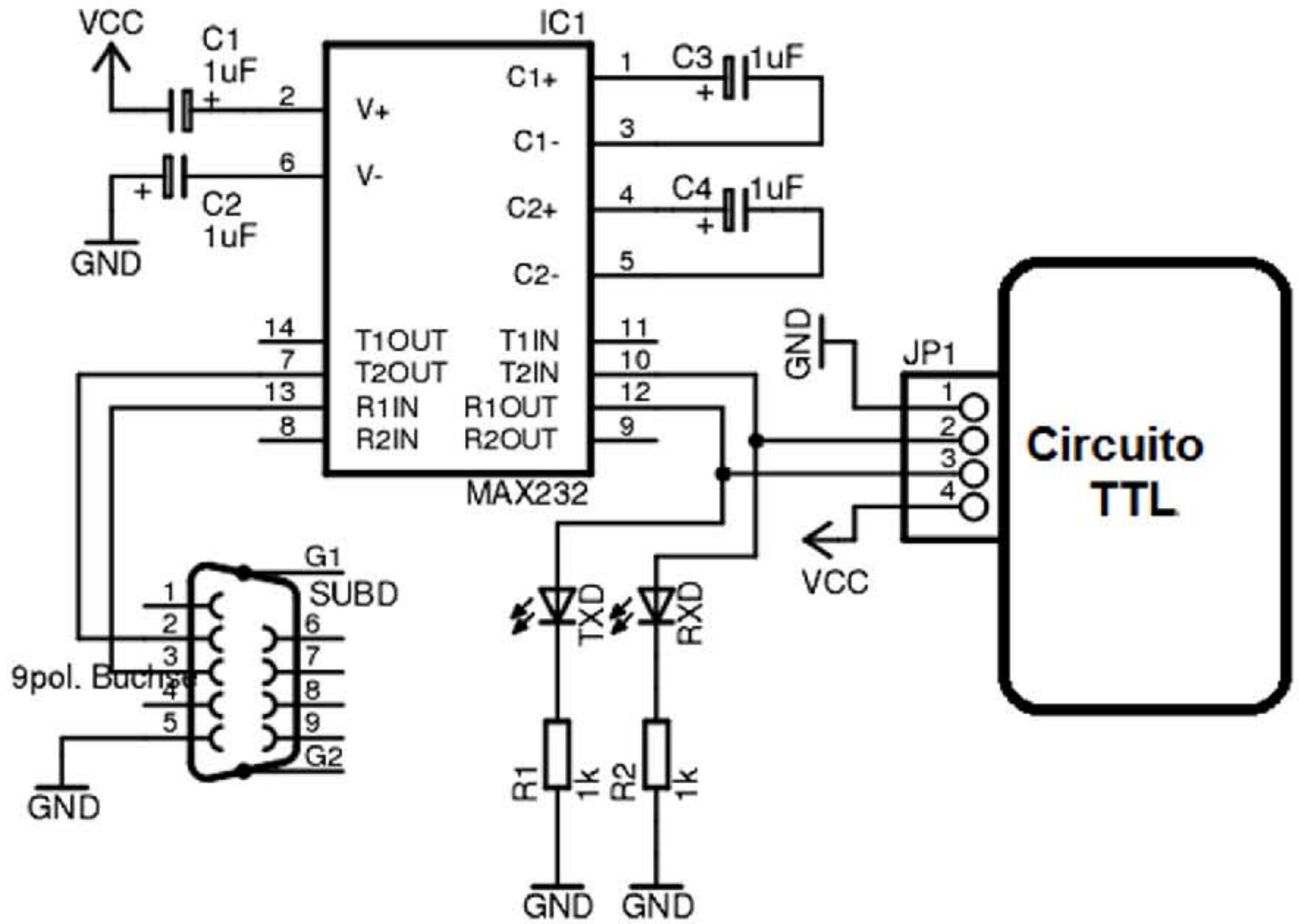
- Na indústria é mais comum o uso de RS-485



http://picprojects.org.uk/projects/dmx/dmx805/

© Prof. Vargas

Aplicação



http://baudaeletronica.blogspot.com.br/2012/05/max232.html

© Prof. Vargas

OS REGISTROS ENVOLVIDOS

© Prof. Vargas

RCSTA: RECEIVE STATUS AND CONTROL REGISTER

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

- **SPEN: Serial Port Enable**
  - 1 = Habilita Porta serial (configura RX/DT e TX/CK como Porta Serial)
  - 0 = Desabilita Porta Serial
- **RX9: 9-Bit Receive Enable**
  - 1 = Seleciona 9-bits na recepção
  - 0 = Seleciona 8-bits na recepção
- **SREN: Single Receive Enable**
  - **Asynchronous mode: (Não é usado)**
  - **Synchronous mode – Master**:
    - 1 = Habilita recepção simples
    - 0 = Desabilita

    Este bit é resetado após a recepção concluir.
  - **Synchronous mode – Slave:** (Não é usado)
- **CREN: Continuous Receive Enable**
  - **Asynchronous mode:**
    - 1 = Habilita receptor
    - 0 = Desabilita
  - **Synchronous mode:**
    - 1 = Habilita recepção contínua até **CREN=0** (CREN se sobrepõe a SREN)
    - 0 = Desabilita

© Prof. Vargas

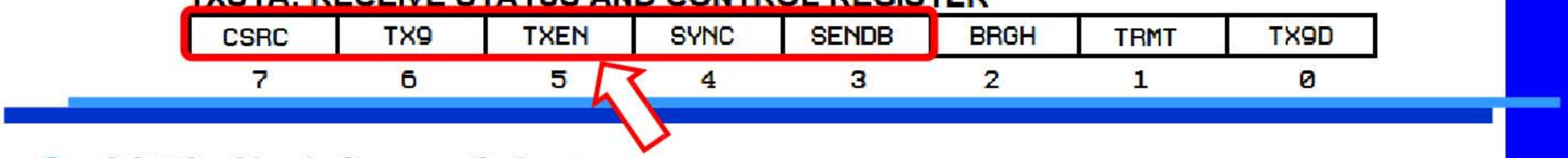
RCSTA: RECEIVE STATUS AND CONTROL REGISTER

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

- **ADDEN: Address Detect Enable**
  - **Asynchronous mode 9-bit (RX9 = 1):**
    - 1 = Habilita detecção de endereço, Habilita interrupção e carga do buffer de recepção quando RSR<8> = 1
    - 0 = Desabilita detecção de endereço, todos bits são recebidos e o 9º bit pode ser usado como paridade
  - **Asynchronous mode 8-bit (RX9 = 0):** (Não é usado)
- **FERR: Framing Error**
  - 1 = Framing error (pode ser atualizado lendo-se o registro RCREG e recebendo o próximo byte válido)
  - 0 = Sem erro de framing
- **OERR: Overrun Error**
  - 1 = Overrun error (pode ser limpo pondo **CREN = 0**)
  - 0 = sem erro de overrun
- **RX9D: 9th bit of Received Data**
  - Este pode ser bit de dado ou de endereço; ou ainda bit de paridade e deve ser calculado pelo firmware do usuário

© Prof. Vargas

TXSTA: RECEIVE STATUS AND CONTROL REGISTER



- **CSRC: Clock Source Select**
  - **Asynchronous mode:** (Não é usado)
  - **Synchronous mode:**
    - 1 = Master mode (Clock gerado internamente pelo BRG)
    - 0 = Slave mode (Clock de fonte externa)
- **TX9: 9-Bit Transmit Enable**
  - 1 = Seleciona 9-bit de transmissão
  - 0 = Seleciona 8-bit de transmissão
- **TXEN: Transmit Enable** (1)
  - 1 = Habilita transmissão
  - 0 = Desabilita
- **SYNC: EUSART Mode Select**
  - 1 = Modo Synchronous
  - 0 = Modo Asynchronous
- **SENDB: Send Break Character**
  - **Modo Asynchronous:**
    - 1 = Envia Sync Break na próxima transmissão (limpa por hardware após o término)
    - 0 = Sync Break transmissão concluída
  - **Modo Synchronous**: (Não é usado)

© Prof. Vargas

TXSTA: RECEIVE STATUS AND CONTROL REGISTER

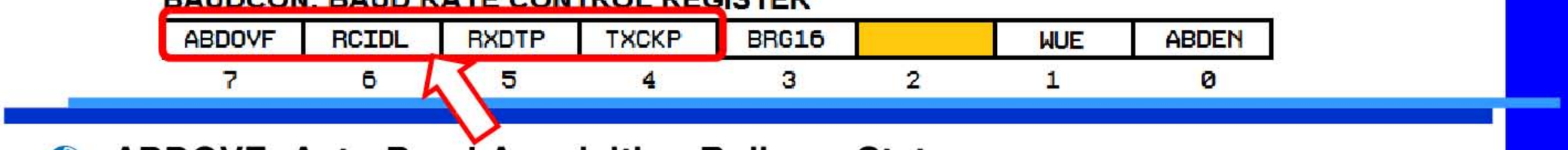
CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
7	6	5	4	3	2	1	0

- **BRGH: High Baud Rate Select**
  - **Modo Asynchronous**:
    - 1 = High speed
    - 0 = Low speed
  - **Modo Synchronous:** (Não é usado neste modo)
- **TRMT: Transmit Shift Register Status**
  - 1 = TSR empty
  - 0 = TSR full
- **TX9D: 9th bit of Transmit Data**
  - Pode ser Dado/Endereço ou bit paridade

Note 1: SREN/CREN se sobrepõe a TXEN no modo Sync com exceção se SREN não tiver efeito no sincronismo do modo Slave.

© Prof. Vargas

BAUDCON: BAUD RATE CONTROL REGISTER



ABDOVF: Auto-Baud Acquisition Rollover Status

- 1 = Um rollover BRG ocorreu durante a detecção no modo Auto-Baud Rate (deve ser limpo por software)
- 0 = Não ocorreu um rollover de BRG

RCIDL: Receive Operation Idle Status

- 1 = Recepção está ociosa
- 0 = Recepção está ativa

RXDTP: Received Data Polarity Select

- Modo Asynchronous:
  - 1 = Dado em RX está invertido
  - 0 = Não está
- Modo Synchronous:
  - 1 = Dado recebido (DT) está invertido. Estado ocioso é nível 0
  - 0 = Não houve inversão. Estado ocioso é nível 1.

TXCKP: Clock and Data Polarity Select

- Asynchronous mode:
  - 1 = Dado TX está invertido
  - 0 = Não está
- Synchronous modes:
  - 1 = Clock (CK) está invertido. Estado ocioso é nível 1.
  - 0 = Não está. Estado ocioso é nível 0.

© Prof. Vargas

BAUDCON: BAUD RATE CONTROL REGISTER

ABDOVF	RCIDL	RXDTP	TXCKP	BRG16		WUE	ABDEN
7	6	5	4	3	2	1	0

- **BRG16: 16-Bit Baud Rate Register Enable**
  - 1 = Gerador de Baud Rate de 16-bit – SPBRGH e SPBRG
  - 0 = Gerador de Baud Rate de 8-bit – só SPBRG (modo Compatível), SPBRGH é ignorado
- Não implementado. Lido como 0
- **WUE: Wake-up Enable**
  - **Modo Asynchronous**:
    - 1 = EUSART continuará a amostrar o pino RX – Interrupção gerada na borda de descida; o bit é limpo por hardware no ciclo seguinte de borda de descida
    - 0 = RX pin not monitored or rising edge detected
  - **Synchronous mode**: (Não é usado neste modo)
- **ABDEN: Auto-Baud Detect Enable**
  - **Asynchronous mode:**
    - 1 = Habilita medição de baud rate no próximo caractere. Requer recepção de um campo de Sync (55h); Limpo por hardware após o término.
    - 0 = Medição de Baud rate desligada ou complete.
  - **Synchronous mode**: (Não é usado neste modo)

© Prof. Vargas

Cálculo do Baud Rate

Config Bits	BRG/EUSART Mode	Formula Baud Rate		
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	FOSC/[64 (n + 1)]
0	0	0	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	

Por exemplo, para Fosc = 8 MHz, Baud rate de 9600, modo assíncrono , com 8 bit BRG:

$Baud\ Rate_{desejado} = F_{OSC}/(64\ ([SPBRGH:SPBRG] + 1))$

SPBRG

Isolando SPBRGH:SPBRG (Aqui denominado X):

$X = [F_{osc} \div (Baud\ Rate_{desejado} \times 64)\ ] - 1 = [8 \times 10^6 \div (9600 \times 64)] - 1 = [12.02] =$ **12**

$Baud\ Rate_{calculado} = 8 \times 10^6/(64\ (12 + 1)) =$ **9615**

$Erro = [(Baud\ Rate_{desejado} - Baud\ Rate_{calculado})/\ Baud\ Rate_{desejado}\ ] * 100 =$

$= [(9600 - 9615)/9600] \times 100 =$ **- 0.16 %**

© Prof. Vargas

Ou consultando uma tabela...

TABLE 20-3: BAUD RATES FOR ASYNCHRONOUS MODES

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0											
	FOSC = 40 MHz			FOSC = 20 MHz			FOSC = 10 MHz			FOSC = 8 MHz		
	Atual Rate (K)	% Error	SPBRG value (decimal)	Atual Rate (K)	% Error	SPBRG value (decimal)	Atual Rate (K)	% Error	SPBRG value (decimal)	Atual Rate (K)	% Error	SPBRG value (decimal)
0.3	–	–	–	–	–	–	–	–	–	–	–	–
1.2	–	–	–	1.221	1.73	255	1.202	0.16	129	1.201	-0.16	103
2.4	2.441	1.73	255	2.404	0.16	129	2.404	0.16	64	2.403	-0.16	51
9.6	9.615	0.16	64	9.766	1.73	31	9.766	1.73	15	9.615	-0.16	12
19.2	19.531	1.73	31	19.531	1.73	15	19.531	1.73	7			
57.6	56.818	-1.36	10	62.500	8.51	4	52.083	-9.58	2			
115.2	125.00	8.51	4	104.17	-9.58	2	78.125	-32.18	1			

Daqui depreendemos que **SBPRG** = 12
E carregamos:

```
SPBRG  = 0x0C
SPBRGH = 0x00
```

© Prof. Vargas

Passos para Transmissão

1. Definir Baud Rate (SPBRG)
2. Habilitar a transmissão Assíncrona:
   - ☒ RCSTA.SPEN = 1
3. Habilitar transmissão a 9 bits
   - ☒ TXSTA.TX9 = 1 ou não (=0)
4. Carregar o valor a ser transmitido em TXREG
5. Se usar 9 bits, carregue TXSTA.TX9D com o valor do bit

© Prof. Vargas

Passos para Recepção

1. Definir Baud Rate (SPBRG)
2. Habilitar a transmissão Assíncrona:
   - ☒ `RCSTA.SPEN = 1`
   - ☒ `TXSTA.SYNC = 0`
3. Habilitar recepção em 9 bits
   - ☒ `RCSTA.RX9 = 1`
4. Se usar interrupção:
   - ☒ `Use PIE1.RCIE = 1`
   - ☒ `INTCON.GIEH e GIEL`
   - ☒ `Habilitar recepção com RCSTA.CREN = 1`
5. PIR1.RCIF = 1 quando o byte chegar
   - ☒ `Não esqueça de limpar o flag (PIR1.RCIF = 0)`
6. Colete o dado recebido em RCREG

© Prof. Vargas