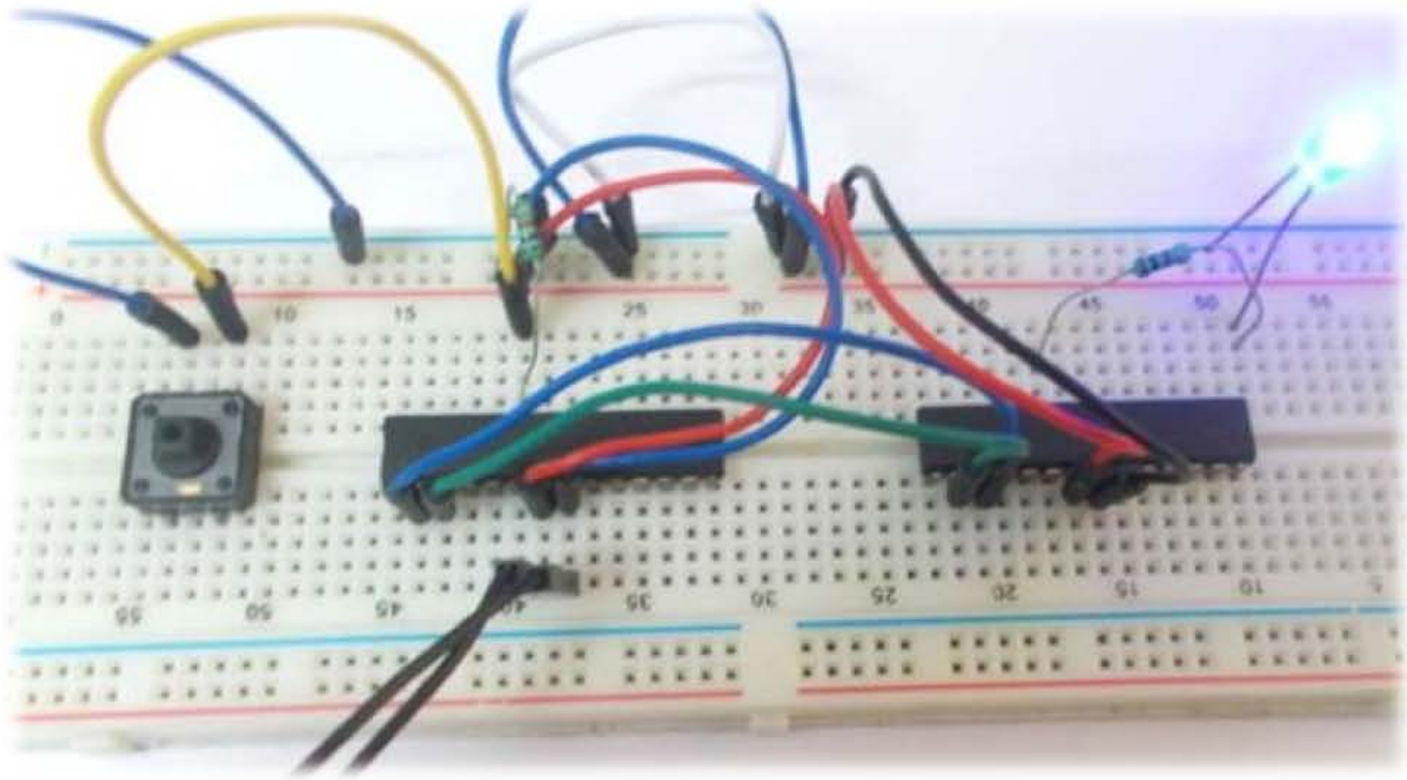
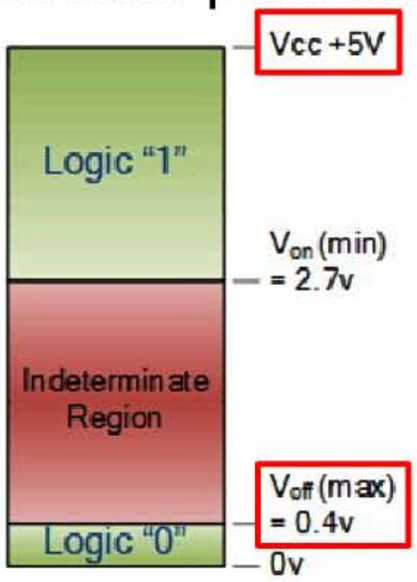
COMUNICACAO SERIAL



© Prof. Engº esp Luiz Antonio Vargas Pinto
www.vargasp.com
Rev. 01/2021

Características

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.

Canais de comunicação

- Caminhos para o trânsito da informação:
  - ☒ Fio elétrico de cobre
  - ☒ Ondas de rádio (Wireless)
  - ☒ LASER (Light Amplification by Stimulated Emission of Radiation)
  - ☒ Futuramente – comunicação quântica

Comunicação Serial

- Paralela x Serial
  - ☒Taxa de transferência paralela é mais rápida
  - ☒ Muitas linhas $\Rightarrow$ **R$** ↑
- Velocidade de envio (Baud Rate)
  - ☒Taxa em Kbps, Mbps (Bit Per Second)
  - ☒Exemplo:

Taxa de 9600 bauds ⇨ 9600 bits por segundo – Crédito a Èmile Baudot inventor do telégrafo

8 bits simultâneos (paralelo)

Tempo

Um bit de cada vez (Série)

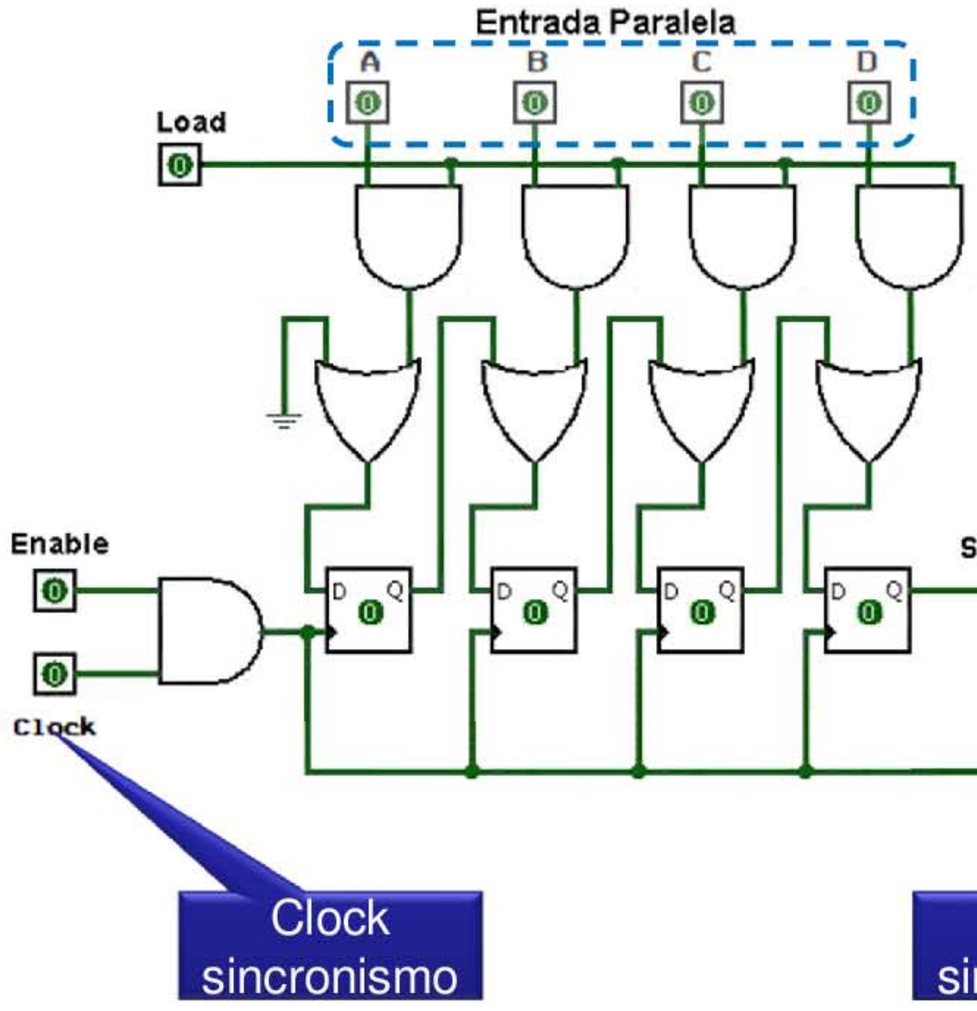
Tempo

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

Síncrono

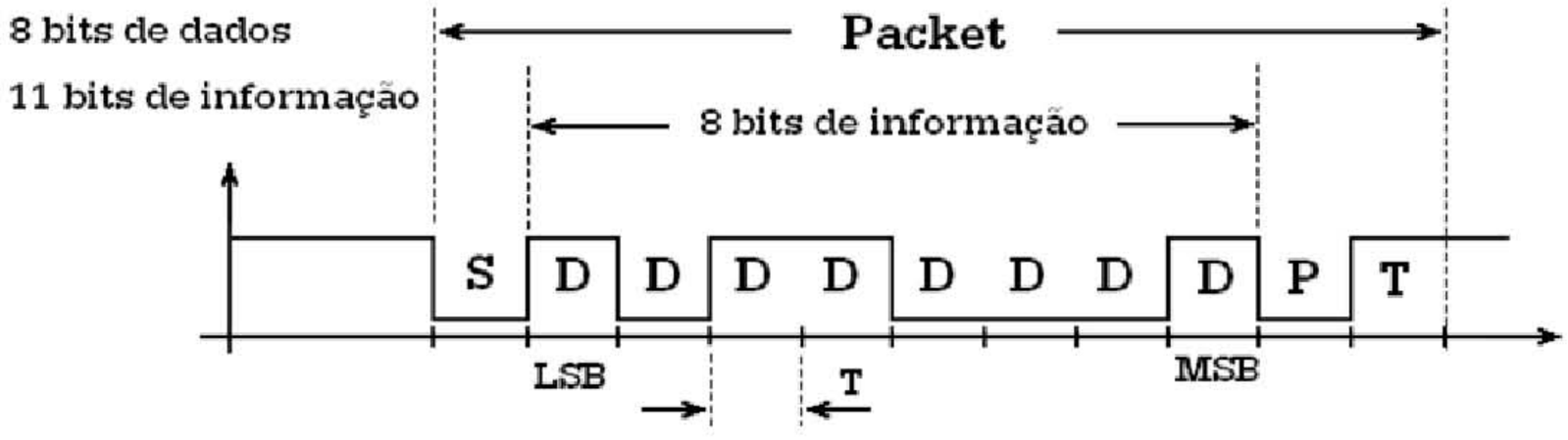
Origem



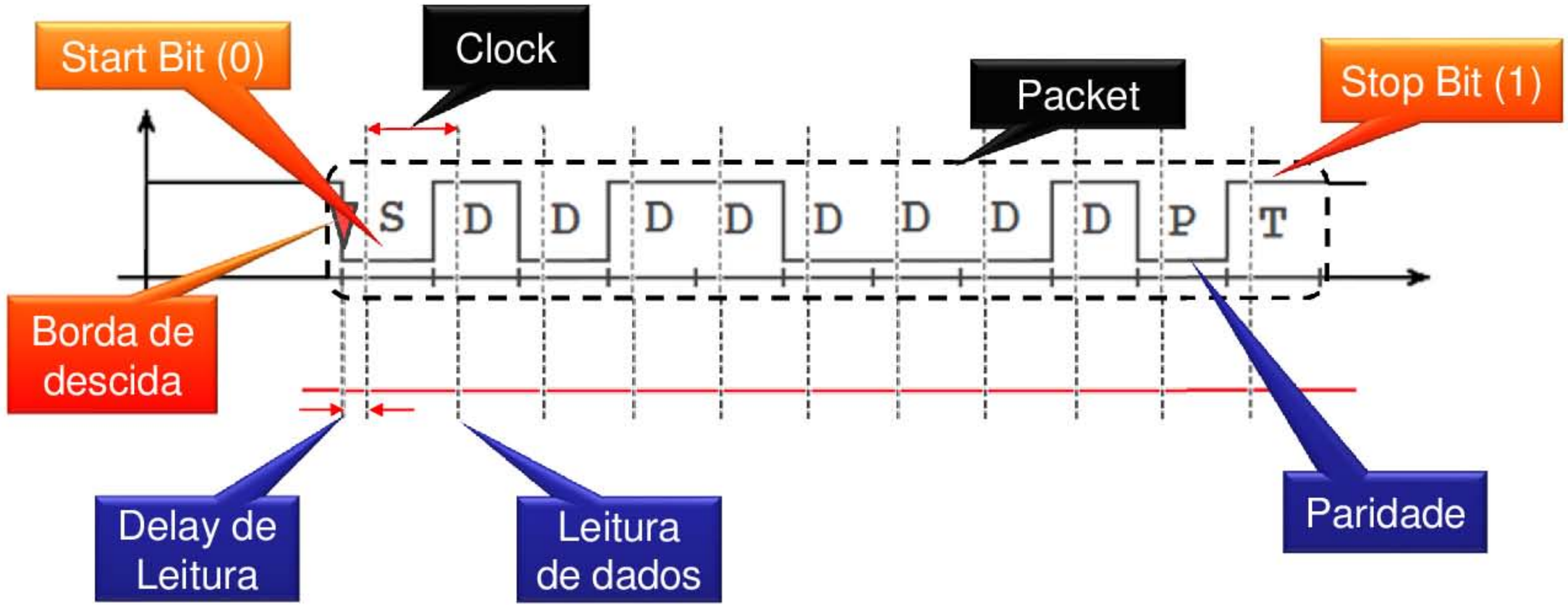
Destino

Comunicação Assíncrona

- Dado e clock trafegam em um único canal
- Clock de precisão em ambos Tx e Rx
- Packet mais comum: de 10 ou 11 bits com 8 de mensagem.
  - O comprimento do packet deve ser pequeno para minimizar variações dos osciladores Tx e Rx



Comunicação Assíncrona



- O packet é concluído com bits paridade e stop bit
- 1 ou 2 bits de erros podem ser corrigidos pelo receptor

Paridade

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

Dado	Bit de Paridade
10110010	0
10001010	1

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)

Checksum

Checksum

- **CHECKSUM** é uma técnica que insere na comunicação um byte de verificação
  - ☒ Σ (bytes) da mensagem completa em complemento de 2.
- 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
- Um número + (seu negativo) = 0
  - ☒ Complemento de um e/ou dois

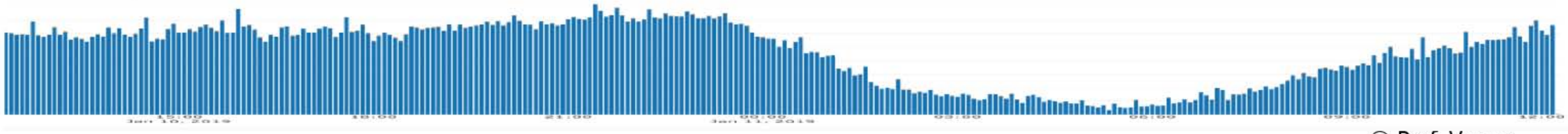
Checksum

```
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.

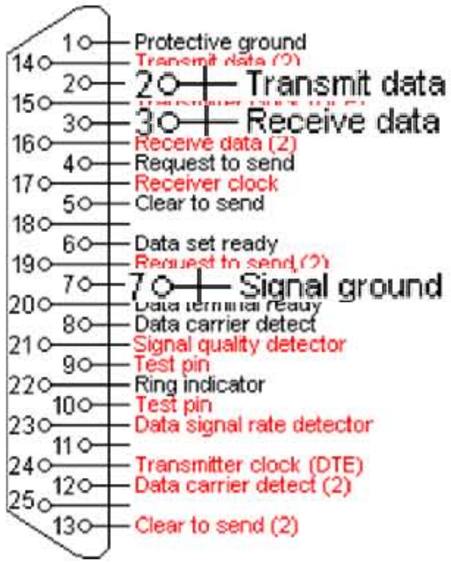


A INTERFACE SERIAL RS-232C

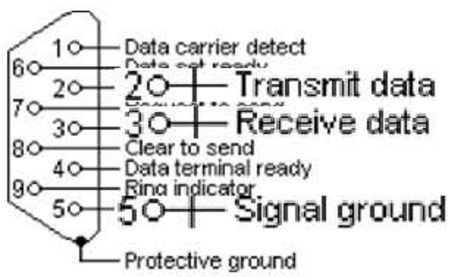
Interface Serial RS-232C

- 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).

Conectores



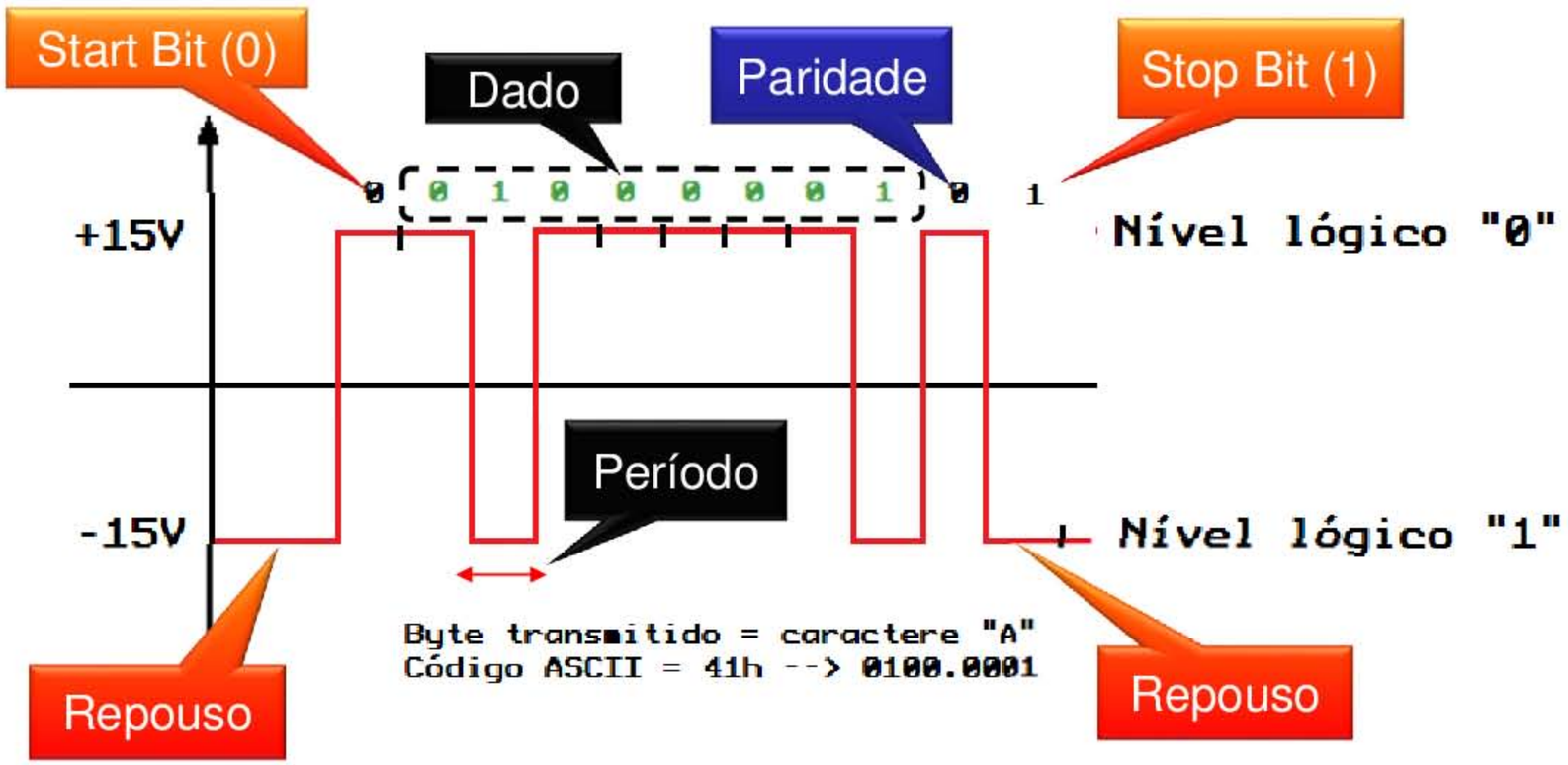
DB25



DB9

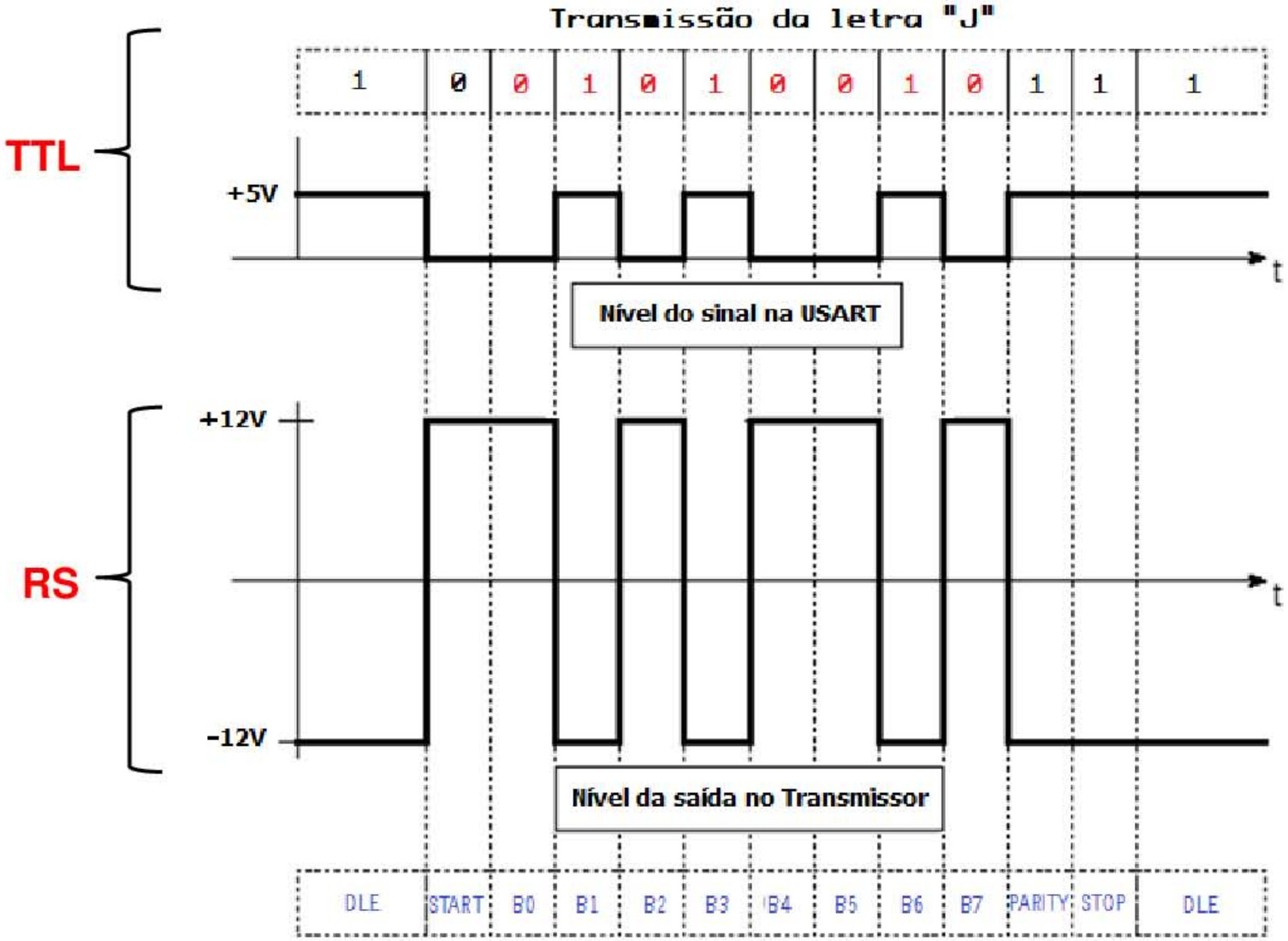
Níveis de tensão

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



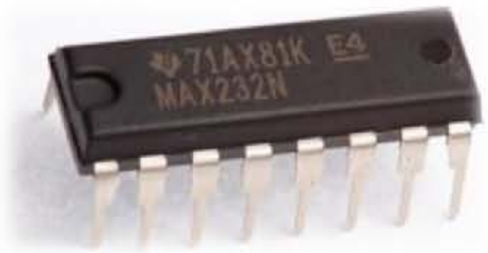
Níveis de tensão

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



MAX-232

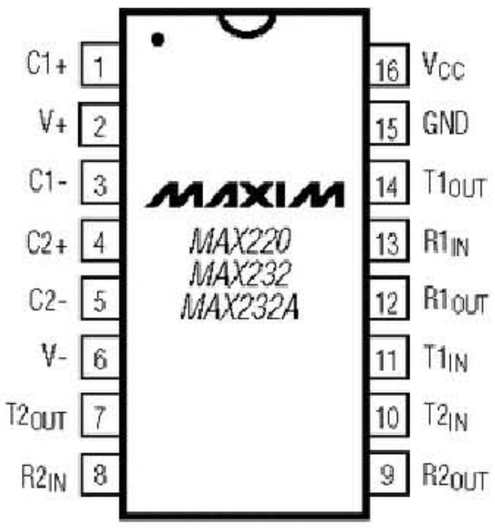
- A adequação de níveis de tensão podem ser obtidos com o uso do CI MAX232



DIP

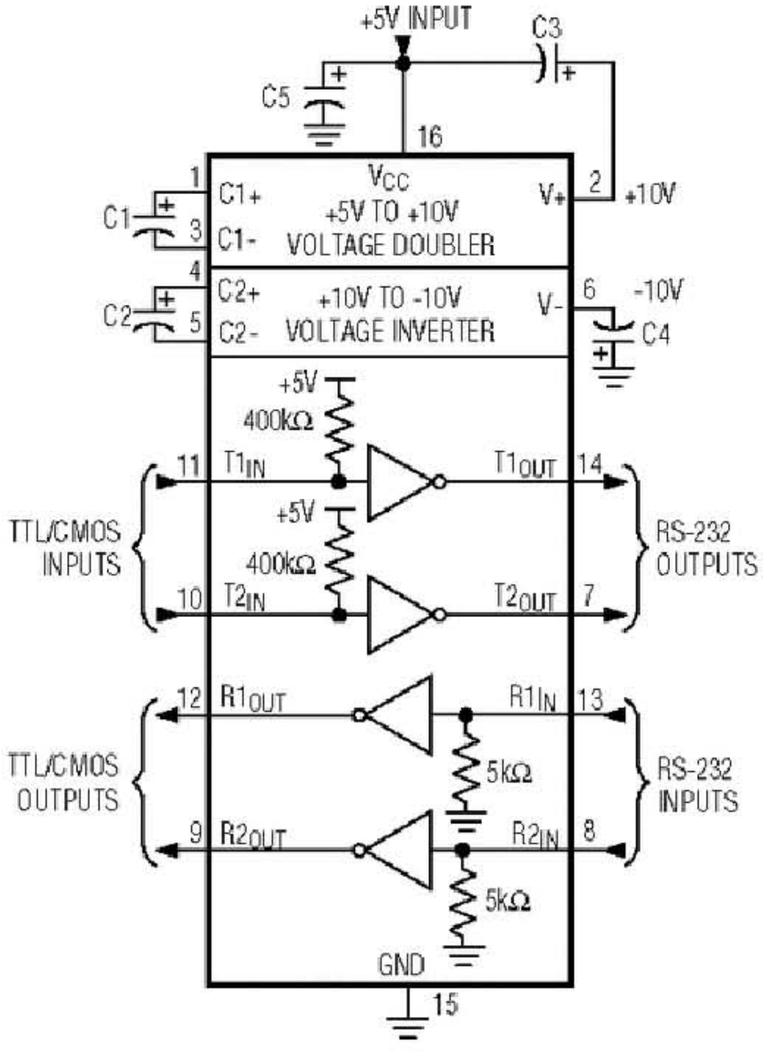
MAX232

SMD



DIP/SO

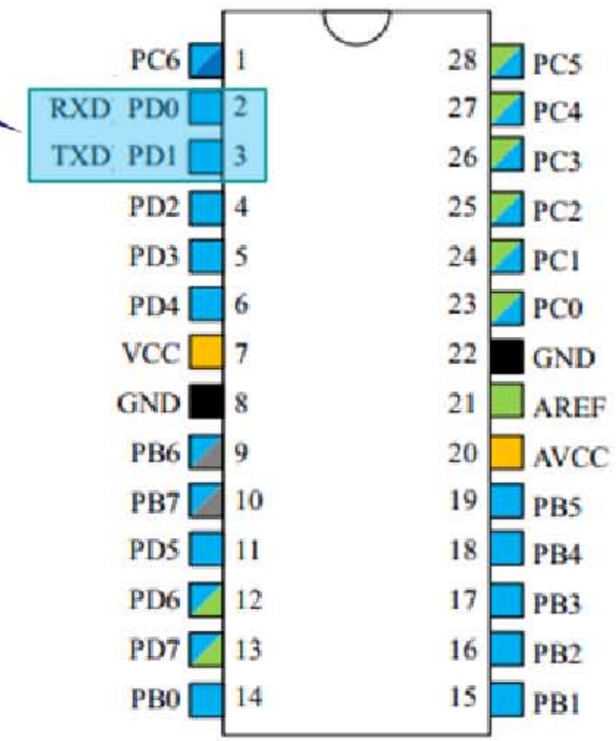
CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1



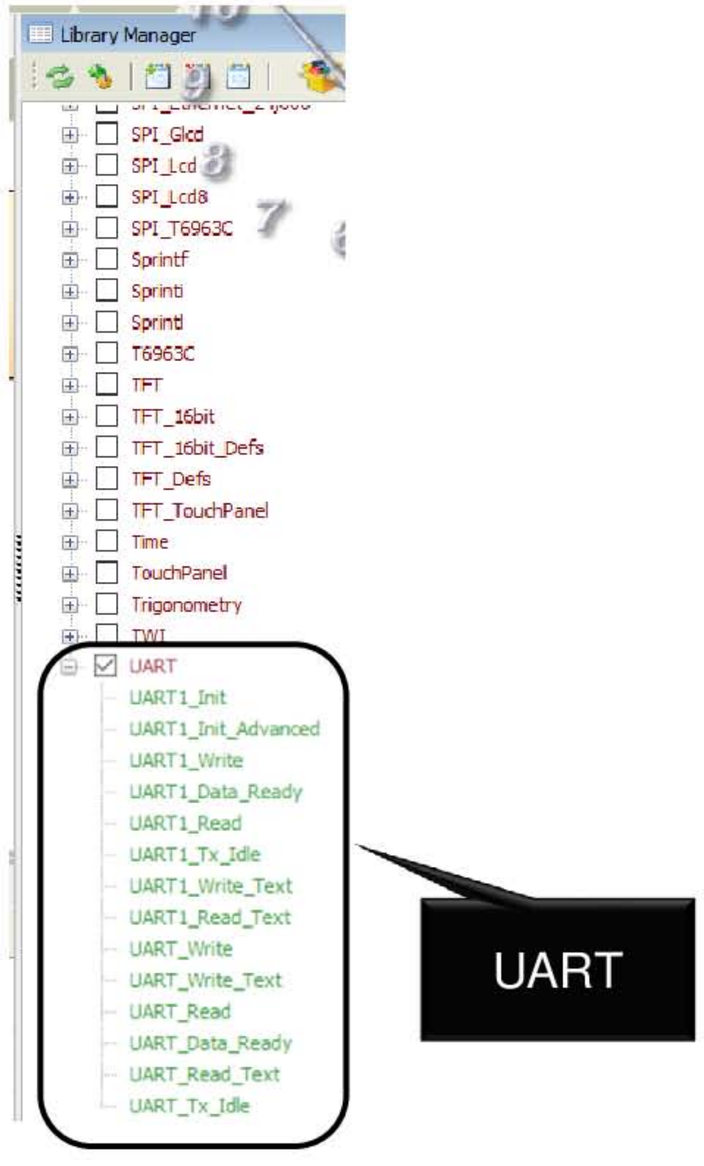
SERIAL RS-232C COM ATMEGA 328P

Pinagem

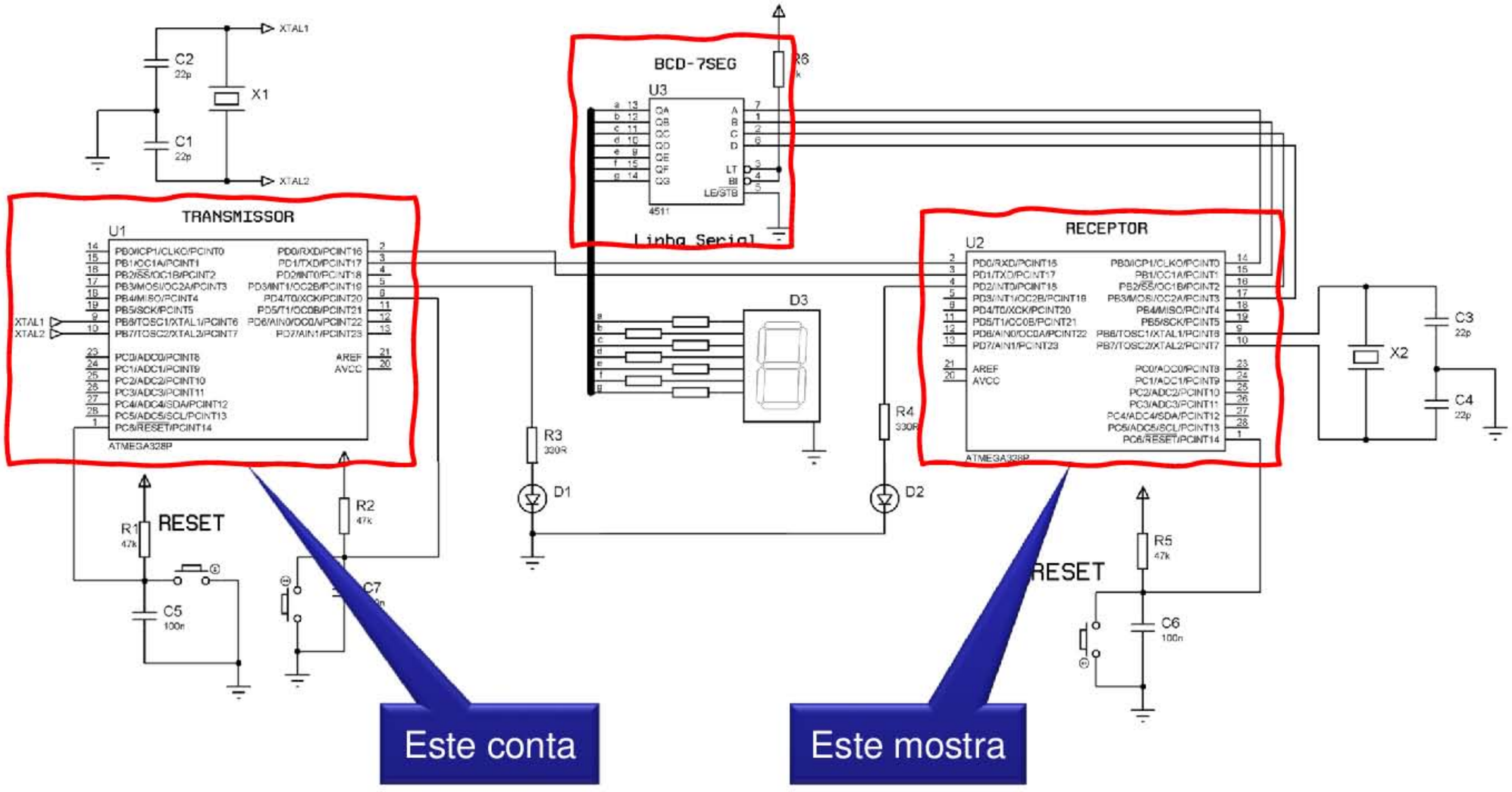
Comunicação serial



Biblioteca



Hardware de teste



Programa teste Transmissor

```
/*-----------------------------------------------------------------
  Programa : TEX.C
  Data     : 30/06/2020
  Autor    : Prof. Vargas
             Adaptado do Help do compilador MikroC
  Descrição:
             UART é configurada para 9600 bps.  Testa para ver se
             a UART está liberada.
             Se sim, envia o valor do contador para o destinatário
             Pisca o LED indicador de transmissão e atualiza o
             contador para a próxima transmissão
-----------------------------------------------------------------*/
char contador,uart_rd;
void main()
{
  UART1_Init(9600);          // Inicializa UARTa 9600 bps
  Delay_ms(100);             // Espera UART estabilizar
  PORTD3_bit = 0;
  DDD4_bit = 0;
  DDD3_bit = 1;
  contador=1;
  while (1)                  // loop permanente
  {
    if (Button(&PIND,4,1,0))
    {
      do                     //Aguarda tirar o dedo do botão
      {
      }while((Button(&PIND,4,1,0)));
      PORTD3_bit = 1;
      Delay_ms(80);          //Pisca o LED
      PORTD3_bit = 0;
      UART1_Write(contador);     // e envia via UART
      contador++;
      if(contador==10) contador=0;
    }
  }
}
```

Programa teste Receptor

```
/*-------------------------------------------------------
  Programa : REX.C
  Data     : 30/06/2020
  Autor    : Prof. Vargas
             Adaptado do Help do compilador MikroC
  Descrição:
             Esta, configurada com 9600 bps só responde ao
             destinatário se este enviar um pedido
-------------------------------------------------------*/
char uart_rd;

void main()
{
  DDRB = 0b11001111;
  UART1_Init(9600);                 // Inicializa UART a 9600 bps
  Delay_ms(100);                    // Espera UART estabilizar
  while (1)                         // loop permanente
  {
    if (UART1_Data_Ready())         // Aguarda uma recepção
    {
      uart_rd = UART1_Read();       // Lê dado recebido
      PORTB = uart_rd;              // envia para display
      Delay_ms(1);
    }
  }
}
```

Referências bibliográficas

1. Universidade Tuiuti – Paraná, Engenharia da Computação – 2003
2. Help do MikroC for AVR Mikroelektronic