Comunicação USART

DCA0210 - Linguagens Formais e Autômatos

Mário Sérgio Cavalcante[1]

[1] Departamento de Engenharia de Computação e Automação
mariocavalcante@dca.ufrn.br

Natal - Rio Grande do Norte

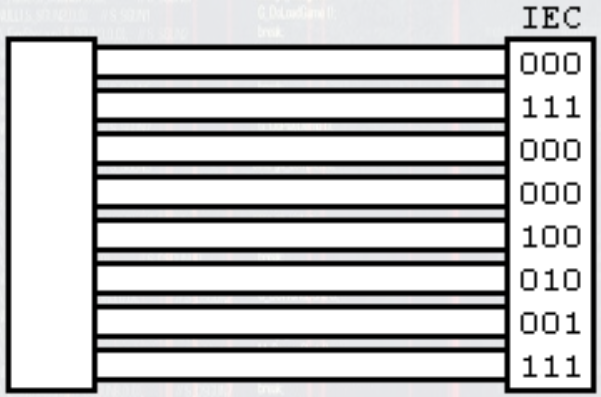
Sumário

Introdução

Paralelo:

Características:

- Maior Velocidade;
- Maior Custo;
- Mais susceptível a ruídos;
- Curtas distâncias.



Serial:

Características:

- Menor Velocidade;
- Menor Custo;
- Menos susceptível a ruídos;
- Longas distâncias.

USART - Características

- É um periférico de comunicação serial com inúmeras possibilidades de configurações de trabalho, o que lhe permite ser aplicada em uma infinidade de sistemas eletrônicos. Como por exemplo nas comunicações RS232 e RS485.
- No arduino é usada principalmete para a gravação do AtMega328 através do computador;

USART - Algumas características

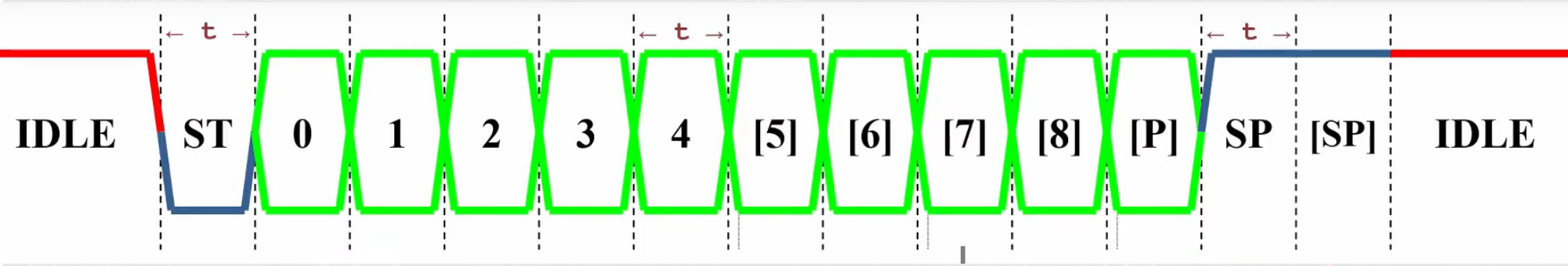
- A grande vantagem da USART é que muitos dispositivos eletrônicos antigos e modernos suportam seu protocolo de comunicação, inclusive em equipamentos industriais.
- Operação Full-Duplex;
- Operação Síncrona e Assíncrona.

USART - Algumas Características

- Suporta frames seriais com 5,6,7,8 ou 9 bits e 1 ou 2 bits de parada.
- Gerador de paridade par ou ímpar e conferência de paridade por hardware;
- Três fontes separadas de interrupção (transmissão completa, recepção completa e esvaziamento do registrador de dados).
- Pode ser utilizada como interface SPI mestre;
- UART é a versão simplificada síncrona.

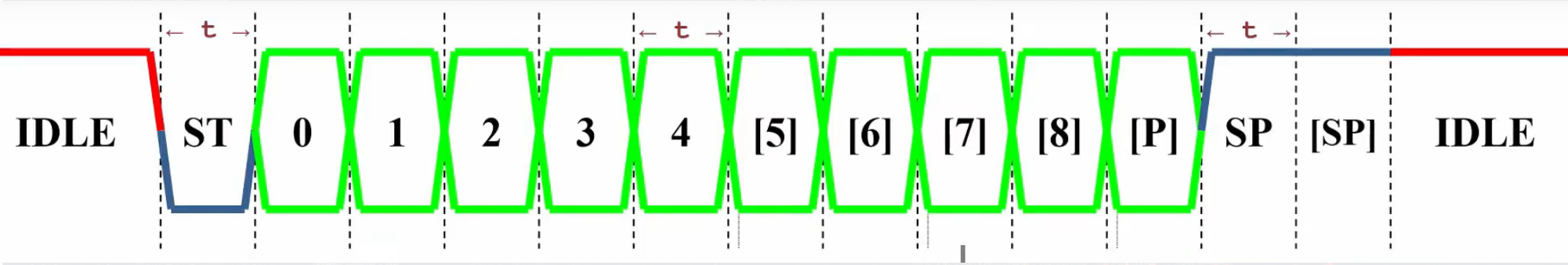
USART - Frame

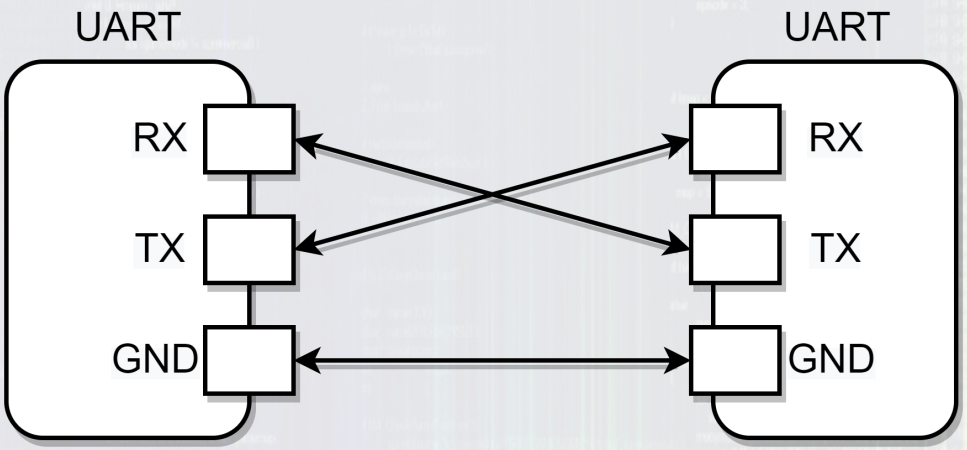
USART - Frame



- IDLE - Espera. Sem comunicação na linha. (Rx ou Tx), a linha deve ficar em nível lógico alto.
- ST - Bit de inicio (**start bit**). Sempre baixo (0).
- N - Bits de Dados (0 à 8);
- P - bit de paridade. Par ou ímpar;
- SP - Bit de Parada (**stop bit**). Sempre alto (1);
- t - Tempo do bit;
- [∗] - Opcional;

USART - Frame





2.1: Taxa

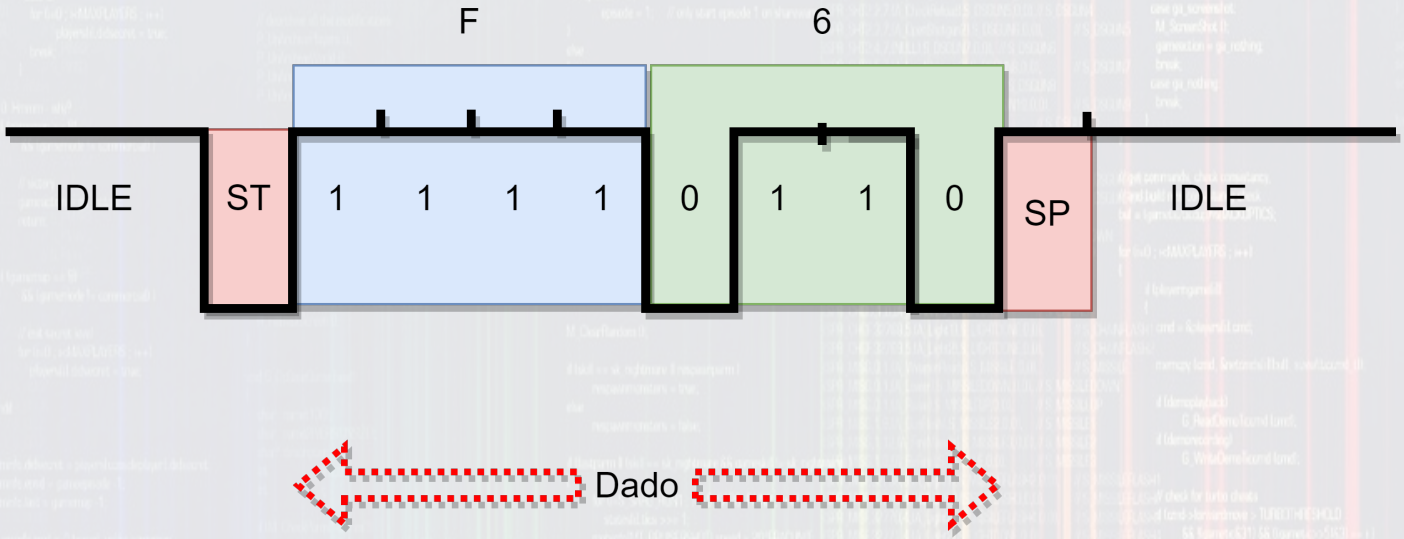
$$\text{taxa(bps)} = \frac{1}{t}$$

Observação 2.1

O transmissor e o receptor devem utilizar a mesma configuração de frames. □

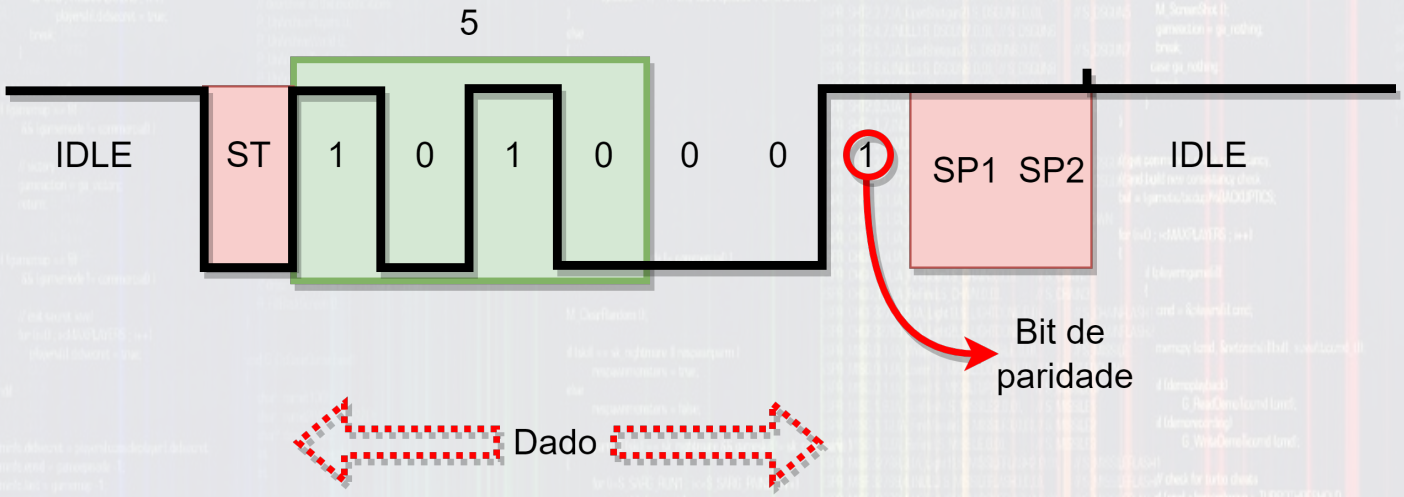
Frame USART

- Exemplo: Transmitir 0x6F, 8Bits, sem paridade, um stop bit.



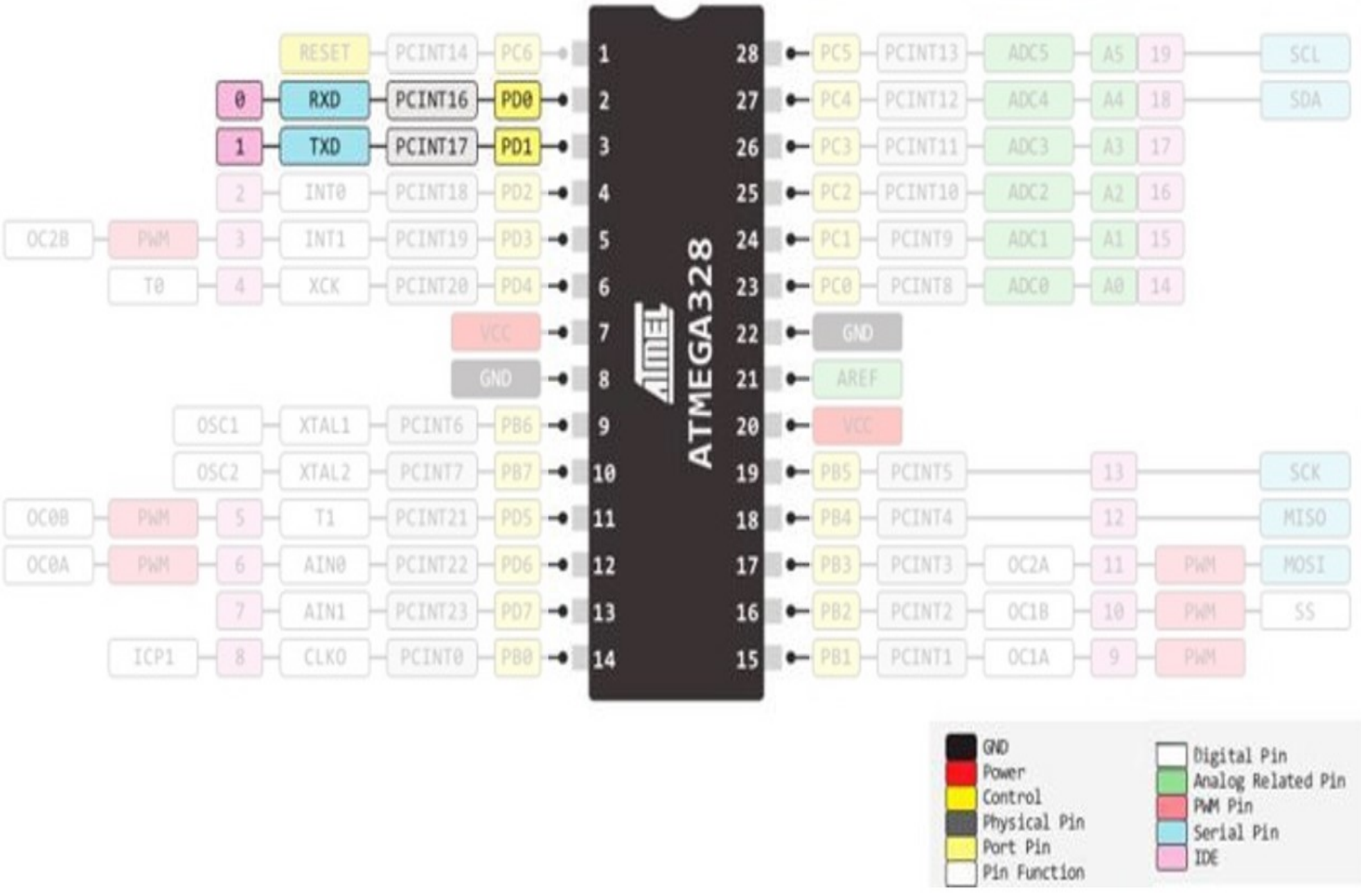
Frame USART

- Exemplo: Transmitir 0x05, 6Bits, paridade ímpar, dois stop bit.

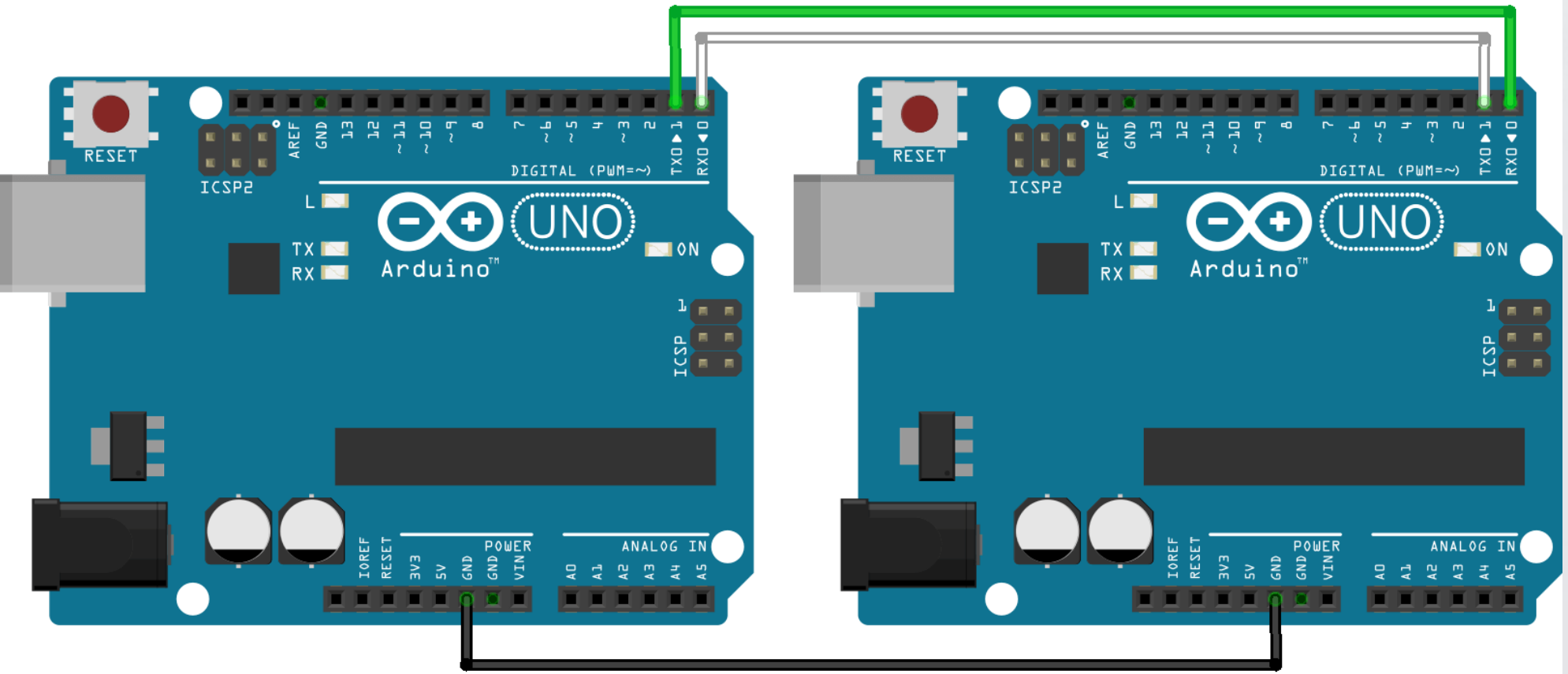


USART - ATmega328p

- Para utilizar a USART no microcontrolador, essencialmente, é necessário conhecer como:
  - Configurar (velocidade, interrupção, modo de operação, etc.);
  - Ler os dados recebidos;
  - Escrever os dados a serem transmitidos.

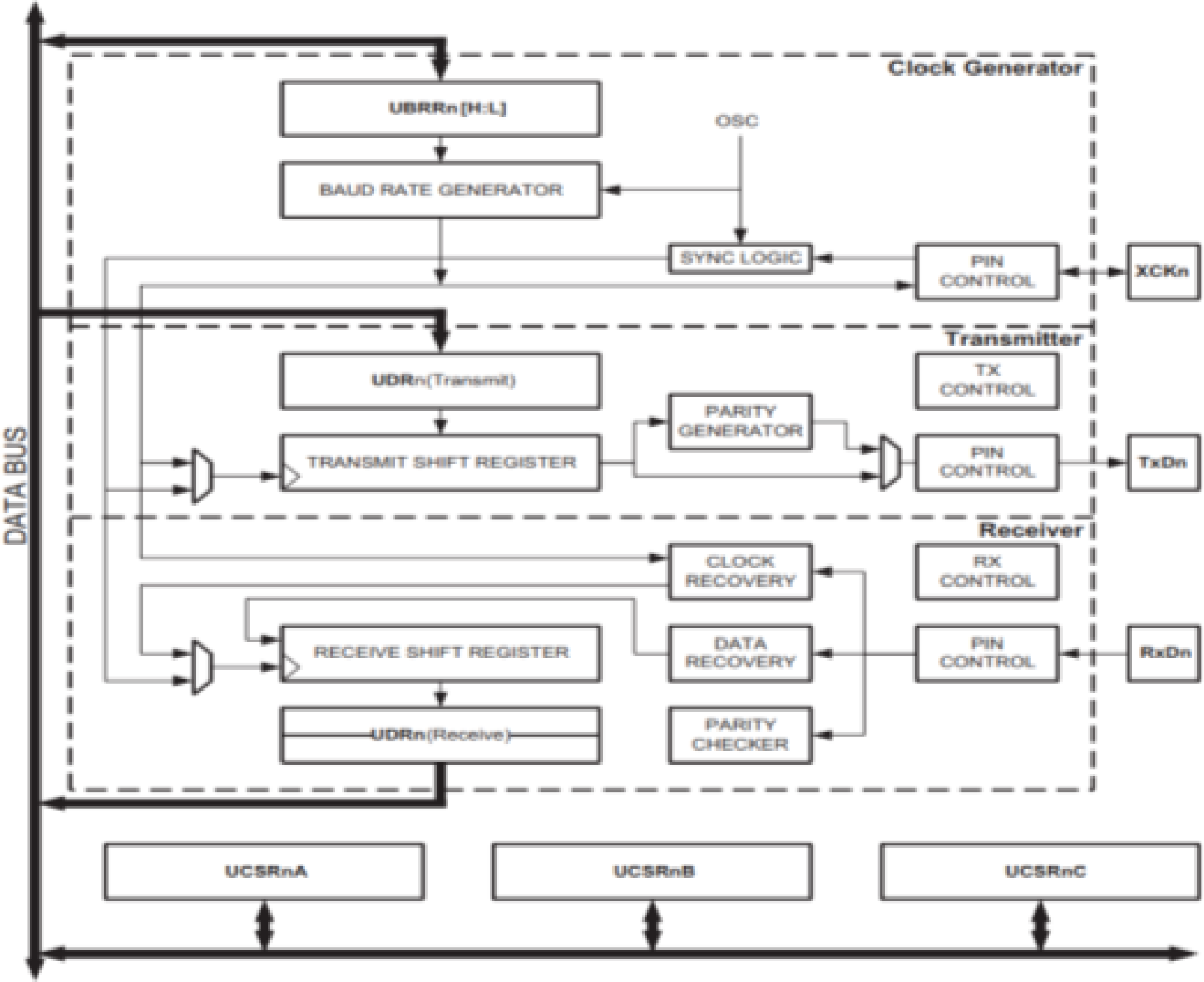

ATMEGA328
RESET PCINT14 PC6 1
0 RXD PCINT16 PD0 2
1 TXD PCINT17 PD1 3
2 INT0 PCINT18 PD2 4
OC2B PWM 3 INT1 PCINT19 PD3 5
T0 4 XCK PCINT20 PD4 6
VCC 7
GND 8
OSC1 XTAL1 PCINT6 PB6 9
OSC2 XTAL2 PCINT7 PB7 10
OC0B PWM 5 T1 PCINT21 PD5 11
OC0A PWM 6 AIN0 PCINT22 PD6 12
7 AIN1 PCINT23 PD7 13
ICP1 8 CLKO PCINT0 PB0 14
28 PC5 PCINT13 ADC5 A5 19 SCL
27 PC4 PCINT12 ADC4 A4 18 SDA
26 PC3 PCINT11 ADC3 A3 17
25 PC2 PCINT10 ADC2 A2 16
24 PC1 PCINT9 ADC1 A1 15
23 PC0 PCINT8 ADC0 A0 14
22 GND
21 AREF
20 VCC
19 PB5 PCINT5 13 SCK
18 PB4 PCINT4 12 MISO
17 PB3 PCINT3 OC2A 11 PWM MOSI
16 PB2 PCINT2 OC1B 10 PWM SS
15 PB1 PCINT1 OC1A 9 PWM
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE

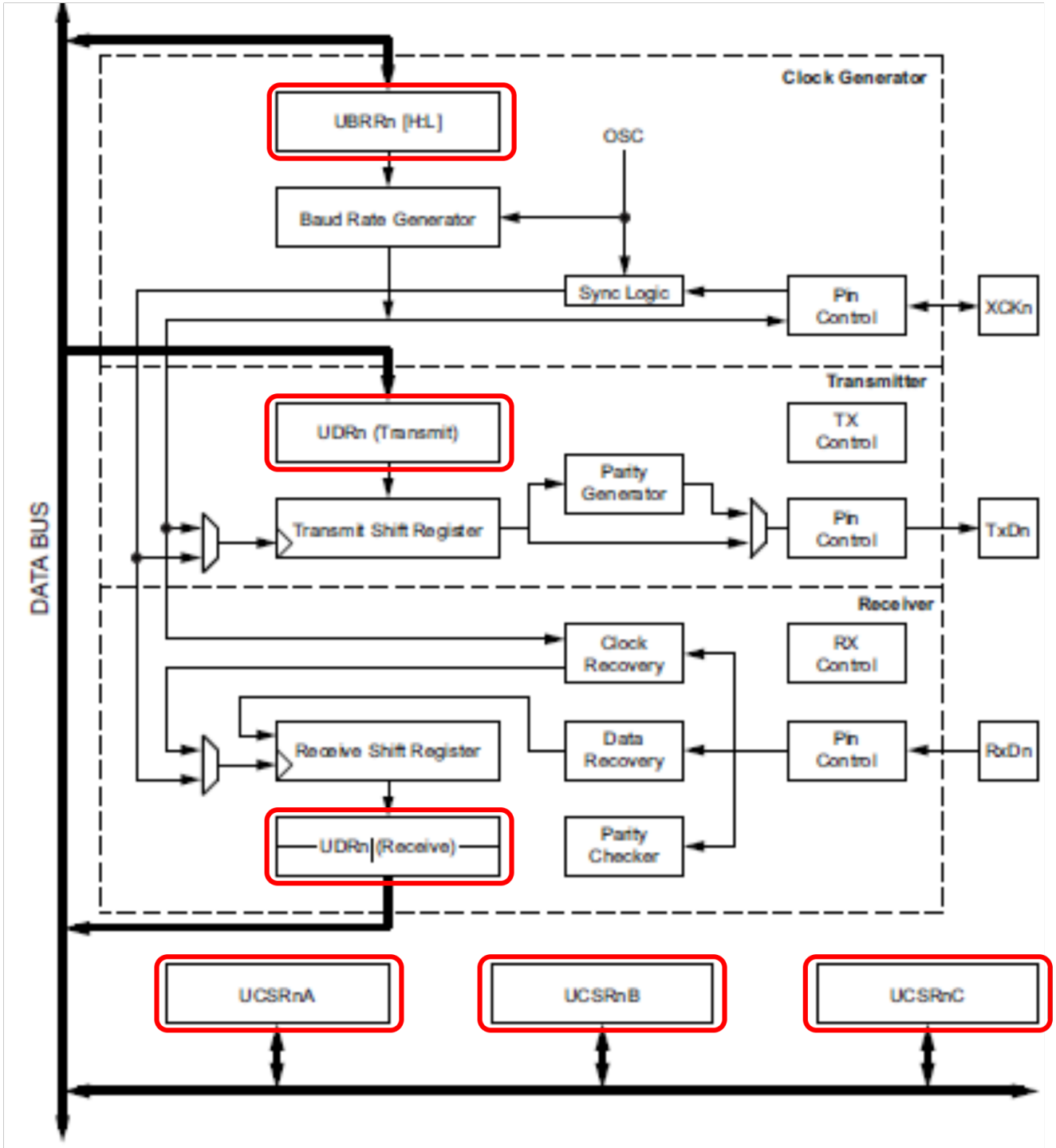
No Arduino:



fritzing

Registradores do USART


Clock Generator
UBRRn [H:L]
OSC
BAUD RATE GENERATOR
SYNC LOGIC
PIN CONTROL
XCKn
Transmitter
UDRn(Transmit)
TX CONTROL
PARITY GENERATOR
TRANSMIT SHIFT REGISTER
PIN CONTROL
TxDn
Receiver
CLOCK RECOVERY
RX CONTROL
RECEIVE SHIFT REGISTER
DATA RECOVERY
PIN CONTROL
RxDn
UDRn(Receive)
PARITY CHECKER
DATA BUS
UCSRnA
UCSRnB
UCSRnC



- UBSRRn
  - USART Baud Rate Register
- UDR0
  - Usart I/O Data Register
- UCSR0A
  - Usart Control and Status Register A
- UCSR0B
  - Usart Control and Status Register B
- UCSR0C
  - Usart Control and Status Register C

Registrador UBRR0H:L

UBRR0H	-	-	-	-	UBRR[11:8]
UBRR0L	UBRR[7:0]				

- **Bits 11:0**- **UBRR011:0: USART Baud Rate Register** - Este é o registrador de 12 bits que contém o valor da taxa de comunicação. Qualquer transmissão em andamento será corrompida se houver mudança desse valor. Qualquer escrita atualiza imediatamente a taxa de comunicação.

UBRR0L e UBRR0H – Usart Baud Rate Register

Modo de Operação	Equação para o cálculo da taxa de transmissão	Equação para o cálculo do valor de UBRR0
Modo Normal Assíncrono (U2X0 = 0)	$\text{TAXA} = \frac{f_{osc}}{16 * UBRR0 + 1}$	$UBRR0 = \frac{f_{osc}}{16 \cdot \text{TAXA}} - 1$
Modo de Velocidade Dupla Assíncrono (U2X0 = 1)	$\text{TAXA} = \frac{f_{osc}}{8 * UBRR0 + 1}$	$UBRR0 = \frac{f_{osc}}{8 \cdot \text{TAXA}} - 1$
Modo Mestre Síncrono	$\text{TAXA} = \frac{f_{osc}}{2 * UBRR0 + 1}$	$UBRR0 = \frac{f_{osc}}{2 \cdot \text{TAXA}} - 1$

UDR0 - USART I/O Data Register

Bit	7	6	5	4	3	2	1	0
UDR0 (Leitura)	RXB[7:0]							
UDR0 (Escrita)	TXB[7:0]							
Lê/Escreve	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
Valor Inicial	0	0	0	0	0	0	0	0

- Os registradores de recebimento e envio de dados possuem o mesmo endereço lógico. A distinção fica a cargo do Hardware.
- O UDR0 só deve ser escrito quando o bit **UDRE0** do registrador UCSR0A estiver ativo.

UCSR0A - Usart Control and Status Register A

Bit	7	6	5	4	3	2	1	0
UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Lê/Escreve	L	L/E	L	L	L	L	L/E	L/E
Valor Inicial	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Lê/Escreve	L	L/E	L	L	L	L	L/E	L/E
Valor Inicial	0	0	1	0	0	0	0	0

- BIT 7:6 - RXC0 e TXC0
  - Usart Receive Complete e Usart Transmit Complete. Quando eles estão como 1, indicam quando a recepção e a transmissão, respectivamente, foram completas.

UCSR0A - Usart Control and Status Register A

Bit	7	6	5	4	3	2	1	0
UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Lê/Escreve	L	L/E	L	L	L	L	L/E	L/E
Valor Inicial	0	0	0	0	0	0	0	0

- BIT 5 - **UDRE0**
  - **Usart Data Register Empty**: Quando esse bit está em 1 indica que o registrador de dados está vazio, ou seja, aconteceu uma transmissão ou uma recepção com sucesso.

UCSR0A - Usart Control and Status Register A

Bit	7	6	5	4	3	2	1	0
UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Lê/Escreve	L	L/E	L	L	L	L	L/E	L/E
Valor Inicial	0	0	0	0	0	0	0	0

- BIT 4 - **FE0**
  - **Frame Error**: É configurado como 1 quando acontece um erro nos frames. Por exemplo, erro na quantidade de stop-bits. Quando os frames do receptor e do transmissor não estão configurados de forma semelhante.

UCSR0A - Usart Control and Status Register A

Bit	7	6	5	4	3	2	1	0
UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Lê/Escreve	L	L/E	L	L	L	L	L/E	L/E
Valor Inicial	0	0	0	0	0	0	0	0

- BIT 3 - **DOR0**
  - **Data OverRun**: Ocorre quando o registrador de entrada está cheio, não foi lido e um novo bit de início é detectado Este bit deve sempre ser zerado quando se escreve no registrador UCSR0A

UCSR0A - Usart Control and Status Register A

Bit	7	6	5	4	3	2	1	0
UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Lê/Escreve	L	L/E	L	L	L	L	L/E	L/E
Valor Inicial	0	0	0	0	0	0	0	0

- BIT 2 - **UPE0**
  - **Usart Parity Error**: Indica se existe um erro de paridade no dado recebido

UCSR0A - Usart Control and Status Register A

Bit	7	6	5	4	3	2	1	0
UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Lê/Escreve	L	L/E	L	L	L	L	L/E	L/E
Valor Inicial	0	0	0	0	0	0	0	0

- BIT 1 - **U2X0**
  - **Double the Usart Transmission Speed**: Este bit só tem efeito no modo de operação assíncrona - Dobra a velocidade da comunicação.

UCSR0A - Usart Control and Status Register A

Bit	7	6	5	4	3	2	1	0
UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Lê/Escreve	L	L/E	L	L	L	L	L/E	L/E
Valor Inicial	0	0	0	0	0	0	0	0

- BIT 0 - **MPCM0**
  - **Multi processor Communication Mode**: comunicação com vários processadores Quando ativo, todos os frames recebidos serão ignorados se não contiverem uma informação de endereço

UCSR0B - Usart Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- BIT 7:6 - **RXIEn e TXIEn**
  - **RX Complete Interrupt Enable e TX Complete Interrupt Enable**: Para habilitar as interrupções de recepção e transmissão é necessário setar esses bits como 1. As interrupções só irão acontecer se o **bit I** do **SREG** estiver ativo

UCSR0B - Usart Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- BIT 5 - **UDRIE**
  - **UDRIE** Este bit habilita a interrupção por registrador de dados vazio (bit UDRE 0).
- BIT 4:3 - **RXENn e TXENn**
  - **Receiver e Transmiter Enable**: Estes bits habilitam a recepção e a transmissão.

UCSR0C - Usart Control and Status Register C

UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- BIT 7:6 - **UMSELn1 e UMSELn0**
  - **UMSEL01 e UMSEL00** : USART Mode Select Estes bits selecionam o modo de operação da USART.

UMSEL01	UMSEL00	Modo de Operação
0	0	Assíncrono
0	1	Síncrono
1	0	Reservado
1	1	SPI Mestre

UCSR0C - Usart Control and Status Register C

UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- Bits 5:4 - **UPM01:0**
  - **Parity Mode**: Estes bits habilitam e ajustam o gerador de paridade e de conferência Se habilitado, o transmissor irá gerar e enviar o bit de paridade em cada frame o receptor irá gerar o valor de paridade para comparação Se uma desigualdade for detectada, o bit UPE 0 torna se ativo

UPM01	UPM00	Modo de Operação
0	0	Desabilitado
0	1	Reservado
1	0	Habilitado, Paridade par
1	1	Habilitado, Paridade impar

UCSR0C - Usart Control and Status Register C

UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- Bits 3 - **USB0S0**
  - **Stop bit Select**: Este bit seleciona o número de bits de parada a serem inseridos pelo transmissor.

USBS0	Stop BIT
0	1 Stop Bit
1	2 Stop Bit

UCSR0C - Usart Control and Status Register C

UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- Bits 2:1 - **UCSZ02:0** - Estes bits, combinados com o bit UCSZ02 do registrador UCSR0B, ajustam o número de bits de dados no frame do transmissor e receptor, conforme a tabela ao lado

UCSZ02	UCSZ01	UCSZ00	Tamanho do Caracter
0	0	0	5 Bits
0	0	1	6 Bits
0	1	0	7 Bits
0	1	1	8 Bits
1	0	0	Reservado
1	0	1	Reservado
1	1	0	Reservado
1	1	1	9 bits

UCSR0C - Usart Control and Status Register C

UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- Bits 2:1 - **UCPOL 0**
  - **Clock Polarity**: Este bit é usado somente no modo síncrono. Deve ser zero quando o modo assíncrono é usado Ele ajusta o sinal de clock (XCK) para amostragem e saída de dados

UCPOL0	Mudança do Dado Transmitido (Saída do pino TxD0)	Amostragem do Dado Recebido (Entrada do Pino RxD0)
0	Borda de subida de XCK	Borda de descida de XCK
1	Borda de descida de XCK	Borda de subida XCK

USART

- Existem muitas possibilidades de configurações e uso da USART Para facilitar o entendimento e uso, os estudos seguintes serão direcionados para as configurações mais comuns.
- Serão utilizadas as seguintes configurações:
  - Comunicação assíncrona;
  - Oito Bits de Dados;
  - Sem Bit de paridade;
  - Um stop bit;
  - Velocidade de 2.400 a 115.200 bps;

Iniciando a configuração

Observação 4.1

$$UBRR0 = \frac{f_{osc}}{16 \cdot \text{TAXA}} - 1$$

□

```
1  #define F_CPU 16000000UL
2  #define BAUD 9600
3  #define MYUBRR F_CPU/16/BAUD-1
4  #include <avr/io.h>
5
6  //Função para inicialização da USART
7  void UART_Init(void){
8      UBRR0H = (uint8_t) (MYUBRR >>8); //Ajusta a taxa de transmissão
9      UBRR0L = (uint8_t) (MYUBRR);
10     UCSR0A = 0; // Desabilita velocidade dupla.
11     UCSR0B = (1 << RXEN0) | (1  << TXEN0); //Habilita o transmissor e o receptor
12     UCSR0C = (1 << UCSZ01) | (1 << UCSZ00); // Ajusta o formato do frame:
13     //8 bits de dados e 1 bit de parada
14 }
```

Iniciando a configuração

Observação 4.2

$$UBRR0 = \frac{f_{osc}}{16 \cdot \text{TAXA}} - 1$$

□

```
1  #define F_CPU 16000000UL
2  #define BAUD 9600
3  #define MYUBRR F_CPU/16/BAUD-1
4  #include <avr/io.h>
5
6  //Função para inicialização da USART
7  void UART_Init(void){
8      UBRR0H = (uint8_t) (MYUBRR >>8); //Ajusta a taxa de transmissão
9      UBRR0L = (uint8_t) (MYUBRR);
10     UCSR0A = 0; // Desabilita velocidade dupla.
11     UCSR0B = (1 << RXEN0) | (1  << TXEN0); //Habilita o transmissor e o receptor
12     UCSR0C = (1 << UCSZ01) | (1 << UCSZ00); // Ajusta o formato do frame:
13     //8 bits de dados e 1 bit de parada
14 }
```

Transmissão - Tx

```
16  //Verifica se novo dado pode ser enviado pela UART
17  //Retorna valor 32 se novo dado pode ser enviado ou Zero caso não
18  uint8_t uartTxOk(void){
19      return (UCSR0A & (1 << UDRE0));
20  }
21
22  //Envia um byte pela porta UART
23  void uart_Transmit (uint8_t data){
24      UDR0 = data;//Coloca o dado no registrador de transmissão e o envia
25  }
26
27  //Envia uma string pela porta UART.
28  void uartString (char *c){
29      for(;*c!=0;c++){
30          while(!uartTxOk());
31          uart_Transmit(*c);
32      }
33  }
```

Recepção - Rx

```
34  /*Verifica se UART possui novo dado
35  Retorna valor 128 se existir novo dado recebido. Zero se não
36  */
37  uint8_t uartRxOk(void){
38      return (UCSR0A & (1<< RXC0));
39  }
40  //Ler byte recebido na porta UART
41  uint8_t uartRX(){
42      return UDR0;
43  }
44
```

Recebendo um Byte

```
45  int main(){
46      uint8_t dado_rx; //Variável para armazernar dado recebido.
47      UART_Init();
48      uartString("Envie qualquer caracter\r \n");
49      while(1){
50          if(uartRxOk()){ // Verifica se existe novo dado
51              dado_rx = uartRX(); //Armazena dado
52              uartString("Você enviou: ");
53              while (!uartTxOk());
54              uart_Transmit(dado_rx);//envia o caracter recebido
55              uartString("\r\n");
56          }
57      }
58  }
```

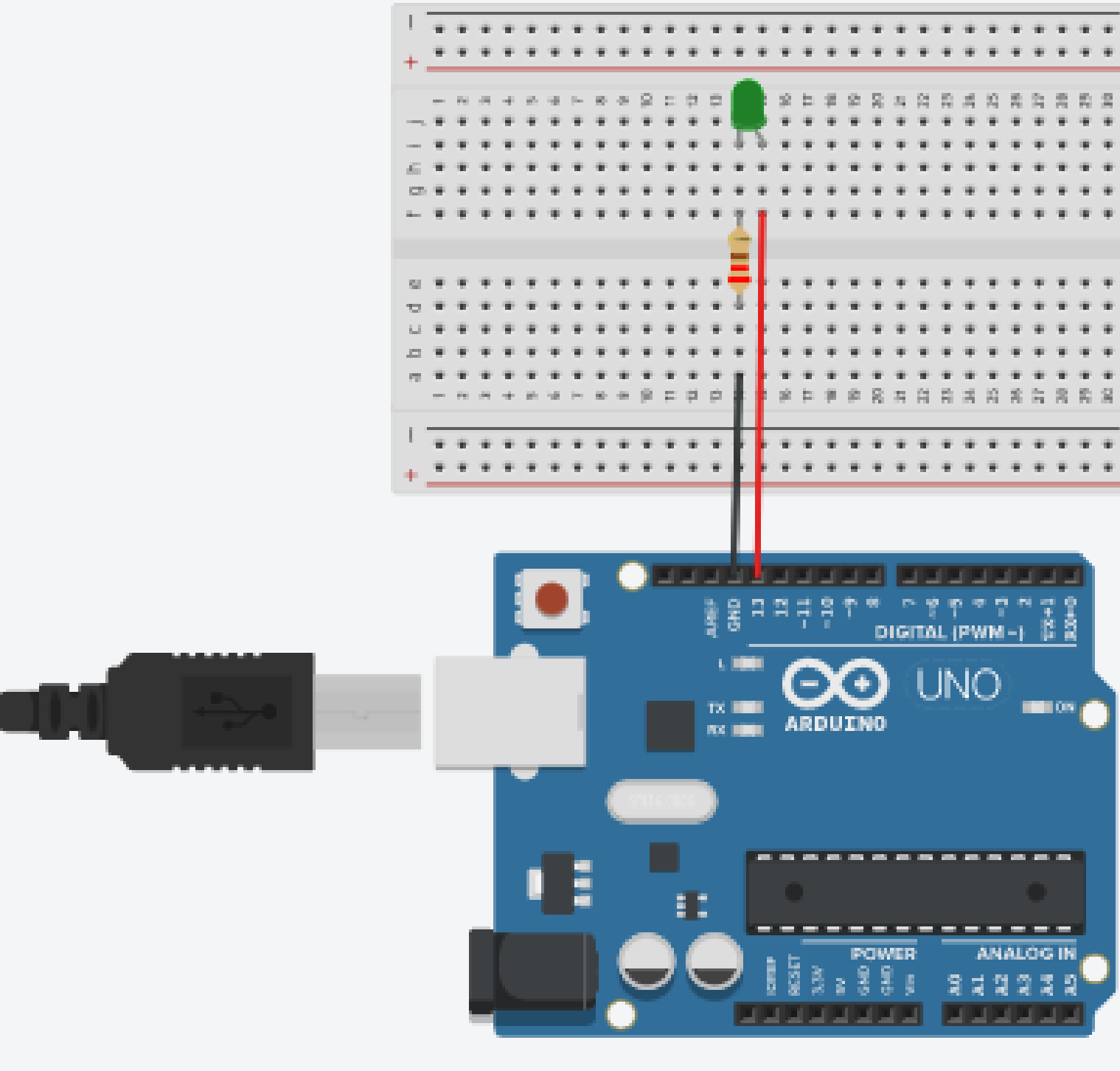
Acender um LED - Configurações Iniciais

- Vamos acender um LED utilizando um caracter específico recebido pela USART.

```
45  int main(){
46      uint8_t dado_rx; //Variável para armazernar dado recebido.
47      DDRB = 0x20; // Pinos PB configurados como entrada, exceto PB5
48      PORTB = 0x20; //PB5 em alto
49      UART_Init();
50      uartString("Digite L ou D.\r \n");
```

Acender um LED

```
51      while(1){
52          if(uartRxOk()){ // Verifica se existe novo dado
53              dado_rx = uartRX(); //Armazena dado
54              switch(dado_rx){//Testa o valor Lido
55                  case 'L':
56                  case 'l':
57                  uartString("Ligar LED. \r\n");
58                  PORTB |= 1 << PB5; //Liga LED
59                  break;
60                  case 'D':
61                  case 'd':
62                  uartString("Desliga LED.\r\n");
63                  PORTB &= ~(1 << PB5); //Desliga LED
64                  break;
65              }
66          }
```


DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN

Interrupção

- Para habilitar e desabilitar as interrupções, vamos modificar o valor do bit RXCIE0 e TXCIE0 no registrador UCSR0B.

Interrupção

```
43  /*Habilita ou desabilita a interrupção de recepção da USART
44  x = 0, desabilita, qualquer outro valor, habilita a interrupção.
45  */
46  void uartIntRx(uint8_t _hab){
47      if(_hab){
48          UCSR0B |= (1 << RXCIE0); //Habilita a interrupção de recep.
49      }else{
50          UCSR0B &=~(1 << RXCIE0); //Desabilita a interrupção de recep.
51      }
52  }
53  /*Habilita ou desabilita a interrupção de transmissão da USART
54  x = 0, desabilita, qualquer outro valor, habilita a interrupção.
55  */
56  void uartIntTx(uint8_t _hab){
57      if(_hab){
58          UCSR0B |= (1 << TXCIE0); //Habilita a interrupção de trans.
59          }else{
60          UCSR0B &=~(1 << TXCIE0); //Desabilita a interrupção de trans.
61      }
62  }
```

Interrupção

```
64  ISR(USART_RX_vect){
65      uint8_t dado_rx; //Variável para armazenar dado recebido;
66      dado_rx = uartRX(); //Armazena o dado;
67      uartString("Você digitou: ");
68      while(!uartTxOk());//Aguarda o último dado ser enviado;
69      uartTxOk(dado_rx); //Envia o caracter recebido
70      uartString("\r\n"); //Nova linha
71  }
72  int main(){
73      UART_Init(); //Inicialização do USART
74      uartString("Digite L ou D.\r \n"); //
75      uartIntRx(1);//Habilita a interrupção de recep.
76      sei(); //Habilita a interrupção geral.
77      while(1);
78  }
79
```