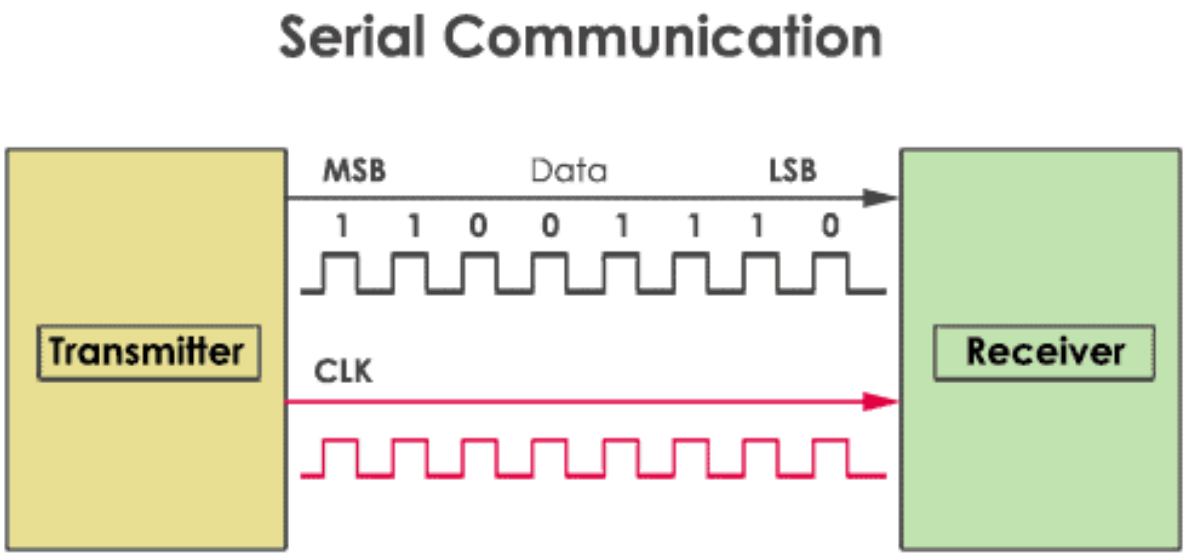
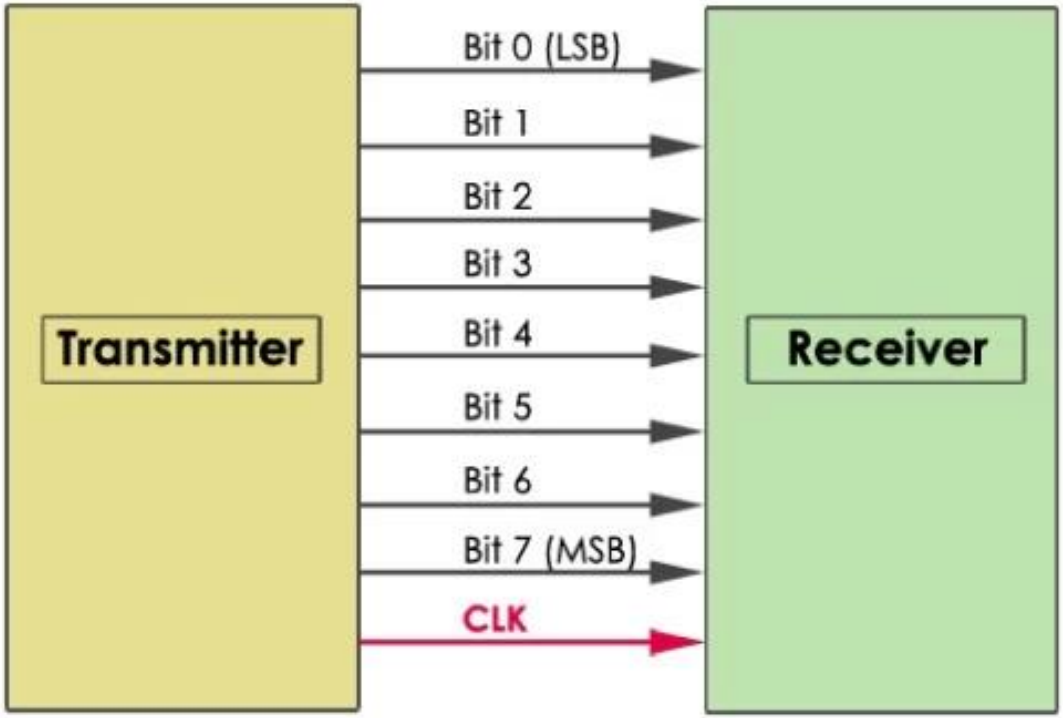
COMUNICAÇÃO SERIAL

A comunicação serial é uma das formas mais comuns de transferência de dados entre dispositivos eletrônicos;

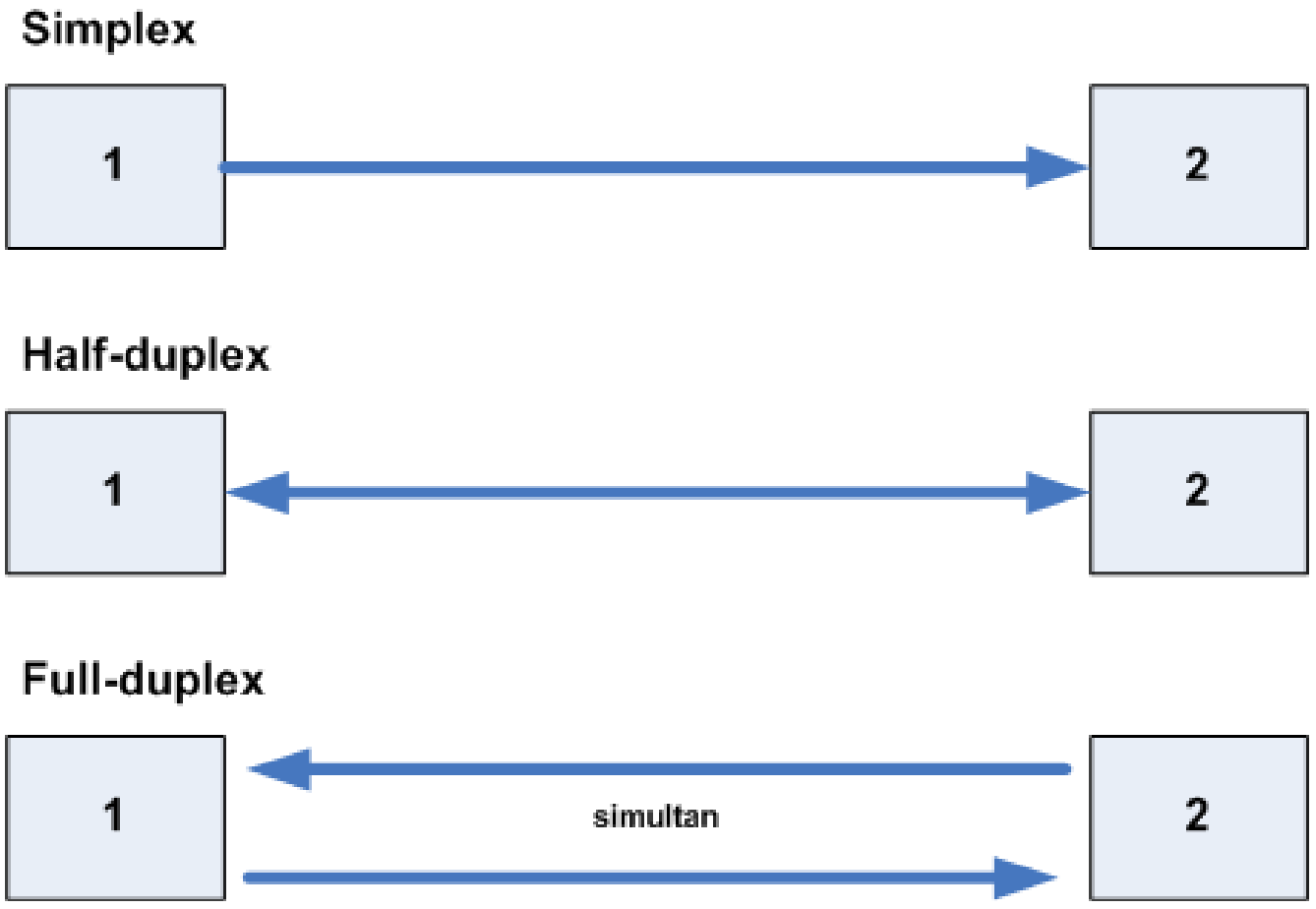
É mais eficiente para dispositivos distantes entre si, pois demandam menos “fios” para fazer a comunicação:



Outra forma é através da Comunicação Paralela:
Nesta, há a necessidade de um “fio” para a transmissão de cada bit.
Por exemplo, em um sistema de 8 bits, precisa de pelo menos 9 fios;

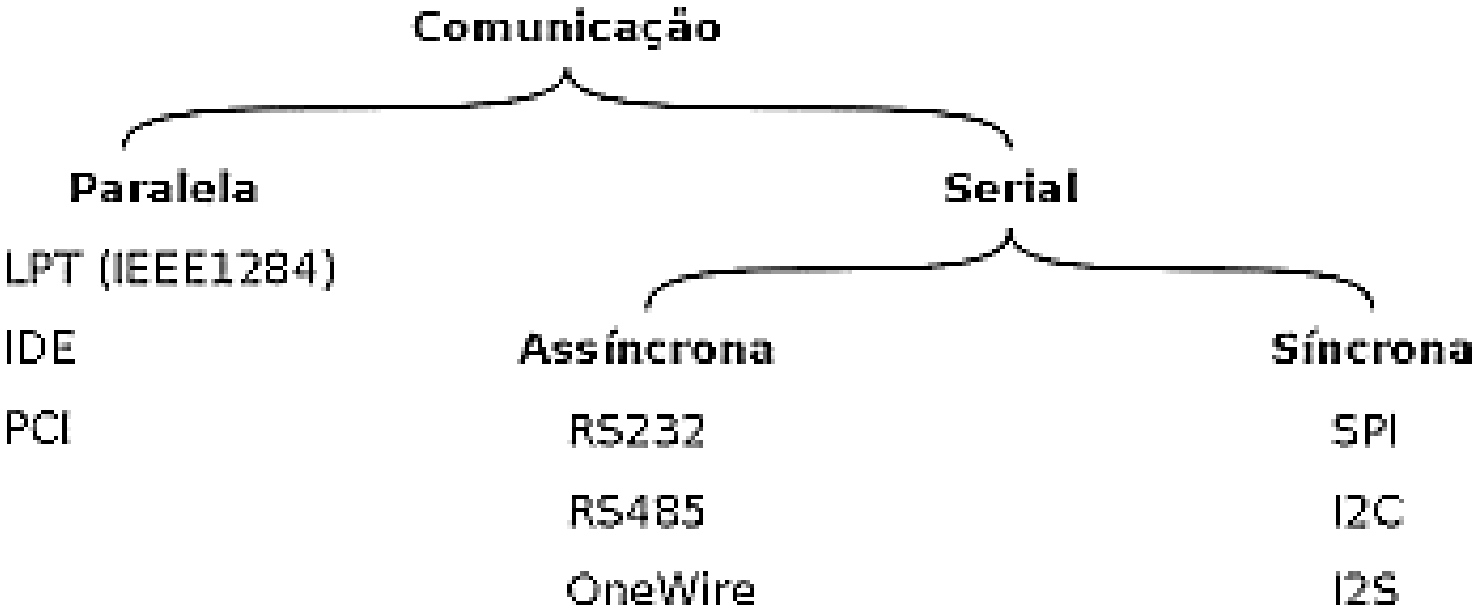


Modos de Transmissão:



Comunicação Serial

Existem diversas formas de comunicação serial, por exemplo: UART (RS232), USB, RS485, SPI, I2C, ONEWIRE, etc.

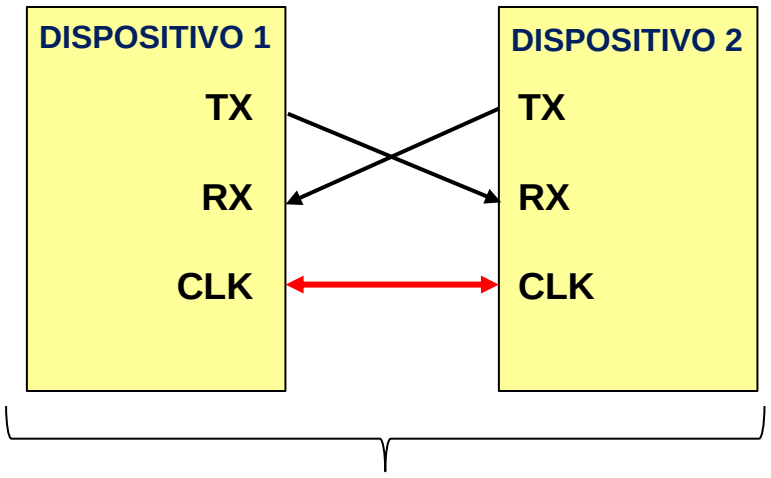


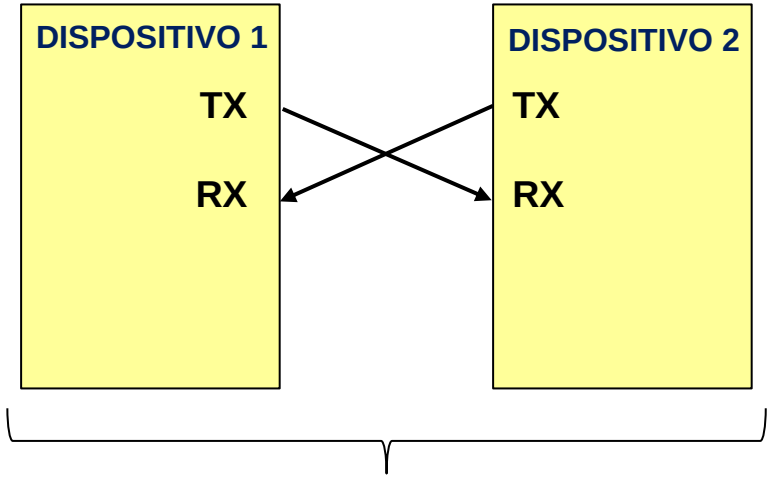
Comunicação Serial

A comunicação serial pode ser assíncrona ou síncrona:

- Assíncrona: não existe um sinal de *clock* para sincronizar a transmissão e a recepção;

- Síncrona: precisa de um sinal de *clock* para sincronizar a transmissão e a recepção;





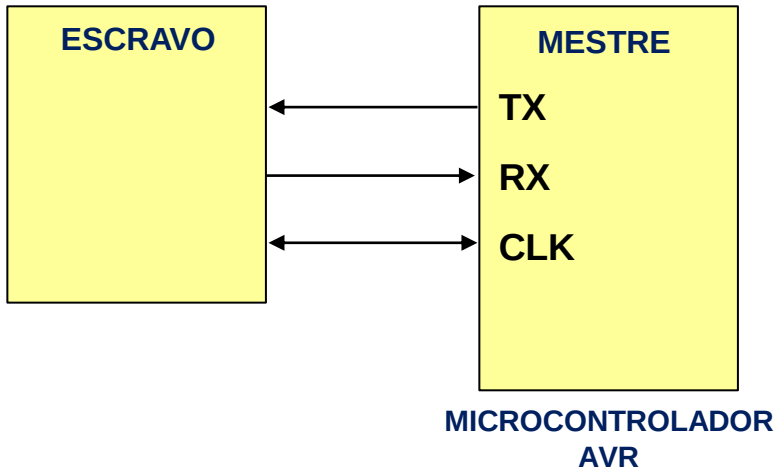
USART - Universal Synchronous Asynchronous Receiver Transceiver

A **USART** significa transmissor e receptor síncrono e assíncrono universal.

É uma comunicação serial de dois protocolos.

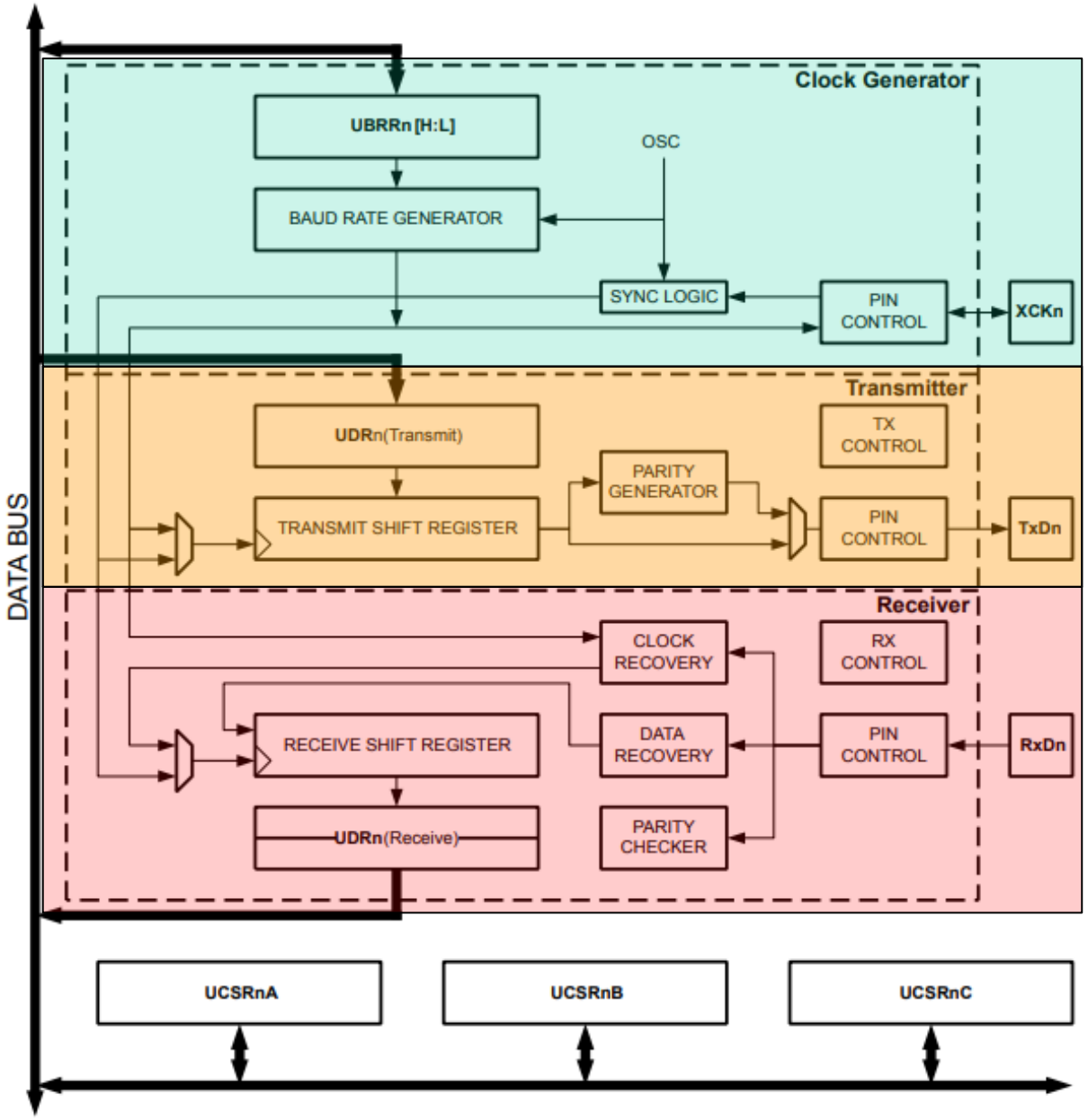
Este protocolo é usado para transmitir e receber os dados bit a bit em relação aos pulsos de *clock* em um único fio.

Os microcontroladores AVR tem dois pinos: TX e RX, que são especialmente usados para transmitir e receber dados em série.



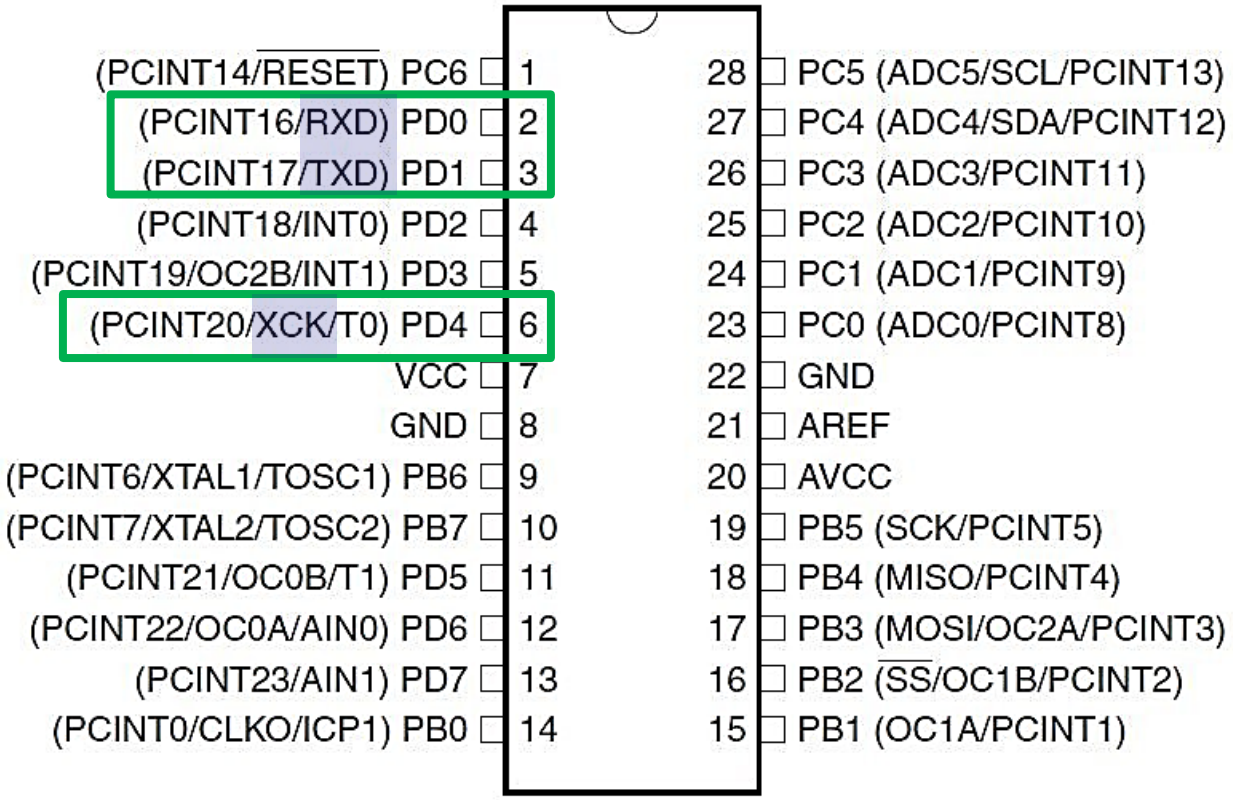
Principais características da USART AVR

- Operação Full Duplex (Registros Seriais de Recepção e Transmissão Independentes)
- Operação Assíncrona ou Síncrona
- Operação Síncrona com clock Mestre ou Escravo
- Gerador de Baud Rate de Alta Resolução
- Suporta Quadros Seriais com 5, 6, 7, 8 ou 9 bits de dados e 1 ou 2 bits de parada
- Geração de paridade ímpar ou par e verificação de paridade suportada pelo hardware
- Detecção de transbordamento de dados
- Detecção de erro de enquadramento
- Filtragem de ruído Inclui detecção de bit de início falso e filtro passa-baixo digital
- Três interrupções separadas em TX completo, Registro de dados TX vazio e RX completo
- Modo de comunicação multi-processador
- Modo de comunicação assíncrona de velocidade dupla


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

Configuração de pinos USART

- RXD: pino do receptor USART (pino 2)
- TXD: pino do transmissor USART (pino 3)
- XCK: pino de relógio USART (pino 6)



A USART dos microcontroladores AVR operam em três modos.

Modo Normal Assíncrono

Neste modo de comunicação, os dados são transmitidos e recebidos bit a bit, sem pulsos de *clock*, pela taxa de transmissão predefinida definida pelo registro **UBBR**.

Modo de velocidade dupla assíncrona

Neste modo de comunicação, os dados transferidos com o dobro da taxa de transmissão são configurados pelo registro **UBBR** e configurados pelos bits U2X no registro **UCSRA**.
É usado onde configurações precisas de taxa de transmissão e relógio do sistema são necessárias.

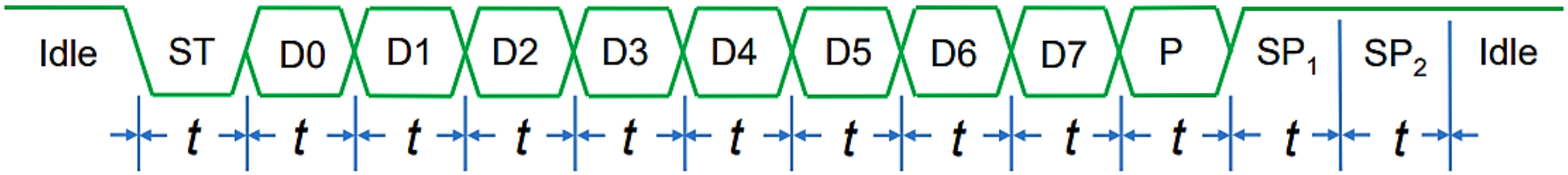
Modo Síncrono

Neste sistema, a transmissão e recepção de dados em relação ao pulso do *clock* é definida como UMSEL = 1 no registro **UCSRC**.

Modo Assíncrono (UART):

Um frame UART consiste de:

- 1 start-bit;
- 7 ou 8 bits de dados;
- 0 ou 1 bit de paridade (pode ser paridade par ou paridade ímpar);
- 1 ou 2 stop-bits;
- Quando o canal está parado (sem transmissão) a linha fica em nível lógico 1;



Na prática, normalmente se usa 1 start-bit, 7 ou 8 bits de dados e 1 stop-bit;

BAUD RATE: é a taxa de transmissão.

Define a quantidade de bits transmitidos por segundo (um frame UART tem entre 10 e 12 bits);

• Alguns valores padrões de *baud rate* são: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 56000 e 115200;

UDR0 - USART I/O Data Register 0

É o registrador de transmissão e recepção de dados.
Tem o papel de armazenar o conteúdo a ser transmitido ou recebido, dependendo da ação a ser realizada.
Compartilham o mesmo endereço pois não há como transmitir e receber ao mesmo tempo.

Bit	7	6	5	4	3	2	1	0
	TXB / RXB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TXB / RXB[7:0]: USART Transmit / Receive Data Buffer

UCSR0A - USART Controle and Status Register A

É um registrador de configuração e de indicação do estado da comunicação.

Bit	7	6	5	4	3	2	1	0
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit 7 – RXC0: USART Receive Complete
Bit 6 – TXC0: USART Transmit Complete
Bit 5 – UDRE0: USART Data Register Empty
Bit 4 – FE0: Frame Error
Bit 3 – DOR0: Data OverRun
Um *Data OverRun* ocorre quando o buffer de recepção está cheio (dois caracteres). Neste caso há um caractere aguardando no Registrador de Deslocamento e um novo bit de início é detectado.
Bit 2 – UPE0: USART Parity Error
Bit 1 – U2X0: Double the USART Transmission Speed
Bit 0 – MPCM0: Multi-processor Communication Mode

UCSR0B - USART Control and Status Register 0 B

Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – RXCIE0: RX Complete Interrupt Enable 0
Bit 6 – TXCIE0: TX Complete Interrupt Enable 0
Bit 5 – UDRIE0: USART Data Register Empty Interrupt Enable 0
Bit 4 – RXEN0: Receiver Enable 0
Bit 3 – TXEN0: Transmitter Enable 0
Bit 2 – UCSZ02: Character Size 0
Bit 1 – RXB80: Receive Data Bit 8 0
Bit 0 – TXB80: Transmit Data Bit 8 0

UCSR0C - USART Control and Status Register 0 C

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

UMSEL0n: USART Mode Select [n = 1:0]

UMSEL0[1:0]	Mode
00	Asynchronous USART
01	Synchronous USART
10	Reserved
11	Master SPI (MSPIM)(1)

UCSR0C - USART Control and Status Register 0 C

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

UPM0n: USART Parity Mode 0 n [n = 1:0]

UPM0[1:0]	ParityMode
00	Disabled
01	Reserved
10	Enabled, Even Parity
11	Enabled, Odd Parity

UCSR0C - USART Control and Status Register 0 C

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

USBS0: USART Stop Bit Select 0

USBS0	Stop Bit(s)
0	1-bit
1	2-bit

UCSR0C - USART Control and Status Register 0 C

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

UCSZ01 / UDORD0: USART Character Size / Data Order

UCSZ0[2:0]	Character Size
000	5-bit
001	6-bit
010	7-bit
011	8-bit
100	Reserved
101	Reserved
110	Reserved
111	9-bit

UCSR0C - USART Control and Status Register 0 C

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

Bit 0 – UCPOL0: Clock Polarity 0

UCPOL0	Transmitted Data Changed (Output of TxD0 Pin)	Received Data Sampled (Input on RxD0 Pin)
0	Rising XCK0 Edge	Falling XCK0 Edge
1	Falling XCK0 Edge	Rising XCK0 Edge

Etapas de configuração da UART

Etapa 1: Cálculo e definição da taxa de transmissão

A taxa de transmissão de USART / UART é definida pelo registrador **UBRR** *(Baud Rate Register)*
Este registro é usado para gerar a transmissão de dados em uma velocidade específica.

O UBRR é um registrador de 16 bits composto de “duas partes” de 8 bits: como UBRR (H), UBRR (L).

Operating Mode	Equation for Calculating Baud Rate(1)	Equation for Calculating UBRRn Value
Asynchronous Normal mode (U2Xn = 0)	$\text{BAUD} = \frac{f_{\text{OSC}}}{16(\mathbf{UBRR}n + 1)}$	$\mathbf{UBRR}n = \frac{f_{\text{OSC}}}{16\text{BAUD}} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$\text{BAUD} = \frac{f_{\text{OSC}}}{8(\mathbf{UBRR}n + 1)}$	$\mathbf{UBRR}n = \frac{f_{\text{OSC}}}{8\text{BAUD}} - 1$
Synchronous Master mode	$\text{BAUD} = \frac{f_{\text{OSC}}}{2(\mathbf{UBRR}n + 1)}$	$\mathbf{UBRR}n = \frac{f_{\text{OSC}}}{2\text{BAUD}} - 1$

A fórmula da taxa de transmissão é:

BAUD = Fosc / (16 * (UBBR + 1))

A fórmula do registro UBRR é: UBRR = Fosc / (16 * (BAUD-1))

A frequência do microcontrolador AVR é 16 MHz = 16000000;
Vamos supor que a taxa de transmissão seja de 19200 bps, então

UBRR = 16000000 / (16 * (19200-1))

UBRR = 16000000 / (16 * (19200-1))

UBRR = 51.099

Etapa 2: Seleção do modo de transmissão de dados

O modo de transmissão de dados, bit de início e bit de parada e o tamanho dos caracteres são definidos pelo registro de controle e status **UCSRC** *(Control and Status Register)*.

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

Bits 7:6 – UMSEL0n: USART Mode Select 0 n [n = 1:0]
These bits select the mode of operation of the USART0

Table 24-8. USART Mode Selection

UMSEL0[1:0]	Mode
00	Asynchronous USART
01	Synchronous USART
10	Reserved
11	Master SPI (MSPIM)(1)

Etapa 3: Seleção do modo de transmissão de dados

O modo síncrono e assíncrono é selecionado pelo bit **UMSEL** do registro de status de controle.

Se UMSEL = 0, então o USART opera em modo assíncrono, caso contrário, opera em modo síncrono.

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

Etapa 4: Bit de início e bit de parada

Os bits de início e de parada são uma forma de enviar e receber dados em série.

Os microcontrolador AVR tem um bit de início e dois bits de parada.

O bit de parada extra pode ser útil para adicionar um pouco mais de tempo de processamento de recepção.

Assim, podemos aumentar o tempo de processamento usando dois bits de parada para obter os dados adequados.

O número de bits de parada é selecionado pelo bit USBS de **UCSRC** (o registro de status de controle).
USBS = 0 => um bit de parada
USBS = 1 => dois bits de parada.

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

Etapa 5: Definição do tamanho do caractere

Podemos escolher um formato de quadro de dados em cada quadro pelo bit UCSZ do registrador **UCSR0C / UCSR0B**.

UCSZ2	UCSZ1	UCSZ0	Charector size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

UCSR0C

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

UCSR0B

Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

Etapa 6: Armazenamento dos Dados Recebidos

O microcontrolador AVR contém um registro de buffer **UDR** *(Data Buffer Registers)* para transmitir e receber dados.

O UDR é um registro de buffer de 16 bits dos quais 8 bits são usados para receber (RXB) os dados e outros bits são usados para transmitir os dados (TXB).

A transmissão do registro do buffer de dados será o destino do registro UDR para os dados nele gravados.

Receber o registro do buffer de dados retornará o conteúdo do registro UDR.

Etapa 7: Habilitação do transmissor e receptor

Os dados são transmitidos e recebidos pelos pinos RXC e TXC do microcontrolador. .
Estes são configurados pelo registro **UCSRA.**

Os flags RXC e TXC indicam que houve a recepção e transmissão completa.

Bit	7	6	5	4	3	2	1	0
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	1	0	0	0	0	0

```
int main ()
{
   int i = 0;
   unsigned int ubrr = 103    //BAUD_RATE = 9600bps;
   unsigned char dados [] = “VAI CURINTIA ! ! !";

   / / Definir taxa de transmissão
   UBRR0H =0;
   UBRR0L = ubrr;
   UCSR0C = 0b000000110;;       // Definir formato de quadro: 8 dados, primeiro bit
   UCSR0B = 0b00001000;     // Habilitar transmissor
   while (1) {
      i = 0;
      while (dados [i]! = 0) // imprime a string "VAI CURINTIA ! ! !"
      {
        while (! (UCSR0A & (1 << UDRE0)));  // Aguarde o buffer de transmissão vazio
        UDR0 = dados [i];  // Coloca os dados no buffer, envia os dados
        i ++;                    // incrementa contador
        }
   }
}
```