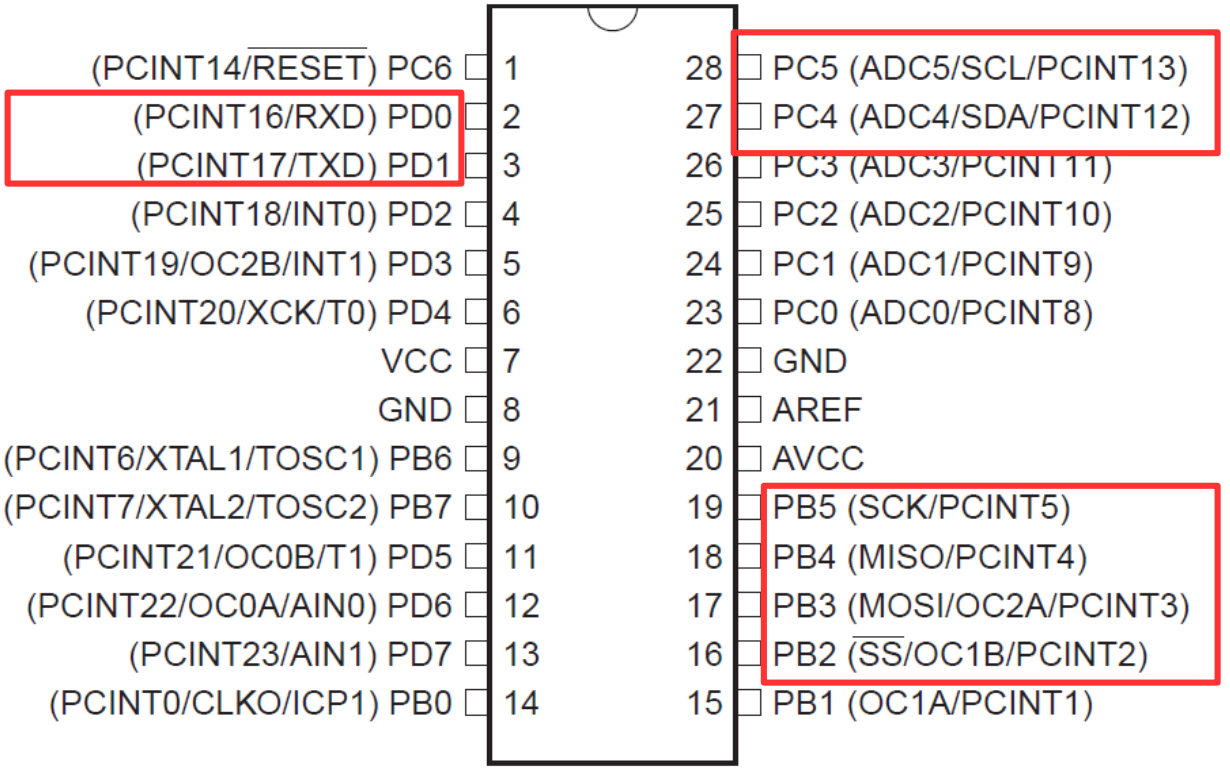
Sistemas embarcados

Comunicações seriais:
UART, SPI, IIC (I²C)

Periférico USART
(Universal Synchronous and Asynchronous serial Receiver and Transmitter)

Sistemas embarcados

UART



IIC

SPI

Nos microcontroladores é comum se ter periféricos para comunicação de dados serial, que pode ser síncrona ou assíncrona

Comunicação serial

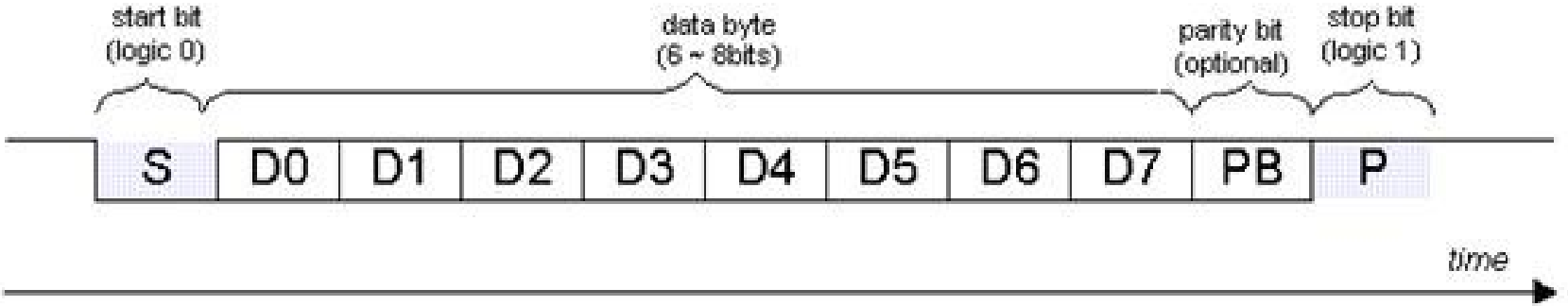
- Comunicações seriais podem ser:
  - Assíncronas (ex.: UART)
  - Síncronas (ex.: SPI e I2C)

A diferença está no uso de um sinal adicional de sincronização (clock)

Comunicação serial

Comunicação de dados seriais

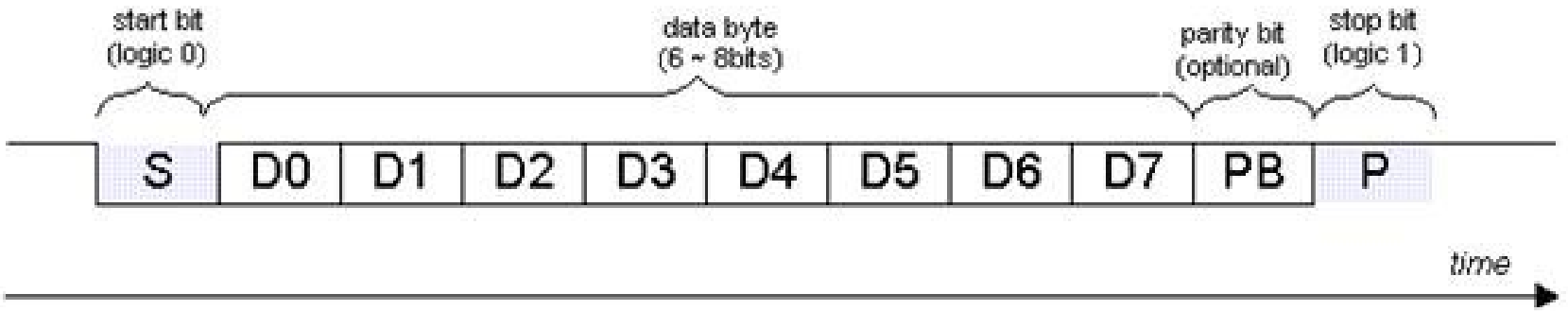
- Dados binários (0s e 1s) são transmitidos e recebidos serialmente (isto é, um de cada vez, seqüencialmente)



- Geralmente, a informação é transmitida/recebida *byte* a *byte*

Comunicação serial UART

- As UARTs se “entendem” por seguir um **protocolo.**



- O **protocolo das UARTs** define que cada mensagem (quadro) inicia um bit de início (*start bit*) e termina com um bit de fim (*stop bit*)

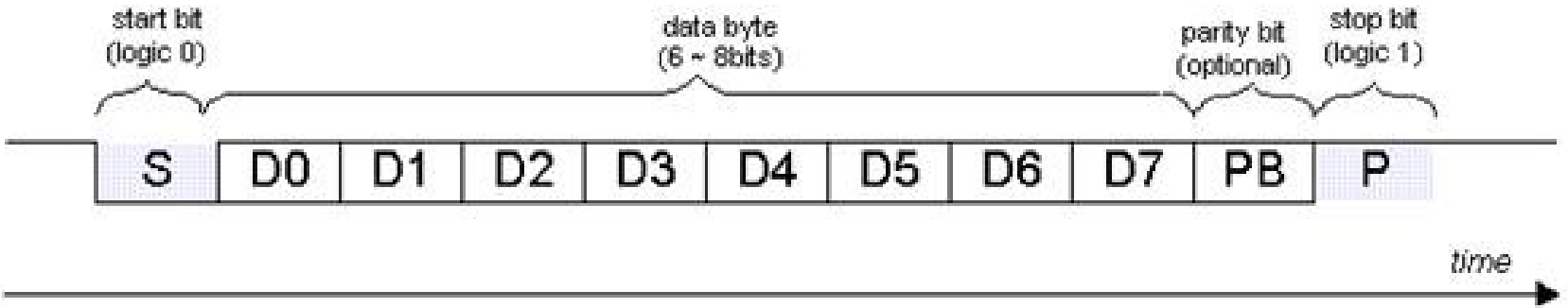
Comunicação serial UART

- O *start bit* tem sempre o valor lógico zero (**0**) e o *stop bit* tem valor lógico um (**1**)
- Após o start bit são transmitidos os *bits de dados* (que formam o *byte*)
- Opcionalmente, pode-se transmitir um bit de verificação (bit de paridade)

Comunicação serial UART

- Na comunicação UART cada bit tem uma duração pré-estabelecida.

- Contando-se o tempo de bit, pode-se determinar o valor de cada bit (0 ou 1)



Comunicação serial UART

- Em sistemas de comunicação de dados, símbolos (*bauds*) representam um ou mais *bits*.
- O tempo de duração do símbolo define a taxa de símbolos (*baudrate*)
- E o tempo de cada *bit* define a taxa de bits (*bitrate)*

Comunicação serial UART

- Símbolo (*baud*) é o sinal físico (p. ex.: sinal elétrico com tensão igual a 5V)

- Bit é o dado lógico (p.ex.: símbolo de 5V de tensão representa o bit lógico **1**)

- Nas UARTs cada **10 (ou 11)** símbolos comunicam **8** bits (geralmente, um byte)

Comunicação serial UART

- O tempo de símbolo (*baudrate*) é controlado pelo periférico de comunicação UART do microcontrolador.

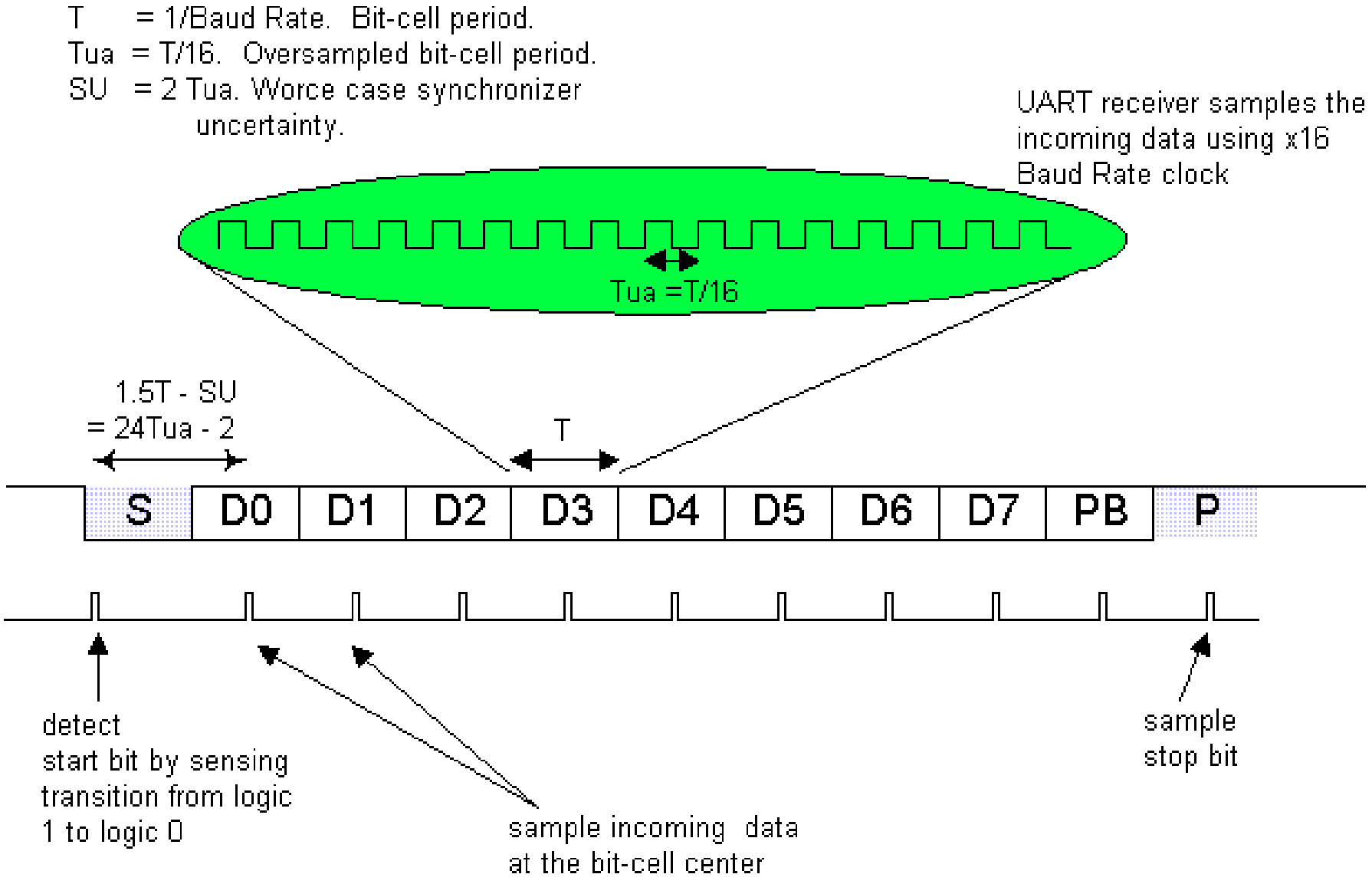
Baud rate
50
300
600
2400
4800
9600
19,200
38,400
57,600
115,200
230,400

A UART possui um gerador de *baud rate* flexível de 12 bits, que suporta *baudrates* de até 115,2 kbps.

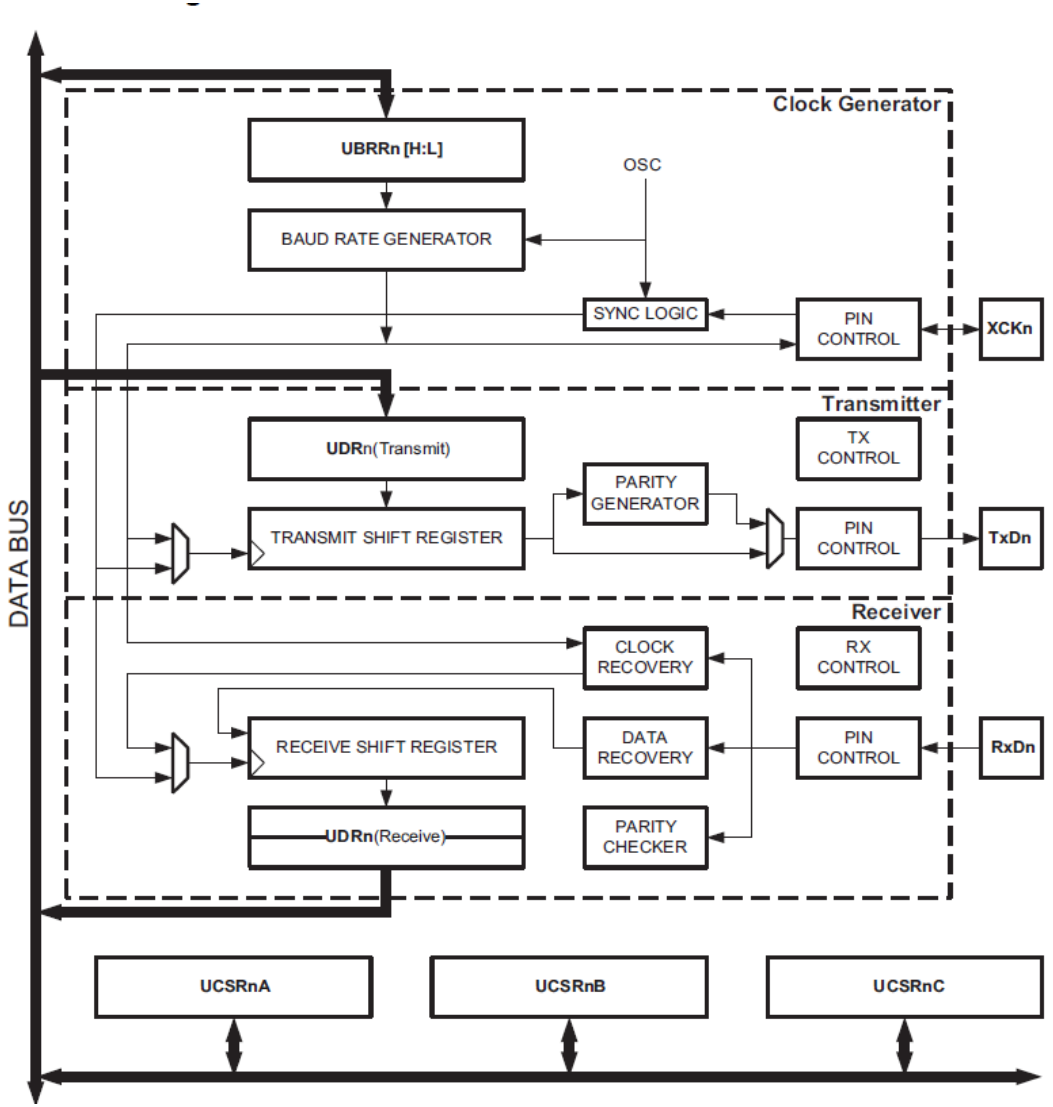
A transmissão (Tx) e a recepção (Rx) dentro de um mesmo módulo UART utilizam o **mesmo** *baudrate.*

Cada módulo UART possui um gerador de *baudrate* separado.

Comunicação de dados seriais



Sistemas embarcados



Programando a UART

- As comunicações seriais são utilizadas com uma determinada taxa de dados por segundo (***baud rate***).

- No Atmega328, isto é feito programando-se o registrador UBRRn

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRRn Value
Asynchronous Normal mode (U2Xn = 0) Modo assíncrono normal	$BAUD = \frac{f_{OSC}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$

Programando a UART

- No Atmega328, isto é feito programando-se o registrador UBRRn

Asynchronous Double Speed mode (U2Xn = 1) Modo assíncrono dobrado	$BAUD = \frac{f_{OSC}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master mode Modo síncrono mestre	$BAUD = \frac{f_{OSC}}{2(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

Programando a UART

- Os módulos UART são operados através dos seguintes registradores.
  - 2 reg. para configurar *baudrate* (UBRRnH, UBRRnL)
  - 3 reg. de controle e estados (ex.: interrupções, erros, modos de operação...) (UCSRnA, UCSRnB e UCSRnC)
  - 1 reg. de dados (UDRn) (escrita p/ TX e leitura p/ RX)

Inicialização da UART

- Primeiro passo: calcular e escrever o valor dos registradores de taxa de baud (baud rate - UBRR). Ex.: UART0 a 9600bps, clock de 1MHz, UBRR=??.

Inicialização da UART

- Primeiro passo: calcular e escrever o valor dos registradores de taxa de baud (baud rate - UBRR). Ex.: UART0 a 9600bps, clock de 1MHz, UBRR=??.

$$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$$

UBRR = 1000000/16/9600 – 1 = 6

UBRR0H = 0x00

UBRR0L = 0x06

Inicialização da UART

Baud Rate [bps]	f_{osc} = 1.0000MHz			
	U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%
4800	12	0.2%	25	0.2%
9600	6	-7.0%	12	0.2%
14.4K	3	8.5%	8	-3.5%
19.2K	2	8.5%	6	-7.0%
28.8K	1	8.5%	3	8.5%
38.4K	1	-18.6%	2	8.5%
57.6K	0	8.5%	1	8.5%
76.8K	–	–	1	-18.6%
115.2K	–	–	0	8.5%

Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Inicialização da UART

Segundo passo: **configurar modo de funcionamento** (8 ou 9 bits de dado, paridade, habilitar transmissor (Tx) e receptor (Rx), habilitar interrupções,...)

Ex.: porta UART 0, dado de 8 bits, sem paridade, Tx e Rx habilitados, interrupções desabilitadas)

Inicialização da UART

7	6	5	4	3	2	1	0	
RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
R	R/W	R	R	R	R	R/W	R/W	

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

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

Table 22-4. UMSELn Bits Settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)(1)

Inicialização da UART

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

Table 22-5. UPMn Bits Settings

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

Table 22-6. USBS Bit Settings

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

Inicialização da UART

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

Table 22-7. UCSZn Bits Settings

UCSZn2	UCSZn1	UCSZn0	Character 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

Inicialização da UART

Ex.: porta UART 0, dado de 8 bits, sem paridade, 1 stop bit, Tx e Rx habilitados, interrupções desabilitadas)

```
#define FOSC 1843200 // Clock Speed
#define BAUD 9600
#define (UBRR FOSC/16/BAUD-1)

void main( void )
{
    UART_Init ( UBRR );
}

void UART_Init( unsigned int ubrr)
{
  /* Set baud rate */
  UBRR0H = (unsigned char)(ubrr>>8);
  UBRR0L = (unsigned char)ubrr;
  /* Enable receiver and transmitter */
  UCSR0B = (1<<RXEN0)|(1<<TXEN0);
  /* Set frame format: 8data, 1stop bit */
  UCSR0C = (3<<UCSZ00);
} // USART_Init
```

Operação da UART

Após a inicialização (que pode ser colocada em uma função. ex.: UART_init(), pode-se escrever as funções para operar a comunicação.

Ex.: void UART_transmite (uint8_t dado_a_enviar)

Ex.: void UART_recebe (uint8_t * dado_a_receber)

Operação da UART

```
void UART_Transmite( uint8_t dado )
{
   /* Espera o fim da transmissão do caractere
anterior para transmitir o próximo */

   while ( !( UCSR0A & (1<<UDRE0)) );

   /* Armazena o caractere a ser transmitido no
registrador de transmissão */

   UDR0 = dado;
}
```

Operação da UART

```
void UART_Recebe( uint8_t *dado )
{
	/*  Espera a recepção do caractere */

	while ( !(UCSR0A & (1<<RXC0)) );

	/* Copia o caractere recebido */

	*dado = UDR0;
}
```

Exercício

1) Escrever funções para inicializar, transmitir e receber dados pela UART.