Sistemas Baseados em Microprocessadores

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores
Faculdade de Engenharia

ATMega328P – Comunicações

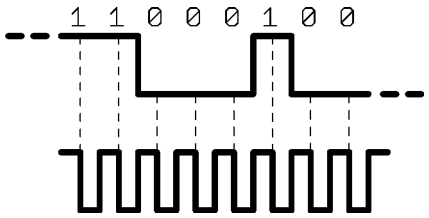
João Paulo de Sousa

Comunicações – alguns conceitos gerais

- **Paralelo** - vários bits em simultâneo
  Ex: ligação direta a um display de 7 segmentos
- **Série** - um bit de cada vez
  - Síncronas (com sinal de relógio explícito)
    Exemplos: comunicação uC ↔ Periféricos
    - SPI – 3 sinais: **MOSI**, **MISO**, **CLK**, Gnd
    - I2C – 2 sinais: **SDA**, **SCL**, Gnd
  - Assíncronas (sem sinal de relógio explícito)
    Exemplo: comunicação uC ↔ PC
    - Sinais: **TxD**, **RxD**, Gnd

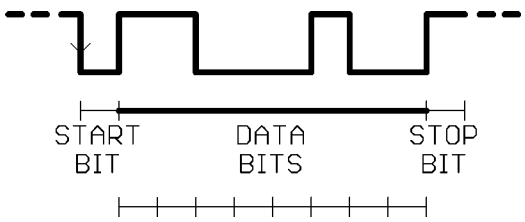
Comunicação série – conceitos gerais

- Síncrona:



- Sinais:
  - Dados
  - Relógio
  - Outros: CS
- Sincronismo Rx-Tx ?
  - Assegurado pelo CLK

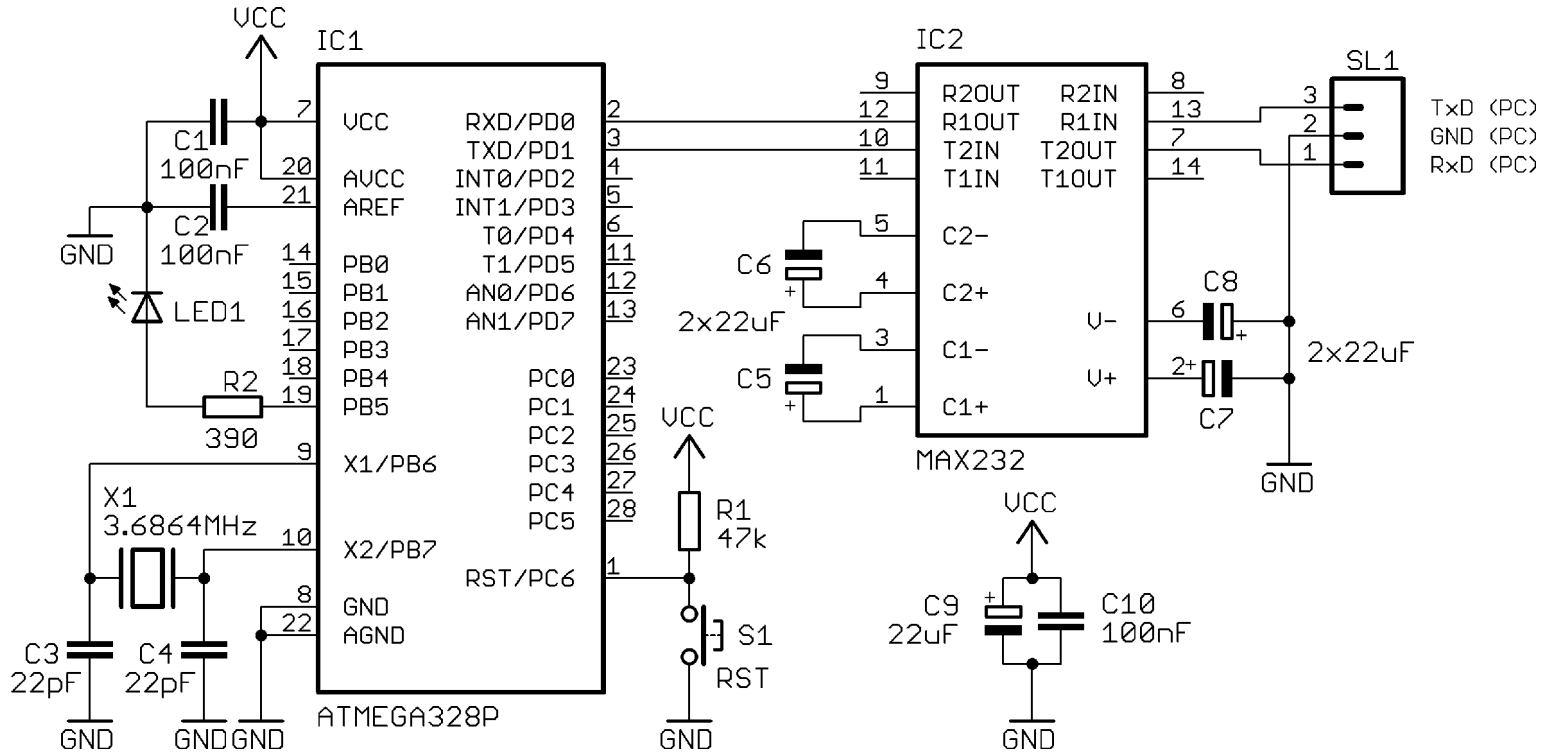
- Assíncrona:



- Bits:
  - Stop bits: 1, 2
  - Data bits: 5...8
  - Outros...
- Sincronismo Rx-Tx ?

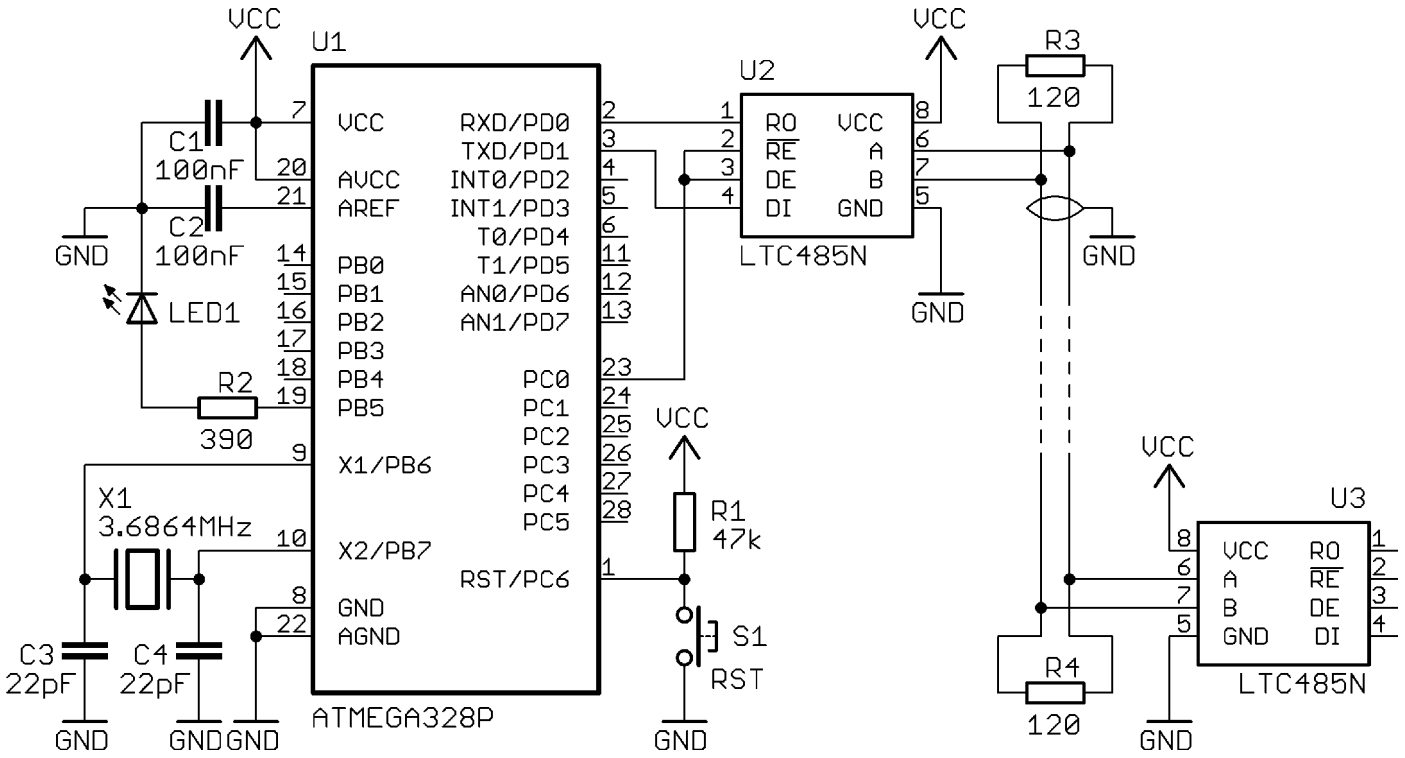
Comunicação série – TIA232

- Unipolar, médias distâncias (<15m)
- 0: 3..15V, 1: -15..-3V



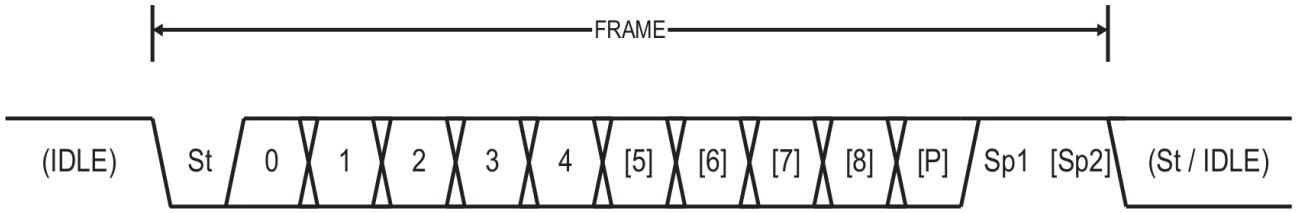
Comunicação série – TIA485

- Diferencial, Grandes distâncias (>1000m)
- 0: dV>0,2V, 1: dV < -0,2V



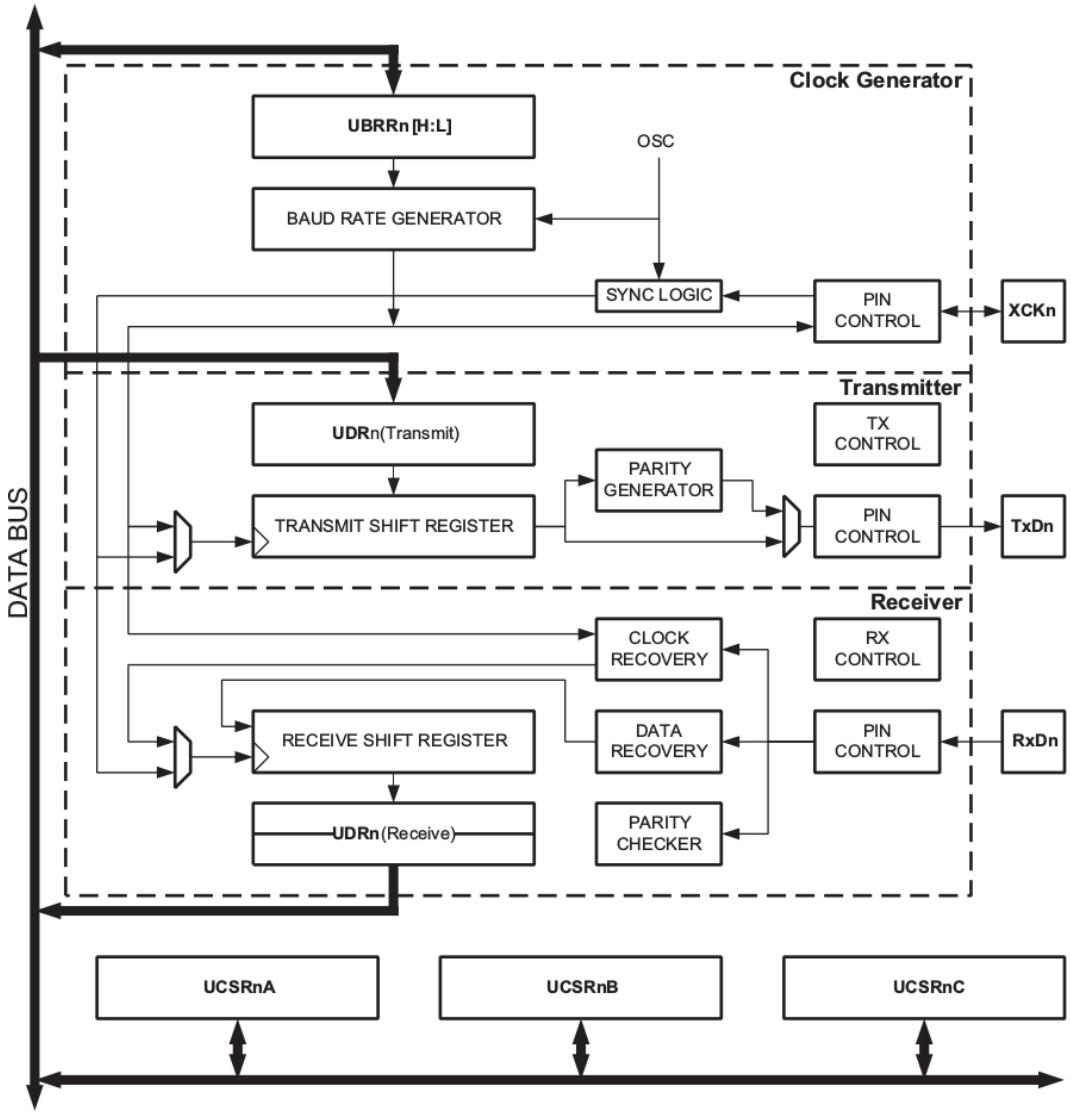
Comunicação série no ATMega328P

- **USART**: Universal Synchronous Assynchronous Receiver Transmitter (assíncronas e síncronas SPI)
- Formato geral de uma trama de comunicação assíncrona:



  - **1** start bit
  - **5** a **9** data bits e **1** bit de paridade opcional
  - **1** ou **2** stop bits
  - Cadência (baud rate) programável

USART – Diagrama de blocos



- Registos de configuração:
  - UBRR0 (16 bits)
  - UCSR0A, UCSR0B, UCSR0C
- Registo de operação:
  - UDR0 (Tx e Rx)

USART - Inicialização

- Cadência em modo assíncrono:

$$BAUD = \frac{F_{CPU}}{K \times (UBRR0+1)}; \quad UBRR0 = \frac{F_{CPU}}{K \times BAUD} - 1$$

  - K=16 (modo normal); K=8 (modo double speed)
  - Pré-divisor do CLK: $F_{CPU} = F_{OSC}/CP$
- Cadências normalizadas (bps):

  …, 1200, 9600, 19200, 38400, 57600, 115200
- Cristais para as conseguir obter com UBRR0 inteiro:

  F_{OSC}=1.8432MHz, 3.6864MHz, 11.0592MHz, etc.

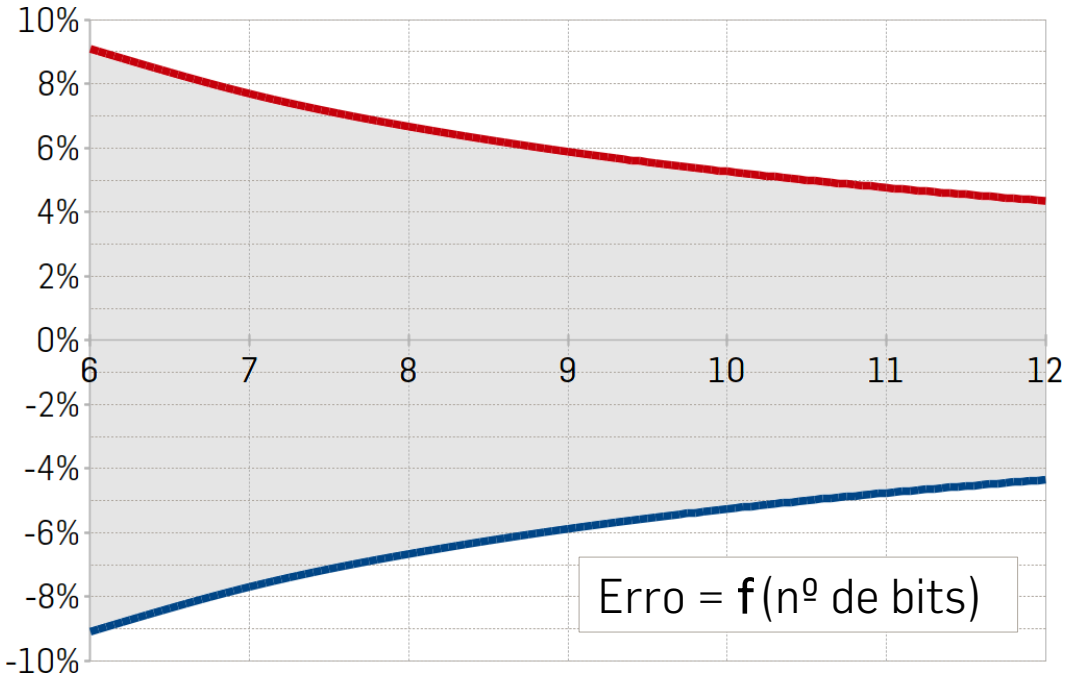
USART - Inicialização

- Cadências pré-definidas e valores de UBRR0

CP: 1	1200		2400		4800		9600		19200		38400		57600		115200		Bps
F_{osc} (MHz)	8	16	8	16	8	16	8	16	8	16	8	16	8	16	8	16	K
1,8432	191	95	95	47	47	23	23	11	11	5	5	2	3	1	1	0	UBRR0
3,6864	383	191	191	95	95	47	47	23	23	11	11	5	7	3	3	1	
7,3728	767	383	383	191	191	95	95	47	47	23	23	11	15	7	7	3	
11,0592	1151	575	575	287	287	143	143	71	71	35	35	17	23	11	11	5	
14,7456	1535	767	767	383	383	191	191	95	95	47	47	23	31	15	15	7	
16,0000	1666	832	832	416	416	207	207	103	103	51	51	25	34	16	16	8	
Bps	1199,8	1200,5	2401,0	2398,1	4796,2	4807,7	9615,4	9615,4	19230,8	19230,8	38461,5	38461,5	57142,9	58823,5	117647,1	111111,1	Bps
Err	-0,02%	0,04%	0,04%	-0,08%	-0,08%	0,16%	0,16%	0,16%	0,16%	0,16%	0,16%	0,16%	-0,80%	2,08%	2,08%	-3,68%	Err

USART - Inicialização

- Quanto maior a trama menor o erro admissível



USART - Registos de configuração e estado

- Registos UCSR0A, UCSR0B e UCSR0C
  - Bits de estado: a branco
  - Bits de configuração: a outras cores
  - Pag. 192 a 196 da folha de características

UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0

USART - Registos de configuração

- Bits de configuração:
  - UMSEL01, UMSEL00: modo de operação:
    - 0,0=Assíncrono; 0,1=Síncrono; 1,0=Reservado; 1,1=SPI
  - UCSZ02 a UCSZ00: Nº de bits de dados
    - 000=5 bits; 001=6 bits; 010=7 bits; 011=8 bits; 111=9 bits
    - Combinações 100; 101 e 110 reservadas
  - UPM01, UPM00: paridade:
    - 00=Sem paridade; 01=Reservado; 10=Par; 11=Ímpar
  - USBS0: Nº de stop bits:
    - 0=Um stop bit; 1=Dois stop bits

USART - Registos de configuração

- Bits de configuração:
  - **RXEN0**, **TXEN0**: enable do recetor e do emissor
  - **U2X0**: modo double speed
  - **MPCM0**: modo multiprocessador
  - **TXB80**: Escrita do 9º bit de dados (multiprocessador)
- Bits de estado:
  - **RXC0**, **TXC0** e **UDRE0**: recetor, emissor ou registo de dados vazios...
  - **FE0**, **DOR0**, **PE0**: erros de formato, leitura e paridade
  - **RXB80**: Leitura do 9º bit de dados (multiprocessador)

USART – Exemplo I

- Enviar para o PC:

  ```
  Hello,
  my name is mega. ATmega.
  ```

- De seguida, na linha de baixo, à cadência de 5/s:

  ```
  0 1 2 3 4 5 6 7 8 9 0 1 2 ...
  ```

  alternando o estado de um LED após cada escrita

Exemplo I – Software

```
#include <avr/io.h>
#include <util/delay.h>

#ifndef F_CPU
#define F_CPU 16000000ul
#endif

#define BAUD 57600
#define UBBR_VAL ((F_CPU/(BAUD<<3))-1)

#define LED PB5
```

```
void init_usart(void) {

    // Definir baudrate
    UBRR0H = (uint8_t)(UBBR_VAL>>8);
    UBRR0L = (uint8_t) UBBR_VAL;
    UCSR0A = (1<<U2X0);  // Double speed

    // Definir formato da trama
    UCSR0C = (3<<UCSZ00) // 8 data bits
           | (0<<UPM00)  // no parity
           | (0<<USBS0); // 1 stop bit

    // Ativar apenas o emissor
    UCSR0B = (1<<TXEN0);
}
```

Exemplo I – Software

```
void myputchar(char c) {
/*
 * Para enviar um byte basta escreve-lo
 * no registo UDRØ, verificando antes se
 * este está disponível (bit UDREØ de UCSRØA)
 *
 */
    // Espera que UDRØ esteja vazio
    while((UCSRØA & (1<<UDREØ)) == Ø);
    UDRØ = c; // Envia para a porta serie
}

void myprint(char *p) {
/*
 * Envia a string apontada por p
 * e terminada por zero
 */
    char c;
    while ((c = *p++) != Ø)
        myputchar(c);
}
```

```
int main (void) {
    unsigned char i = Ø;

    init_usart();


    myprint("\r\n\nHello,\r\n");
    myprint("my name is mega. ATmega.\r\n");

    while(1) {
        myputchar(i+'Ø'); myputchar(' ');
        i = (i+1) % 1Ø;
        PORTB = PORTB ^ (1<<LED);
        _delay_ms(2ØØ);
    }
}
```

426 bytes

Exemplo I – Software (variante com printf)

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>

#ifndef F_CPU
#define F_CPU 16000000ul
#endif

#define BAUD 57600
#define UBBR_VAL ((F_CPU/(BAUD<<3))-1)
#define LED PB5

static int put_char(char c, FILE *stream);
static FILE mystdout = FDEV_SETUP_STREAM(
       put_char, NULL, _FDEV_SETUP_WRITE);

int put_char(char c, FILE *stream) {
    while((UCSR0A & (1<<UDRE0)) == 0);
    UDR0 = c;
    return 0;
}
```

```
int main (void) {
    unsigned char i = 0;

    init_usart();
    stdout = &mystdout;    //output stream

    printf("\r\n\nHello,\r\n");
    printf("my name is mega. Atmega.\r\n");

    while(1) {
        printf("%d, ",i);
        i = (i+1) % 10;
        PORTB = PORTB ^ (1<<LED);
        _delay_ms(200);
    }
}
```

1926 bytes (mais 352%)

USART – Interrupções

- Bits de configuração
  - RXCIE0, TXCIE0, UDRIE0: habilitação do atendimento

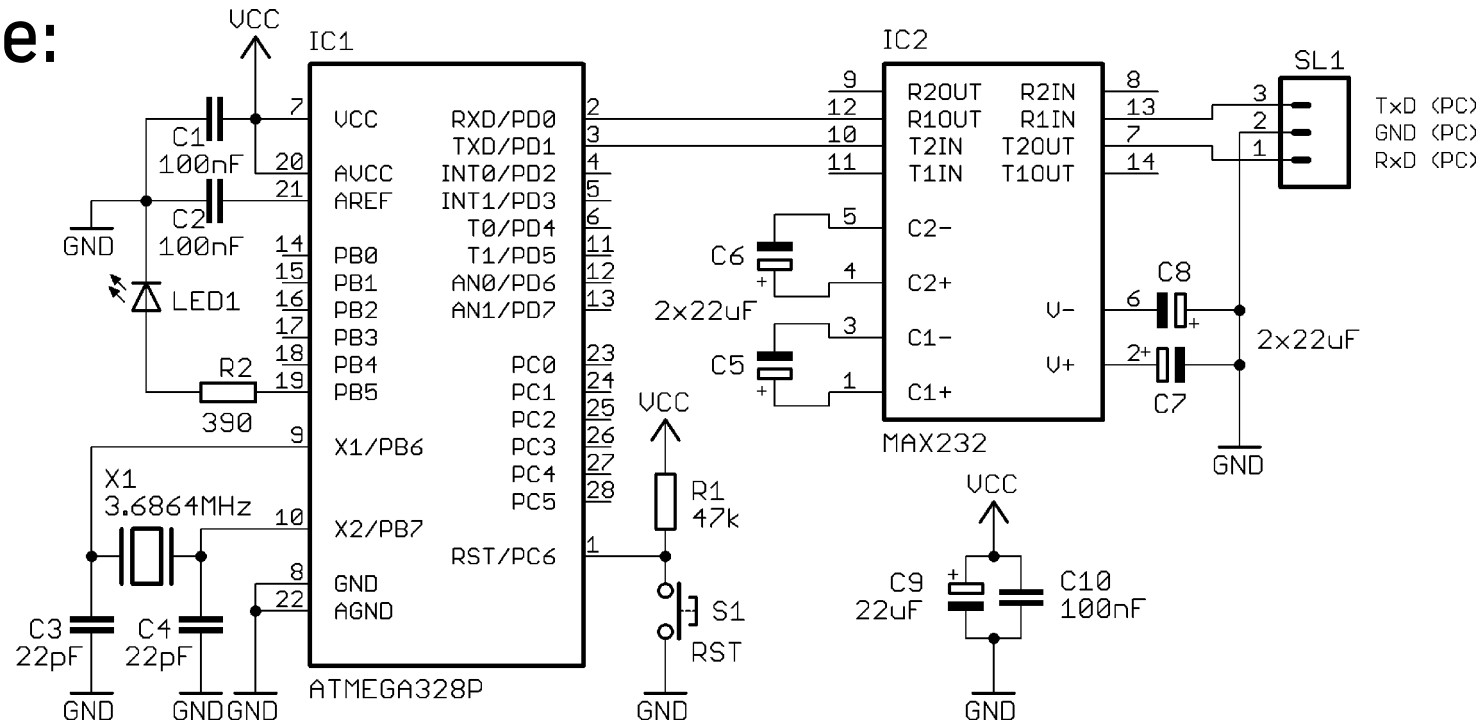
- Bits de estado e vetores:
  - RXC0: USART Receive Complete (USART_RX_vect)
  - TXC0: USART Transmit Complete (USART_TX_vect)
  - UDRE0: USART Data Register Empty (USART_UDRE_vect)

USART – Exemplo II

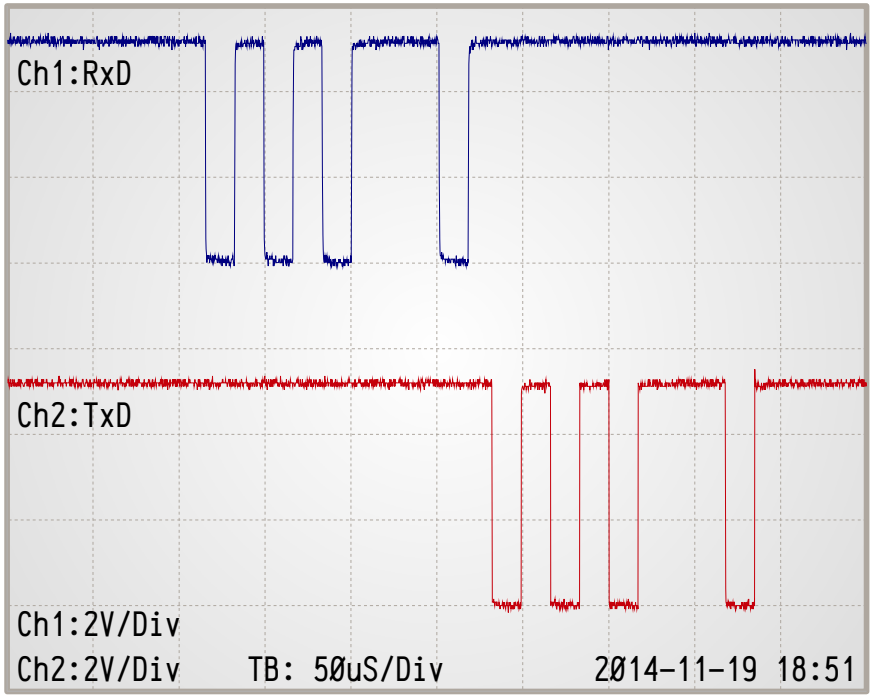
- **Problema:** Devolver pela porta série o que é recebido e complementar um LED sempre que um byte é enviado

  a) Por interrogação b) Por interrupção

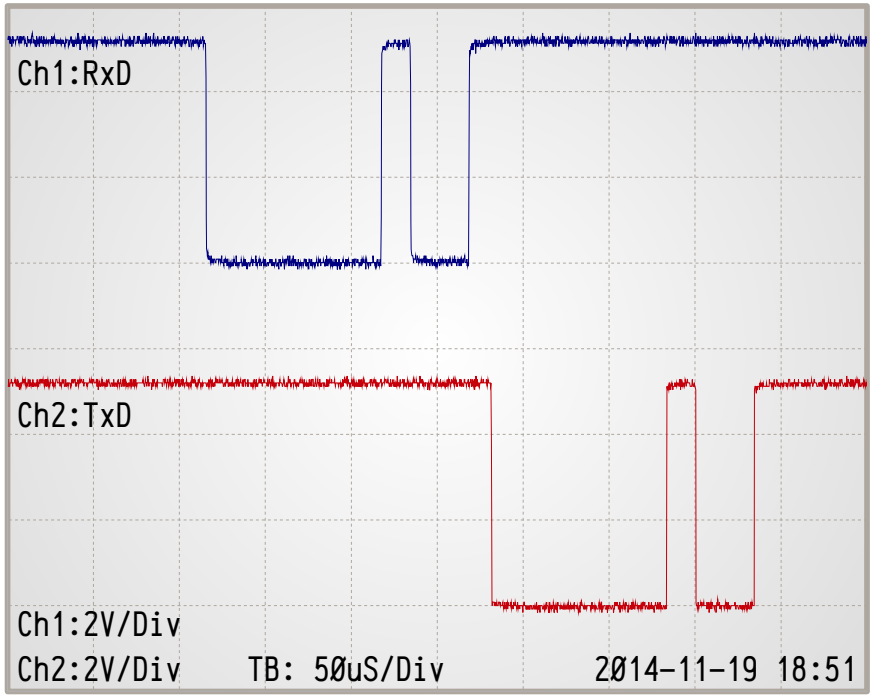
- **Hardware:**



Exemplo II – Formas de onda no uC



Receber e enviar a letra u (75h)
75h = 01110101b



Receber e enviar um espaço (20h)
20h = 00100000b

Exemplo II – Por interrogação

```
#include <avr/io.h>


#define F_CPU 3686400ul
#define BAUD 57600
#define UBBR_VAL ((F_CPU/(BAUD<<4))-1)

#define LED PB5

void init_usart(void) {
    // Definir baudrate
    UBRR0H = (uint8_t)(UBBR_VAL>>8);
    UBRR0L = (uint8_t) UBBR_VAL;

    // Definir formato da trama
    UCSR0C = (3<<UCSZ00) // 8 data bits
           | (0<<UPM00)  // no parity
           | (0<<USBS0); // 1 stop bit

    // Ativar Rx, Tx
    UCSR0B = (1<<RXEN0) | (1<<TXEN0);
}
```

```
int main(void) {
    unsigned char RecByte;

    DDRB = DDRB | (1<<LED);
    init_usart();

    while(1) {
        // espera que chegue alguma coisa...
        while ((UCSRA0 & (1 << RXC0)) == 0);
        RecByte = UDR0;          // guarda ...

        // espera que UDR0 esteja vazio...
        while ((UCSRA0 & (1 << UDRE0)) == 0);
        UDR0 = RecByte;          // ... devolve

        // complementa estado do LED...
        PORTB = PORTB ^ (1<<LED);
    }
}
```

Exemplo II – Por interrupção

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define F_CPU 3686400ul
#define BAUD 57600
#define UBBR_VAL ((F_CPU/(BAUD<<4))-1)

#define LED PB5

void init_usart(void) {
    // Definir baudrate
    UBRR0H = (uint8_t)(UBBR_VAL>>8);
    UBRR0L = (uint8_t) UBBR_VAL;

    // Definir formato da trama
    UCSR0C = (3<<UCSZ00) // 8 data bits
           | (0<<UPM00)  // no parity
           | (0<<USBS0); // 1 stop bit

    // Ativar Rx, Tx e interrupt Rx
    UCSR0B = (1<<RXEN0) | (1<<TXEN0)
           | (1<<RXCIE0);
    sei();
}
```

```
ISR(USART_RX_vect) {
    unsigned char RecByte;
    RecByte = UDR0;    // guarda ...
    UDR0 = RecByte;    // ... devolve
    PORTB = PORTB ^ (1<<LED);  // complementa LED
}

int main(void) {

    DDRB = DDRB | (1<<LED);
    init_usart();

    while(1);          // ciclo infinito...

}
```

Dúvidas

Para aprofundar:

- Datasheet (Junho 2016): capítulo 24
- Modo normal: p. 225 e seguintes
- Multiprocessador: p. 239
- Tabelas de Baud rate: p. 240
- Application Notes:
  - AVR274: Single-wire Software UART
  - AVR304: Half Duplex Interrupt Driven Sw UART