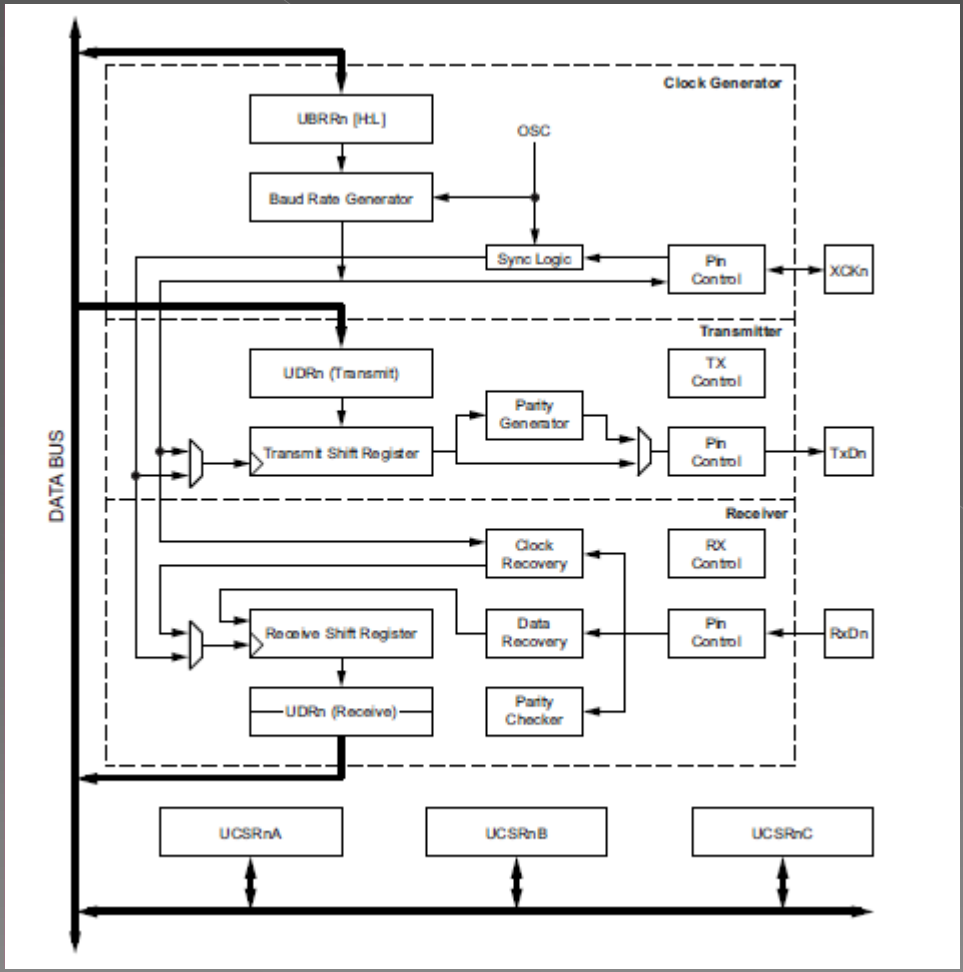
Comunicación Serie - USART

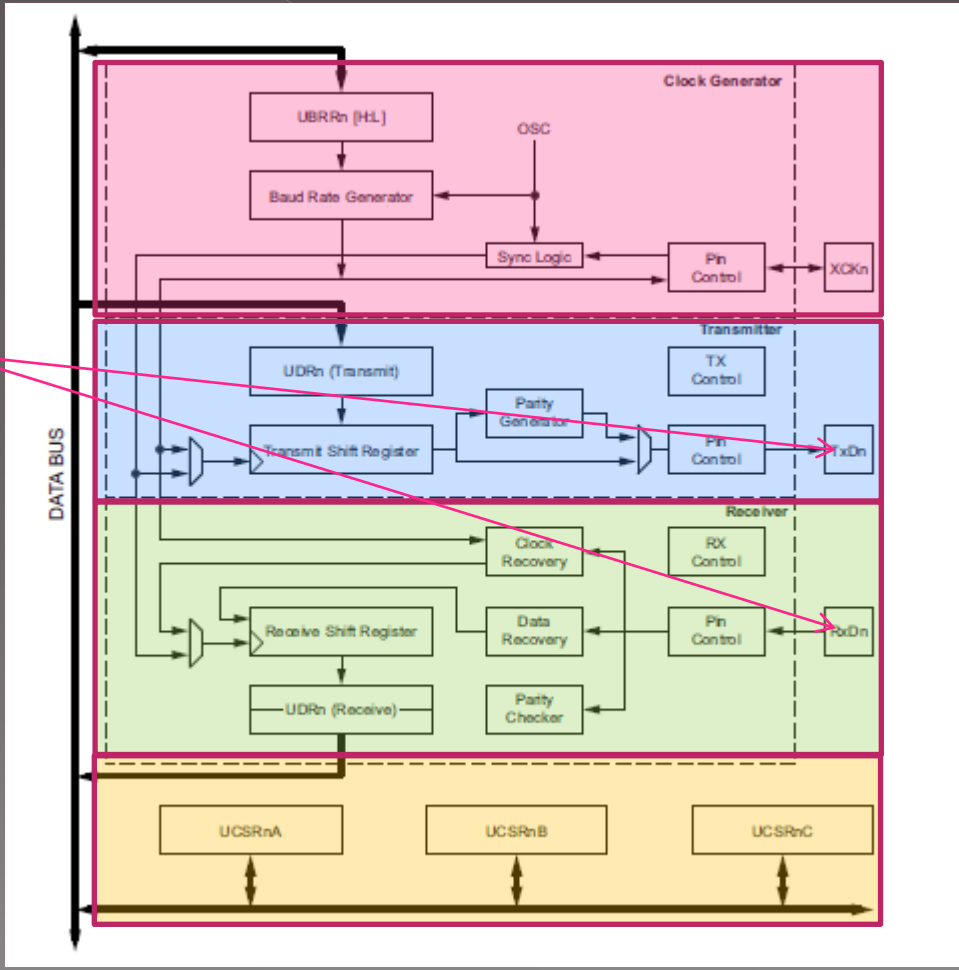
USART

Synchronous/Asynchronous Receiver Transmitter – Transmisor-Receptor Síncrono/Asíncrono Universal



USART

Synchronous/Asynchronous Receiver Transmitter – Transmisor-Receptor Síncrono/Asíncrono Universal



Pines Tx-Rx

Generador de Baudrate

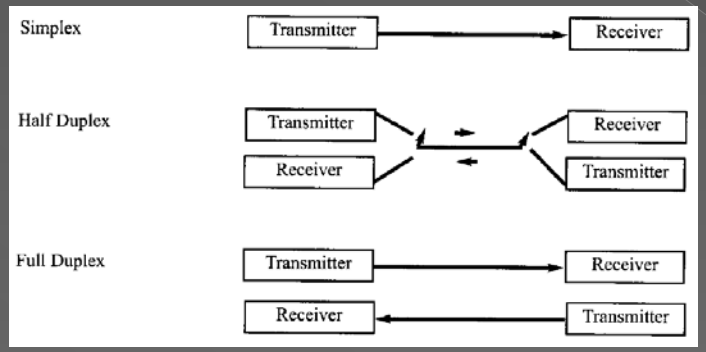
Canal de Transmisión

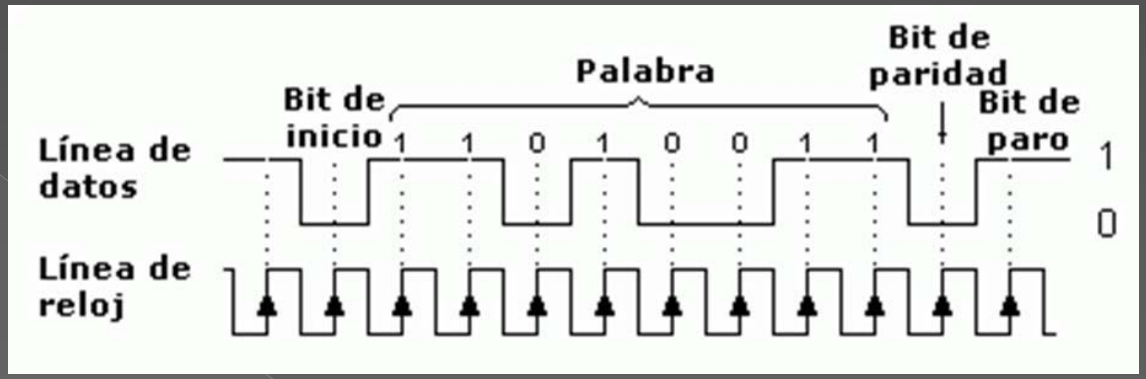
Canal de Recepción

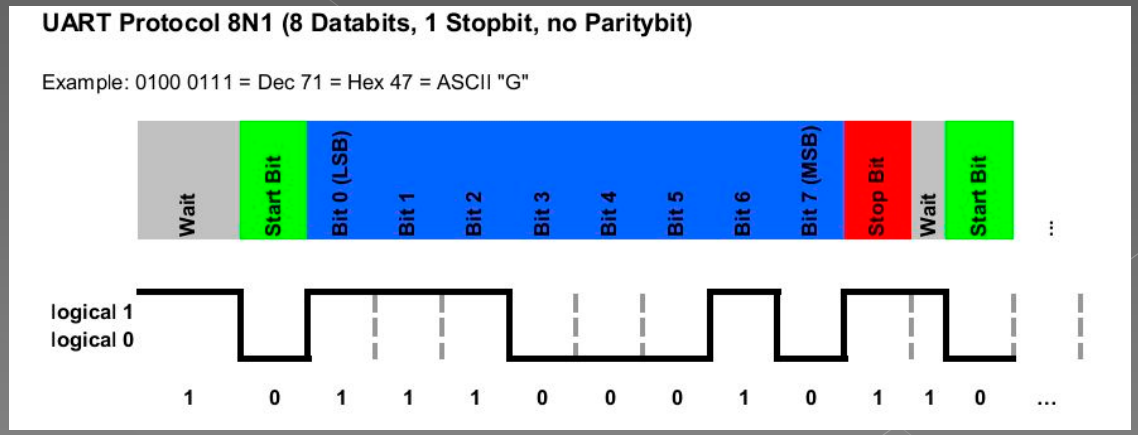
Registros de Configuración

USART

Synchronous/Asynchronous Receiver Transmitter – Transmisor-Receptor Síncrono/Asíncrono Universal







<u>Formato de Comunicación:</u>

- 1 bit de inicio
- 5,6,7,8 o 9 bits de datos
- Con o Sin Bit de Paridad
- 1 o 2 Bits de Parada

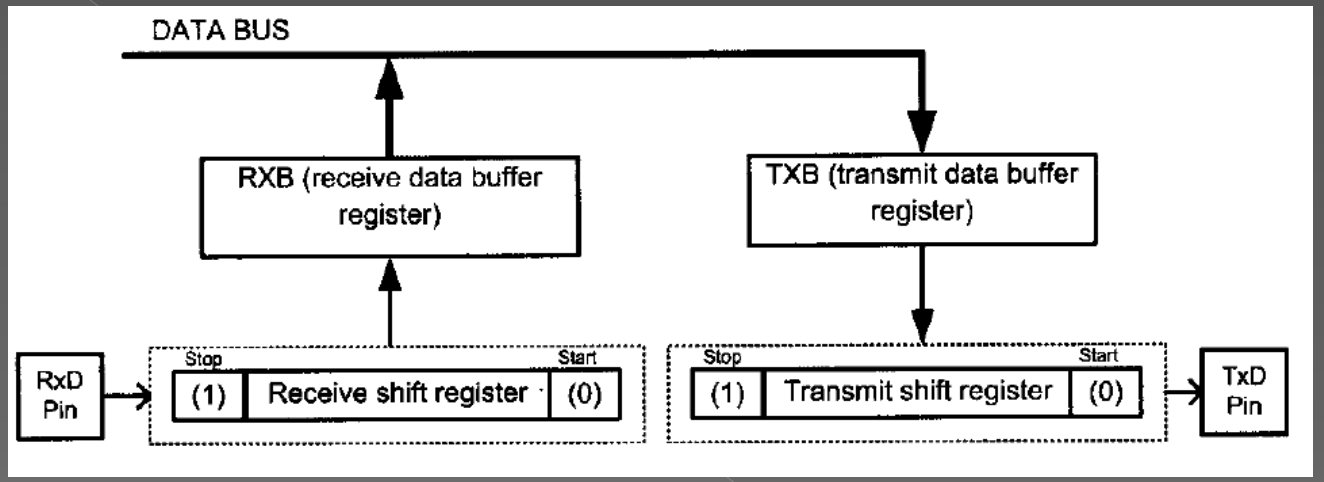
Más general :

8N1 = 8 bits de datos, No paridad, 1 bit de parada

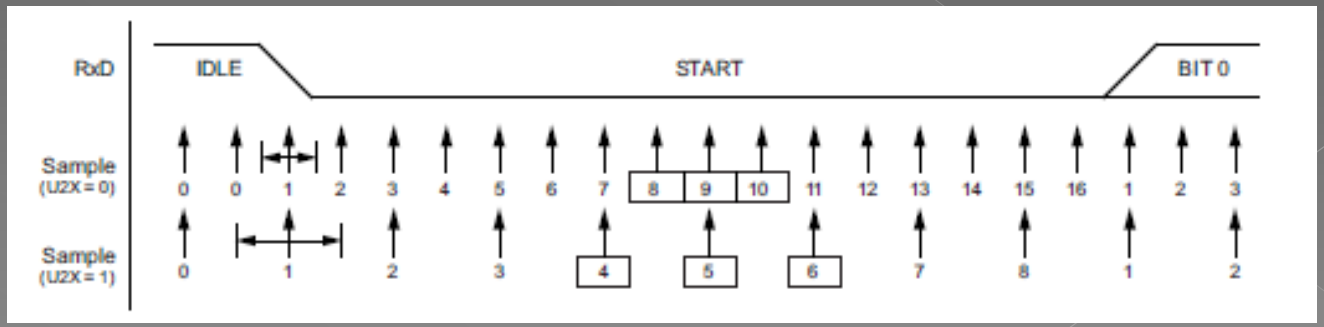
USART

Synchronous/Asynchronous Receiver Transmitter – Transmisor-Receptor Síncrono/Asíncrono Universal

Conversión
Serie-Paralelo (Rx)
Paralelo-Serie (Tx)



Muestreo del Bit de Inicio (similar para Paridad y bits de parada)



USART

Synchronous/Asynchronous Receiver Transmitter – Transmisor-Receptor Síncrono/Asíncrono Universal

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
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	0	0	0	

Registro de Entrada/Salida

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

Registro de Estado y Configuración A

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	

Registro de Estado y Configuración B

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	

Registro de Estado y Configuración C

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

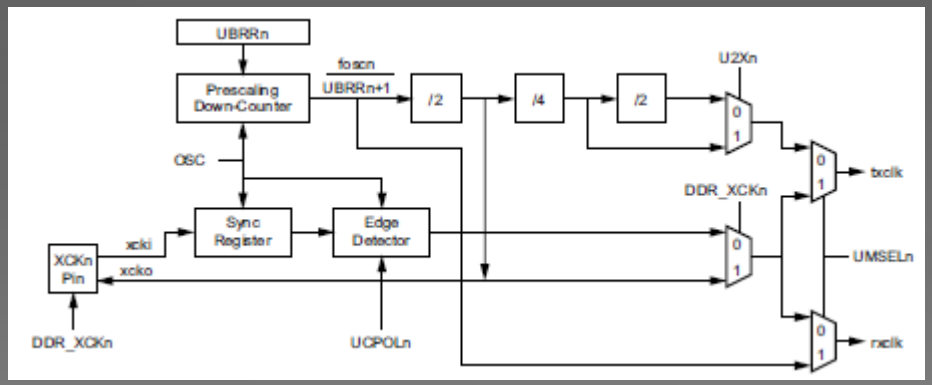
Registro de Baud Rate

USART

Synchronous/Asynchronous Receiver Transmitter – Transmisor-Receptor Síncrono/Asíncrono Universal

Generación de Baud Rate

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	



Registros de Baud Rate

BAUD = Baud Rate deseado
X = Valor a cargar en UBRR

BAUD = Fosc/(16.(X+1))

Así → **X = (Fosc/(16.BAUD))-1**

Error[%] = [(Baud_Rate_Real - Baud_Rate_Esperado)/ Baud_Rate_Esperado]x100
Ej. → para BAUD=9600, Fosc=1Mhz, X= 5
Baud_Rate_Esperado=9600, Xreal= 5,51, Baud_Rate_Real= 10416,66
Error[%] = [(10416,66/9600)-1]x100 = 8,5 % → 7 % usando X = 6

USART

Synchronous/Asynchronous Receiver Transmitter – Transmisor-Receptor Síncrono/Asíncrono Universal

Inicialización de USART

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Baud Rate

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	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	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Flags

Configuración para 8N1 9600bauds

- Comunicación Asíncrona
- Habilita Transmisión y Recepción
- 8 bits
- No Paridad
- 1 Bit de parada

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

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, even parity
1	1	Enabled, odd parity

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

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

USART

Synchronous/Asynchronous Receiver Transmitter – Transmisor-Receptor Síncrono/Asíncrono Universal

Transmisión de caracteres

```
#define F_CPU 16000000
#include <avr/io.h>
#include <util/delay.h>

#define BAUD_RATE = 9600  // (F_CPU/(16*BAUD_RATE))-1 ,16 Mhz, 9600 bauds

int main(void)
{
/* Replace with your application code */

char dato[] = "Hola Mundo !! ¿cómo están? \r";
char  dato2 = 'X';
int i=0;
DDRD = 0x03;
//=======================================
// incializa USART
//=======================================
UBRR0H = 0x00;
UBRR0L = 0x67 ;
UCSR0C = 0b00000110 ;  // asíncrona 81N
UCSR0B = 0b00011000 ;  // habilita Tx y Rx
```

Envía un carácter repetidamente

```
=======================================
while (1)
{
if ((UCSR0A&0x20) == 0x20) //mira la bandera UDRE0
{
UDR0 = dato2;
_delay_ms(100);
}
}
}
```

Envía una cadena de caracteres

```
=======================================
while (1)
{
if (((UCSR0A&0x20) == 0x20)&&(dato[i]!=0x00))
{
UDR0 = dato[i++];
_delay_ms(100);
}
if ((dato[i]==0x00))
{i=0;}
}
}
```

USART

Synchronous/Asynchronous Receiver Transmitter – Transmisor-Receptor Síncrono/Asíncrono Universal

Recepción de caracteres por encuesta

```
#define F_CPU 1000000  //necesario para la librería de delay
#include <avr/io.h>
#include <util/delay.h>

#define BAUD_RATE = 4800     // (F_CPU/(16*BAUD_RATE))-1  1Mhz, 4800 bauds

int main(void)
{
/* Replace with your application code */

 char dato[10]  ;  // buffer de 10 caracteres a recibir
 char  dato2 = 'X';
int i=0;
DDRD = 0x03;
DDRB = 0xFF;
//=======================================
// incializa USART
//=======================================
UBRR0H = 0x00;
UBRR0L = 0x0C ;              // UBRR = 12 para 4800 bauds a 1Mhz
UCSR0C = 0b00000110 ;  // asíncrona 81N
UCSR0B = 0b00011000 ;  // habilita Tx y Rx
```

Recibe un carácter repetidamente

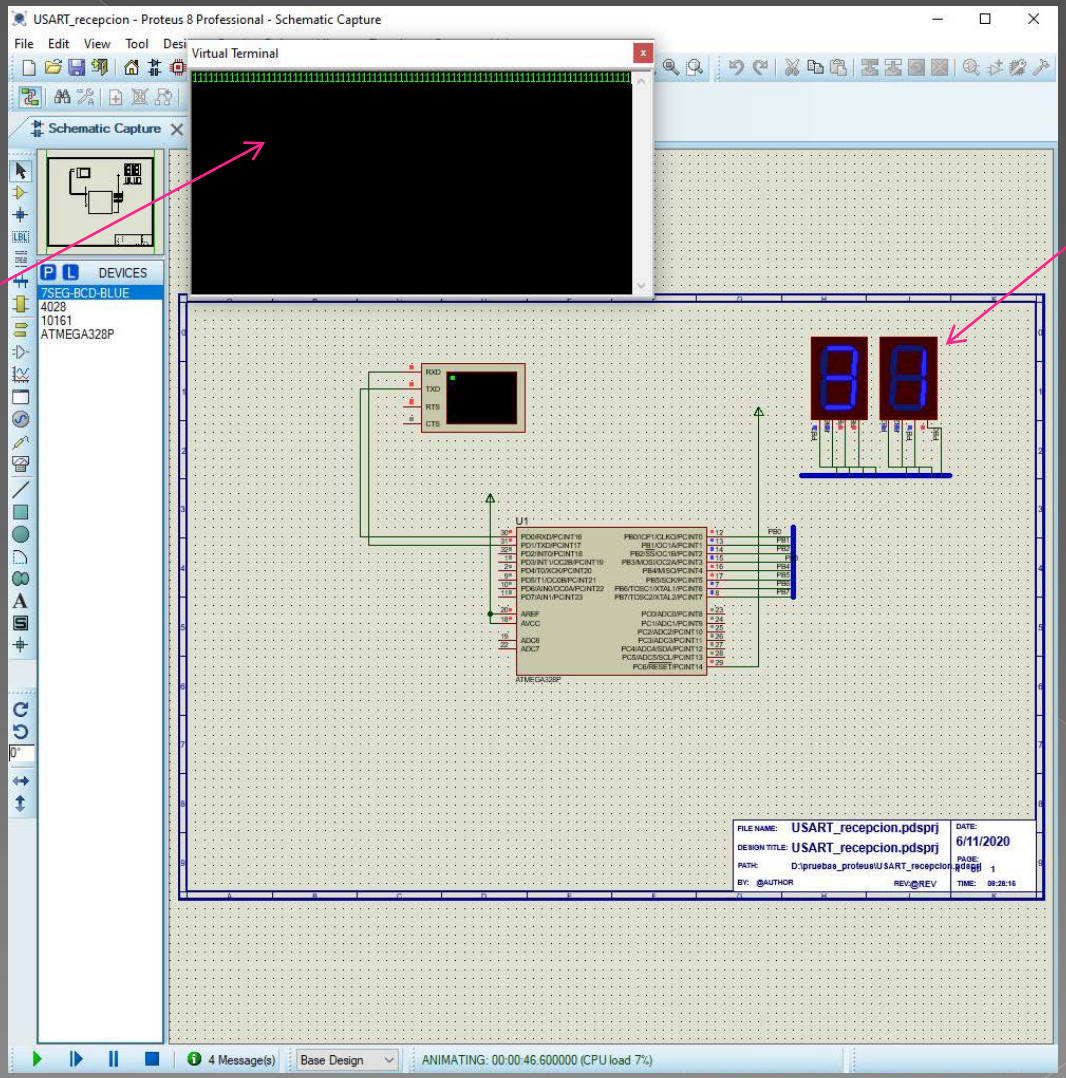
```
while (1)
{
 if((UCSR0A&0x80) == 0x80) // mira el flag RXC0
 {
dato[i++] = UDR0;
 PORTB = dato[i-1];
_delay_ms(100);
 }
```

Hace eco de un carácter recibido

```
=========================================
while (1)
{
 if((UCSR0A&0x80) == 0x80)
 {
dato[i++] = UDR0;
PORTB = dato[i-1];
_delay_ms(100);
 }
  if((UCSR0A&0x20) == 0x20)
 {
UDR0 = dato[i-1] ;
_delay_ms(100);
 }
if (i==0x0A)
   { i=0;}
}
```

USART

Synchronous/Asynchronous Receiver Transmitter – Transmisor-Receptor Síncrono/Asíncrono Universal



Terminal Virtual Serie

Valor ASCII leído

USART

Synchronous/Asynchronous Receiver Transmitter –
Transmisor-Receptor Síncrono/Asíncrono Universal

Otras cosas:

- **FTDI**
- **Administración de recursos**
- **Temporización de tareas**
- **SPI**
- **TWI o I2C**
- **Modelo Productor-Consumidor**

USART

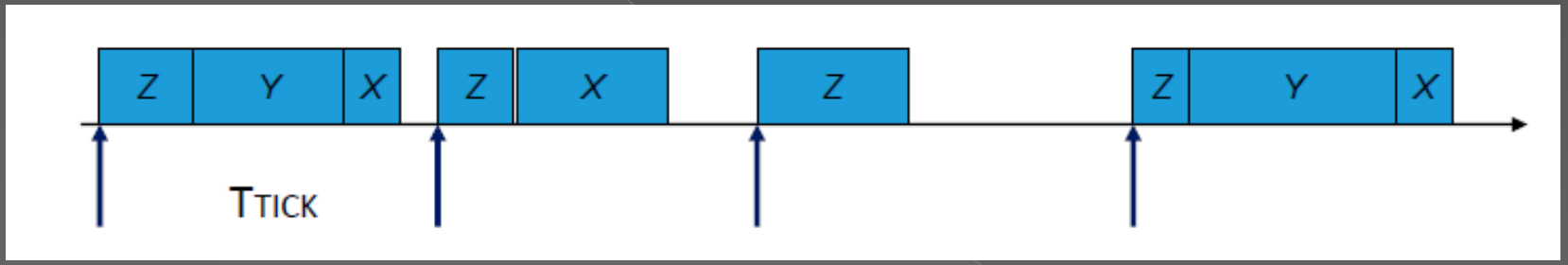
Synchronous/Asynchronous Receiver Transmitter –
Transmisor-Receptor Síncrono/Asíncrono Universal

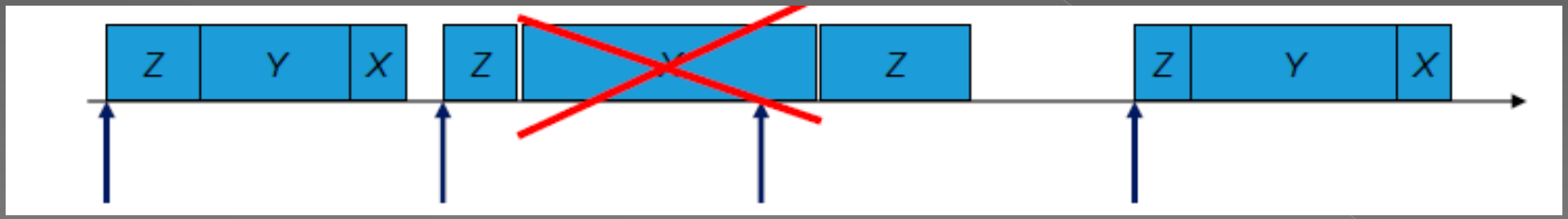
Otras cosas:

- **FTDI**
- **Administración de recursos**
- **Temporización de tareas**
- **SPI**
- **TWI o I2C**
- **Modelo Productor-Consumidor**

Temporización básica con Ticks

Scheduling básico





Temporización-Planificación-Despacho de Tareas

```
Volatile unsigned char Flag_X=0,
Volatile unsigned char Flag_Y=0;
Volatile unsigned char Flag_Z=0;

static unsigned char contX=0,
static unsigned char contY=0;
static unsigned char contZ=0;

/* ------------------------------------------------------------ */
---ISR TIMER : ocurre cada 1 ms
/* ------------------------------------------------------------ */
_interrupt void ISRrtc (void)
{
 SEOS_SCHTasks();
 RTCSC_RTIF=0;
}
```

```
/* ------------------------------------------------------------ */
Variables públicas del planificador
/* ------------------------------------------------------------ */

/* ------------------------------------------------------------ */
Variables privadas del planificador
/* ------------------------------------------------------------ */

void SEOS_SCHTasks (void)
{
  if (++contX==200) {
     Flag_X=1;  //Tarea programada  cada  200 ms
     contX=0;
  }
  if (++contY==50) {
     Flag_Y=1;  //Tarea programada  cada  50 ms
     contY=0;
  }
  if (++contZ==10) {
     Flag_Z=1;  //Tarea programada  cada  10 ms
     contZ=0;
  }
}
```

Planificador

Temporización de Tareas

Interrupción Periódica

Despachador

Prioridad

```
/*------------------------------------------------------------
Main.c
------------------------------------------------------------*/
#include "device.h"
#include "seos.h"

/* ------------------------------------------------------------ */
void main(void)
{
    // Inicializar MCU, RTC, y tareas

    for(;;) {  // Super Loop

      SEOS_Dispatch_Tasks();
      SEOS_Go_To_Sleep();
    }
} /* ------------------------------------------------------------ */
```

```
void SEOS_Dispatch_Tasks(void) {
    if (Flag_Z) {
        Z();  //Tarea programada  cada  10 ms
        Flag_Z =0;
    }
    if (Flag_Y) {
        Y();  //Tarea programada  cada  50 ms
        Flag_Y =0;
    }
    if (Flag_X) {
        X();  //Tarea programada  cada  200 ms
        Flag_X =0;
    }
}
```

+

P

-