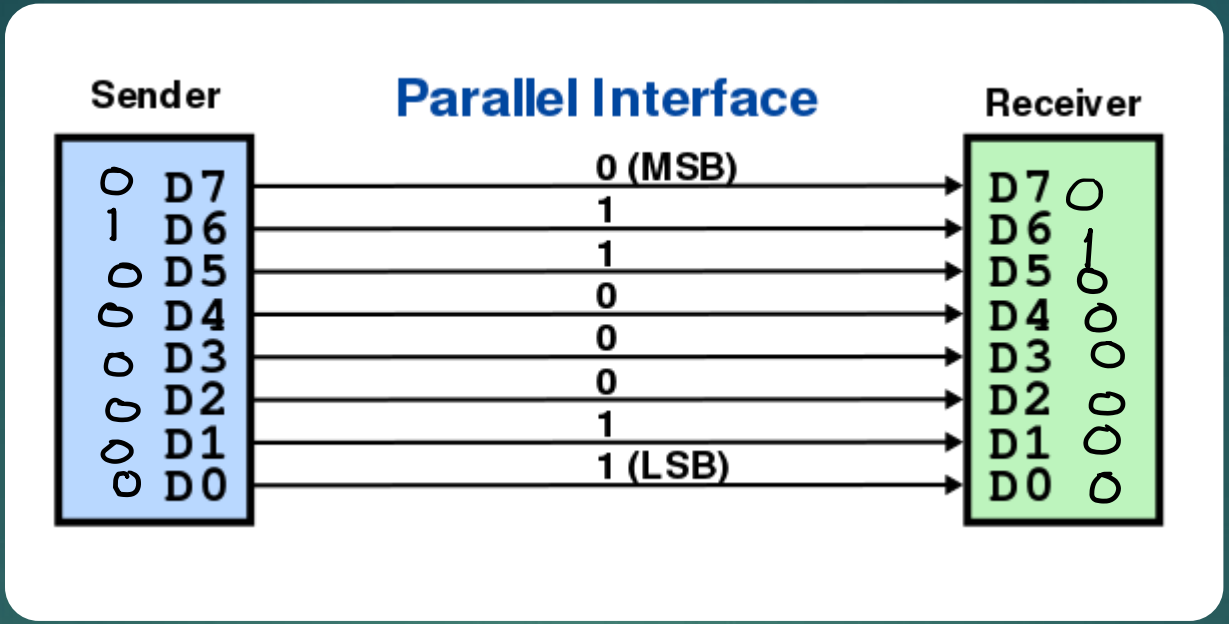
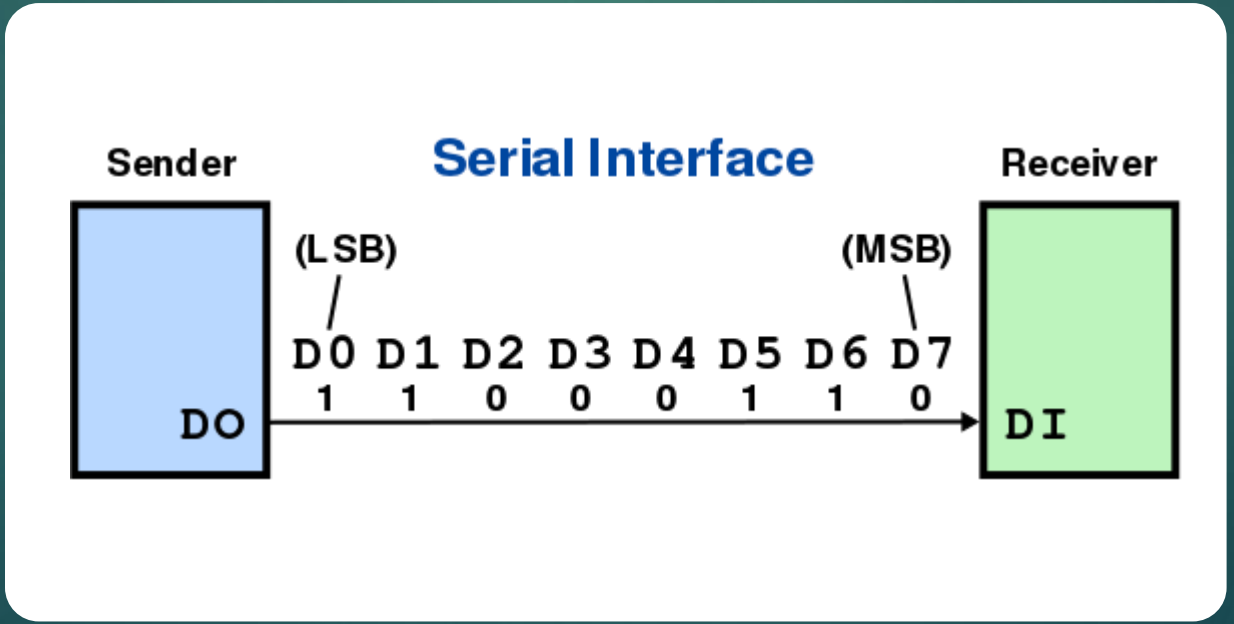
Libro página 186

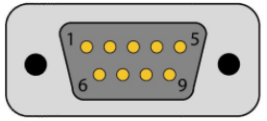
# Fundamentos de Comunicación serial

*Teresa Orvañanos Guerrero*


Sender
Parallel Interface
Receiver
0 D7
1 D6
0 D5
0 D4
0 D3
0 D2
0 D1
0 D0
0 (MSB)
1
1
0
0
0
1
1 (LSB)
D7 0
D6 1
D5 0
D4 0
D3 0
D2 0
D1 0
D0 0

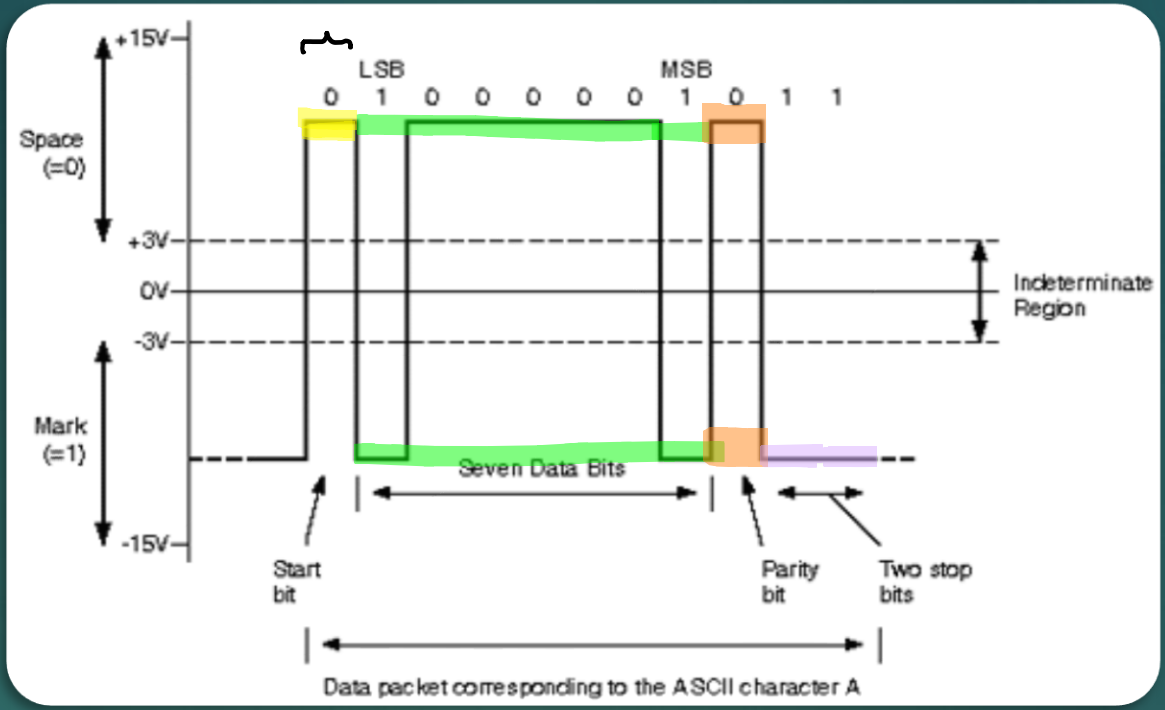

Sender
Serial Interface
Receiver
(LSB)
(MSB)
D0 D1 D2 D3 D4 D5 D6 D7
1 1 0 0 0 1 1 0
DO
DI


**DB9M Connector**




**RS232 Pin Out**

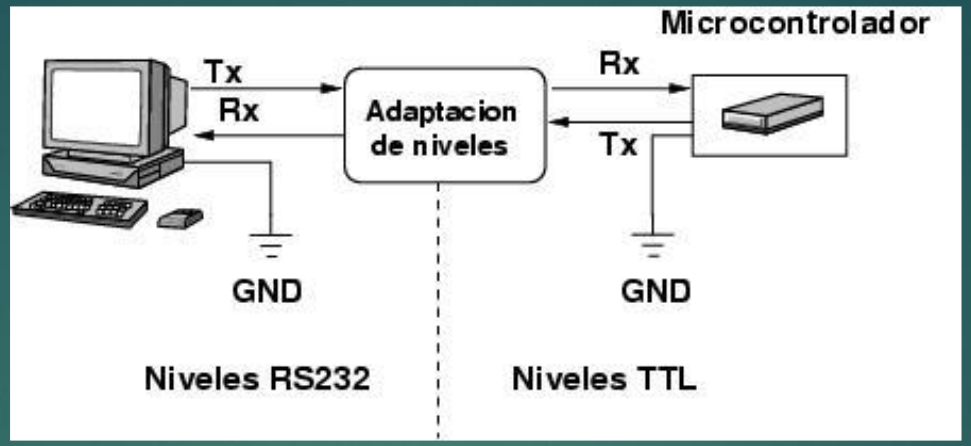
| Pin # | Signal |
|---|---|
| 1 | DCD |
| 2 | RX |
| 3 | TX |
| 4 | DTR |
| 5 | GND |
| 6 | DSR |
| 7 | RTS |
| 8 | CTS |
| 9 | RI |

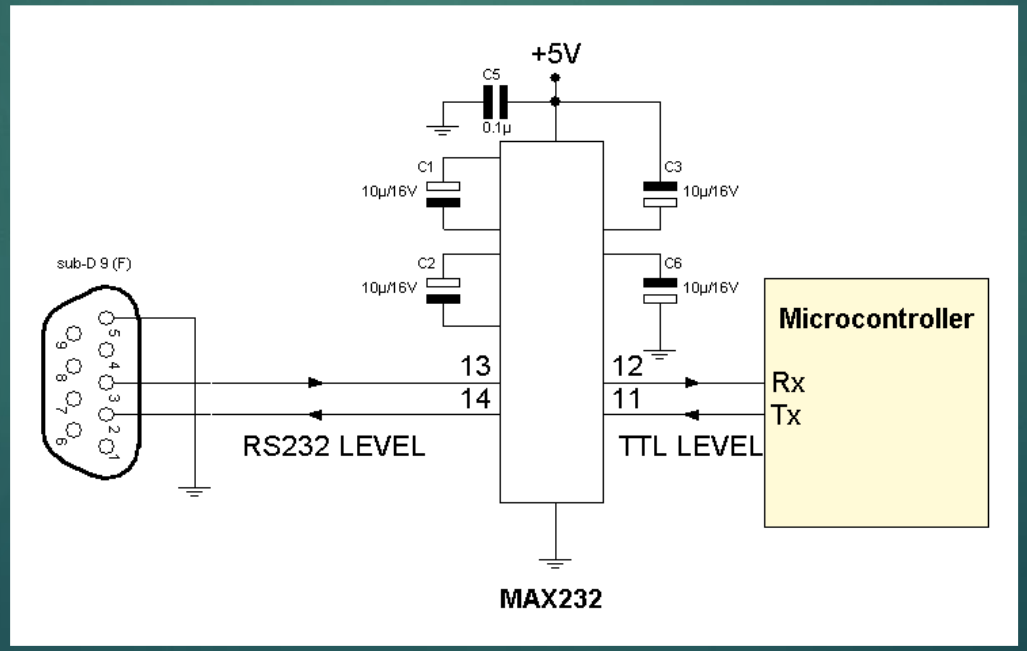


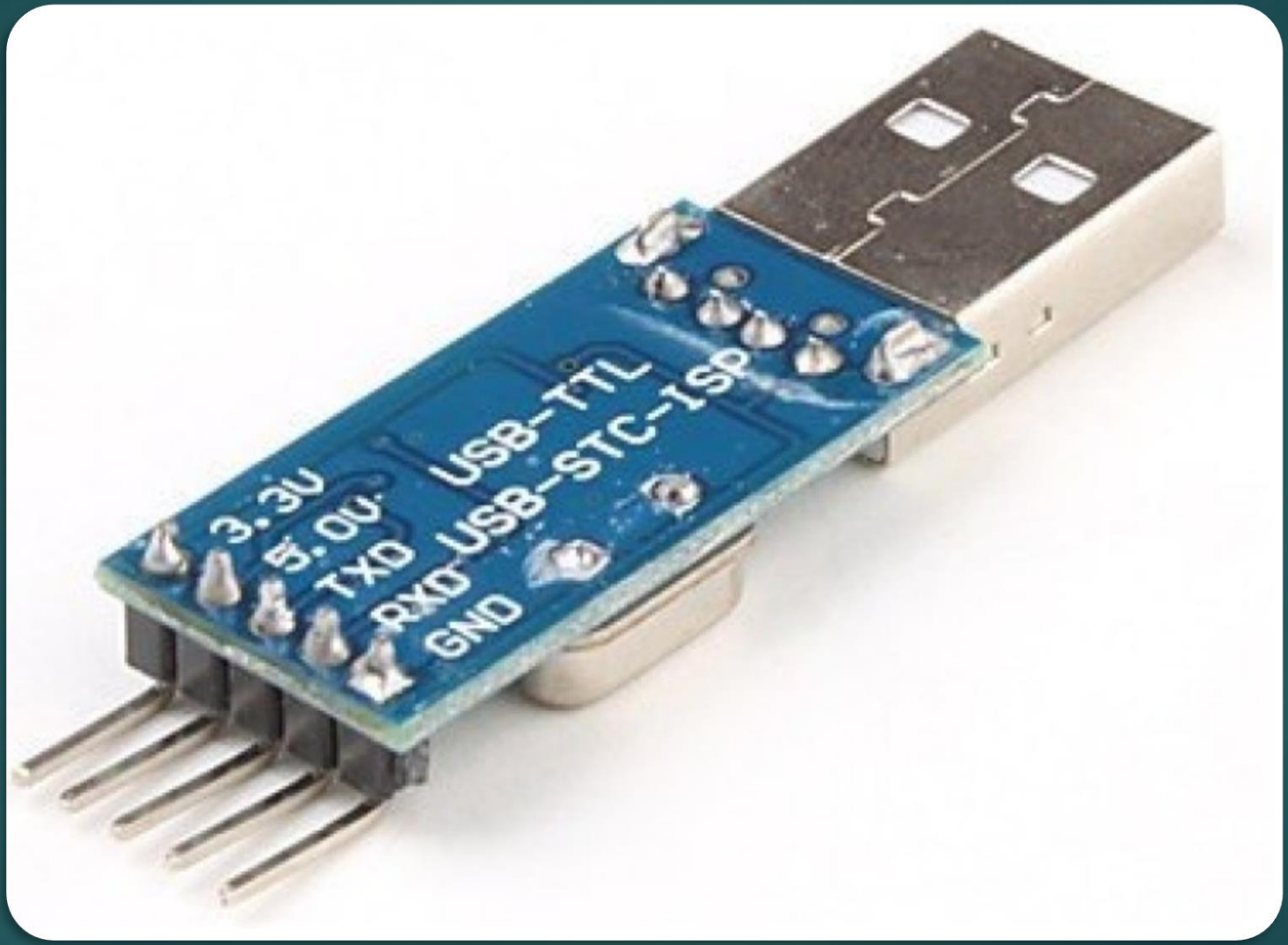

## Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The USART accepts all 30 combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even or odd parity bit
- 1 or 2 stop bits


Microcontrolador
Tx
Rx
Adaptacion
de niveles
Rx
Tx
GND
GND
Niveles RS232
Niveles TTL


+5V
C5
0.1µ
C1
10µ/16V
C3
10µ/16V
C2
10µ/16V
C6
10µ/16V
sub-D 9 (F)
Microcontroller
13
14
12
11
Rx
Tx
RS232 LEVEL
TTL LEVEL
MAX232


3.3V
5.0V
TXD
RXD
GND
USB-TTL
USB-STC-ISP

Datasheet página 140 / Libro página 187

# USART (Universal Synchronous and Asynchronous serial receiver and transmitter)

Teresa Orvañanos Guerrero

En ATmega16A el USART puede trabajar en diferentes modos

- Normal asíncrono
- Asíncrono de doble velocidad

} asíncrona

- síncrono maestro
- síncrono esclavo

} síncrono

En UCSRC en el bit UMSEL

→ 0 = asíncrona

1 = síncrono

UCSRB *

UDR — dato — cargarlo (cuando transmito)
— leerlo (cuando recibo)

UDR

| Bit | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | RXB[7:0] | | | | | | | | UDR (Read) |
| | | TXB[7:0] | | | | | | | | UDR (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 | |

Buffer de transmision y de recepcion

sbis / sbis X if (cero_en_bit(&UDR, 2)) {} // 1a vez saco UDR *
2a vez " UDR *

sbi / cbi ? UDR |= (1<<0); //envio
UDR |= (1<<3); //envio

!!! leer 1° en una variable / cargar UDR completo

uint8_t dato = UDR;

UDR = dato;
0b11110000;

UDR = (1<<1) | (1<<3)

UCSRA

Flag Flag Flag

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | RXC | TXC | UDRE | FE | DOR | PE | U2X | MPCM | UCSRA |
| Read/Write | R | R/W | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |

**RXC**

1 – Hay datos sin leer

0 – El buffer vacío (lectura)

**TXC**

Terminó ☺

1 – Ya se transmit. todos los datos

0 – "Limpio" en automático cuando entra a la int. correspón.

**UDRE**

0 – No listo para transmisión

1 – Buffer vacío listo para transm.

**FE**

0 – Ponerlo! *

1 – Hay un error en el frame

+válido antes de leer al UDR

**DOR**

0 – Ponerlo! *

1 – Hay un data over run (recepción)

+válido antes de leer UDR

**PE**

0 – Ponerlo *

1 – Hay error de paridad

+valido antes de leer al UDR

**U2X**

si síncrona = 0

si asíncrona

0 – normal

1 – doble vel.

**MPCM**

0 – normal

1 – ignora datos recibidos sin info de la dir

* Errores en Recepción

Interruption enable

~EN Enable

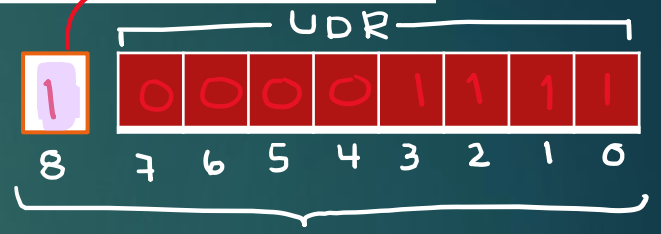
# bits

el 8vo bit de UDR

UCSRB

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | RXCIE | TXCIE | UDRIE | RXEN | TXEN | UCSZ2 | RXB8 | TXB8 | UCSRB |
| 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 | |

UDR = 0b00000111




¡ 9 bits en total !
Leer o escribir ANTES de leer o escribir al UDR

RXCIE 1 – Int. cuando reciba un dato (!!!)
0 – No int

TXCIE 1 – Int. cuando terminó la trans. ←
0 – No int

UDRIE – 1 – Int. cuando listo para trans. ←
0 – No. int

RXEN – 0 – Deshabilita Recepción
1 – Habilitar Recepción

TX EN – 0 – Deshabilita Transmision
1 – Habilitar transm...

Table 67. UCSZ Bits Settings

| UCSZ2 | UCSZ1 | UCSZ0 | 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 |

## Transmitir 9 bits

UCSRB en bit 0

0b [1] 0101 0101

UDR

① UCSRB |= (1<<0);

② UDR = 0b01010101;

## Recibir 9 bits

// 1° leer bit 1 del UCSRB

① uint16_t dato=0;
if (uno_en_bit(&UCSRB, 1)) { dato = (1<<8); }

// leer UDR

② uint8_t temp = UDR; ~~&UDR~~
dato |= temp;

bit 1 de UCSRB

Recibí 0b 1 01010101

UDR

dato = 0b 1 0000 0000

1 0000 0000
0101 0101
---
10101 0101

No paridad
1 bit stop
8 bits

→ 1 0 0 0 0 1 1 0

siempre 1

tamaño dato *

se utiliza en comunicación síncrona

## UCSRC

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | URSEL | UMSEL | UPM1 | UPM0 | USBS | UCSZ1 | UCSZ0 | UCPOL | UCSRC |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | |

1 1

**Table 64.** UMSEL Bit Settings

| UMSEL | Mode |
|---|---|
| 0 | Asynchronous Operation |
| 1 | Synchronous Operation |

**Table 65.** UPM Bits Settings

| UPM1 | UPM0 | Parity Mode | |
|---|---|---|---|
| 0 | 0 | Disabled | No ! |
| 0 | 1 | Reserved | |
| 1 | 0 | Enabled, Even Parity | Par |
| 1 | 1 | Enabled, Odd Parity | Impar |

Paridad

igual q' en el otro dispositivo (comp)

**Table 66.** USBS Bit Settings

| USBS | Stop Bit(s) |
|---|---|
| 0 | 1-bit |
| 1 | 2-bit |

igual q' en el otro disp. (comp)

*

**Table 67.** UCSZ Bits Settings

| UCSZ2 | UCSZ1 | UCSZ0 | 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 |

UBRR 16bits → UBRRH : UBRRL

# para velocidad de transmisión

UBRRH : UBRRL

0000 0000

25

**Table 69.** Examples of UBRR Settings for Commonly Used Oscillator Frequencies

| Baud Rate (bps) | $f_{osc}$ = 1.0000 MHz | | | | $f_{osc}$ = 1.8432 MHz | | | | $f_{osc}$ = 2.0000 MHz | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | U2X = 0 | | U2X = 1 | | U2X = 0 | | U2X = 1 | | U2X = 0 | | U2X = 1 | |
| | UBRR | Error | UBRR | Error | UBRR | Error | UBRR | Error | UBRR | Error | UBRR | Error |
| 2400 | 25 | 0.2% | 51 | 0.2% | 47 | 0.0% | 95 | 0.0% | 51 | 0.2% | 103 | 0.2% |
| 4800 | 12 | 0.2% | 25 | 0.2% | 23 | 0.0% | 47 | 0.0% | 25 | 0.2% | 51 | 0.2% |
| 9600 | 6 | -7.0% | 12 | 0.2% | 11 | 0.0% | 23 | 0.0% | 12 | 0.2% | 25 | 0.2% |
| 14.4k | 3 | 8.5% | 8 | -3.5% | 7 | 0.0% | 15 | 0.0% | 8 | -3.5% | 16 | 2.1% |
| 19.2k | 2 | 8.5% | 6 | -7.0% | 5 | 0.0% | 11 | 0.0% | 6 | -7.0% | 12 | 0.2% |
| 28.8k | 1 | 8.5% | 3 | 8.5% | 3 | 0.0% | 7 | 0.0% | 3 | 8.5% | 8 | -3.5% |
| 38.4k | 1 | -18.6% | 2 | 8.5% | 2 | 0.0% | 5 | 0.0% | 2 | 8.5% | 6 | -7.0% |
| 57.6k | 0 | 8.5% | 1 | 8.5% | 1 | 0.0% | 3 | 0.0% | 1 | 8.5% | 3 | 8.5% |
| 76.8k | – | – | 1 | -18.6% | 1 | -25.0% | 2 | 0.0% | 1 | -18.6% | 2 | 8.5% |
| 115.2k | – | – | 0 | 8.5% | 0 | 0.0% | 1 | 0.0% | 0 | 8.5% | 1 | 8.5% |
| 230.4k | – | – | – | – | – | – | 0 | 0.0% | – | – | – | – |
| 250k | – | – | – | – | – | – | – | – | – | – | 0 | 0.0% |
| Max (1) | 62.5 kbps | | 125 kbps | | 115.2 kbps | | 230.4 kbps | | 125 kbps | | 250 kbps | |

* fórmula

UBRR= 103;

**Table 70.** Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

| Baud Rate (bps) | $f_{osc}$ = 3.6864 MHz | | | | $f_{osc}$ = 4.0000 MHz | | | | $f_{osc}$ = 7.3728 MHz | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | U2X = 0 | | U2X = 1 | | U2X = 0 | | U2X = 1 | | U2X = 0 | | U2X = 1 | |
| | UBRR | Error | UBRR | Error | UBRR | Error | UBRR | Error | UBRR | Error | UBRR | Error |
| 2400 | 95 | 0.0% | 191 | 0.0% | 103 | 0.2% | 207 | 0.2% | 191 | 0.0% | 383 | 0.0% |
| 4800 | 47 | 0.0% | 95 | 0.0% | 51 | 0.2% | 103 | 0.2% | 95 | 0.0% | 191 | 0.0% |
| 9600 | 23 | 0.0% | 47 | 0.0% | 25 | 0.2% | 51 | 0.2% | 47 | 0.0% | 95 | 0.0% |
| 14.4k | 15 | 0.0% | 31 | 0.0% | 16 | 2.1% | 34 | -0.8% | 31 | 0.0% | 63 | 0.0% |
| 19.2k | 11 | 0.0% | 23 | 0.0% | 12 | 0.2% | 25 | 0.2% | 23 | 0.0% | 47 | 0.0% |
| 28.8k | 7 | 0.0% | 15 | 0.0% | 8 | -3.5% | 16 | 2.1% | 15 | 0.0% | 31 | 0.0% |
| 38.4k | 5 | 0.0% | 11 | 0.0% | 6 | -7.0% | 12 | 0.2% | 11 | 0.0% | 23 | 0.0% |
| 57.6k | 3 | 0.0% | 7 | 0.0% | 3 | 8.5% | 8 | -3.5% | 7 | 0.0% | 15 | 0.0% |
| 76.8k | 2 | 0.0% | 5 | 0.0% | 2 | 8.5% | 6 | -7.0% | 5 | 0.0% | 11 | 0.0% |
| 115.2k | 1 | 0.0% | 3 | 0.0% | 1 | 8.5% | 3 | 8.5% | 3 | 0.0% | 7 | 0.0% |
| 230.4k | 0 | 0.0% | 1 | 0.0% | 0 | 8.5% | 1 | 8.5% | 1 | 0.0% | 3 | 0.0% |
| 250k | 0 | -7.8% | 1 | -7.8% | 0 | 0.0% | 1 | 0.0% | 1 | -7.8% | 3 | -7.8% |
| 0.5M | – | – | 0 | -7.8% | – | – | 0 | 0.0% | 0 | -7.8% | 1 | -7.8% |
| 1M | – | – | – | – | – | – | – | – | – | – | 0 | -7.8% |
| Max (1) | 230.4 kbps | | 460.8 kbps | | 250k bps | | 0.5 Mbps | | 460.8 kbps | | 921.6 kbps | |

UDR — dato (8 bits)

UCSRA — banderas e indicadores

- - - - -

UCSRB — Interrupciones? / Transm y/o Recep / # bits* / 8vo bit.

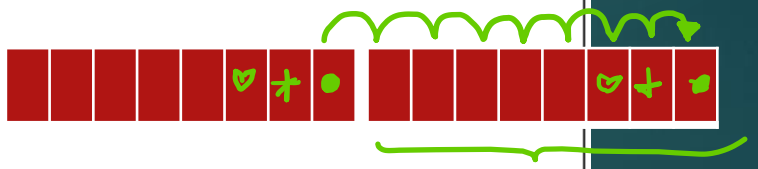
UCSRC — Asíncrona / Paridad / # bits* / bits parada

UBRR (UBRRH : UBRRL) — velocidad (tabla o formula)

# INICIALIZAR / CONFIGURAR

C Code Example[1]

```
#define ~~FOSC~~ FCPU 1843200// Clock Speed
#define BAUD 9600
#define MYUBRR ~~FOSC~~ FCPU/16/BAUD-1
void main( void )
{
        :.
        USART_Init ( MYUBRR );
        :.
}
void USART_Init( ~~unsigned int~~ uint16_t ubrr)
{
        /* Set baud rate */
        UBRRH = (~~unsigned char~~ uint8_t)(ubrr>>8);
        UBRRL = (~~unsigned char~~ uint8_t)ubrr;
        /* Enable receiver and transmitter */
        UCSRB = (1<<RXEN)|(1<<TXEN)| (1<<RXCIE); (Recep, trans, Int)
        /* Set frame format: 8data, 2stop bit */
        UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0); // 8 bits
                                                 2 stop  sin paridad
}
```



```
void main (void)
{
  USAR_Transmit(1);
}
```

0 = FALSO

NO 0 = VERDADERO (1)

C Code Example[1]

```
void USART_Transmit( ~~unsigned char~~ data )
{
        /* Wait for empty transmit buffer */
        while ( !( UCSRA & (1<<UDRE)) ) {}

        /* Put data into buffer, sends the data */
        UDR = data;
}
```

uint8_t

X

UDRE

UCSRA

X

&

0 0 1 0 0 0 0 0

0 0 X 0 0 0 0 0

si UDRE = 0 ∴ * = 0b00000000

NO LISTO

! (falso)

VERDAD

si UDRE = 1 ∴ * = 0b00100000

LISTO

! (verdadero)

FALSO

```
void main (void) {
    ═══
    uint8_t dato= USART_Receive();
    ═══
}
```

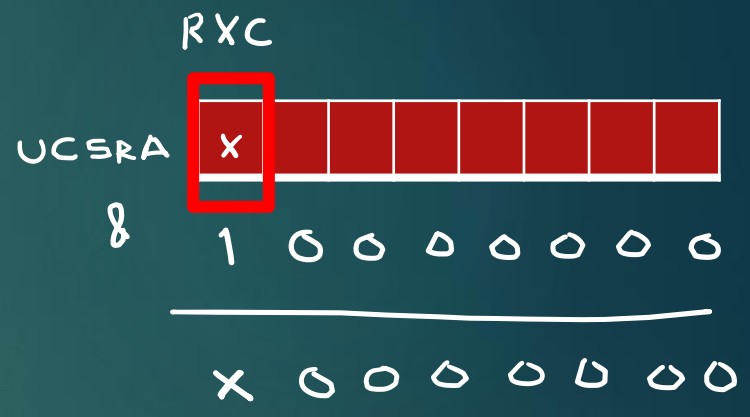
0 falso

No 0 verdad

C Code Example[1]

```
~~unsigned char~~ USART_Receive( void )
{   uint8_t
    /* Wait for data to be received */
    while ( !(UCSRA & (1<<RXC)) ) {} ; //TRABA !!!
                  *   X

    /* Get and return received data from buffer */
    return UDR;
}
```




si RXC = 0 ∴ *= 0b00000000

no hay datos

!(falso)

VERDAD

si RXC = 1 ∴ *= 0b10000000

si hay datos

!(verdad)

FALSO

!!! TRABA

Vs

Interrup Recep

# INTERRUPCIONES

* Inicializar serial con RXCIE=1

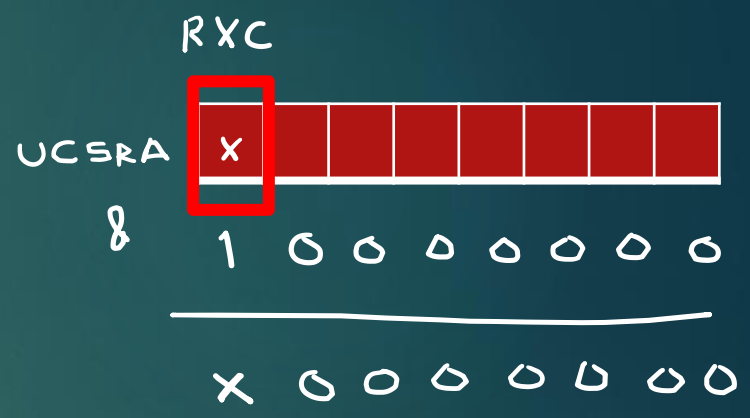
```
volatile uint8_t dato;

int main(void)
{    *
     /* Replace with your application code */
     while (1)
     {
     }
}

ISR(USART_RXC_vect)
{
dato = UDR
}
```

C Code Example[1]

```
void USART_Flush( void )
{
        uint8_t
        ~~unsigned char~~ dummy;
        while ( UCSRA & (1<<RXC) ) dummy = UDR;
}
```




si RXC = 0 ∴ * = 0b00000000 (falso)

no hay datos

si RXC = 1 ∴ * = 0b10000000 (verdad)

si hay datos