

# Protocolo RS-232 Explicación



explicame que es el protocolo rs-232 y que velocidad tiene y como puedo interpretar esa imagen

## ¿Qué es el protocolo RS-232?

**RS-232** es un estándar para la transmisión de datos en serie, comúnmente usado en la comunicación entre computadoras y periféricos (como módems, microcontroladores, etc.). Sus principales características son:

- **Comunicación asíncrona:** no hay reloj compartido, se usan bits de inicio y parada.
- **Voltajes invertidos:**
  - Lógica 1: **entre -3V y -15V** (pero en muchas implementaciones modernas, como esta, se usa 0V).
  - Lógica 0: **entre +3V y +15V** (aquí usan 5V).
- **Trama estándar:** 1 bit de inicio, 8 bits de datos (LSB primero), 1 bit de parada (puede variar).

## ¿Qué velocidad tiene?

La **velocidad (baudrate)** depende del dispositivo. Se mide en **baudios** o bits por segundo. Para determinarlo en un osciloscopio:

1. **Mide la duración de un bit** (ancho de cada pulso).
2. **Aplica la fórmula:**

$$\text{Baudrate} = \frac{1}{\text{duración del bit}}$$

## Cómo interpretar la imagen (paso a paso)

### 1. Análisis temporal:

- Cada división en horizontal equivale a **50 µs** (como dice "Time 50.0 µs").
- El ancho de un bit se puede estimar midiendo visualmente.
- Por ejemplo, si cada bit dura **aproximadamente 104 µs**, entonces:

$$\text{Baudrate} = \frac{1}{104 \times 10^{-6}} \approx 9600 \text{ baudios}$$

### 2. Identificar los bits:

- Hay **lógica inversa**: 5V = 0 lógico, 0V = 1 lógico.

- Primer pulso bajo (0 V) es el **bit de inicio**.
- Luego vienen 8 bits de datos.
- Finalmente, un pulso alto (5 V) = bit de parada.

### 3. Decodificación:

- Observando los niveles lógicos, tenemos:
  - Inicia en 0 V = bit de inicio (1 lógico).
  - Luego: alto-bajo-bajo-bajo-alto-bajo-bajo-bajo (5V-0V-0V-0V-5V-0V-0V-0V) → que equivale a:

nginx								
Nivel	→ 5V	0V	0V	0V	5V	0V	0V	0V
Lógico	→ 0	1	1	1	0	1	1	1

→ LSB primero → 11101110

→ En binario: **11101110**

→ En hexadecimal: **0xEE**

→ En ASCII: **'I'**

### Respuesta final:

- **Carácter ASCII:** 'I' (código 0xEE)
- **Baudrate:** aproximadamente **9600 baudios**

¿Te gustaría que te explique cómo sacar el valor binario paso por paso desde la imagen del osciloscopio?



para una duración de bit de 50 µs, tengo el 101100111011 a un baurate de 20 000 ayudame a determinar el caracter, ten encuenta el bit de inicio o de parada segun sea el caso para el rs-232

Perfecto. Vamos a resolver el ejercicio paso por paso, considerando:

### Datos entregados por ti:

- Duración de bit: **50 µs**
- Señal observada (en lógica invertida RS-232):  
**101100111011**
- Baudrate:

$$\frac{1}{50\mu s} = 20,000 \text{ baudios}$$

- RS-232 con lógica **inversa**:

- 0V → 1 lógico
- 5V → 0 lógico

## ✓ Paso 1: Identificar bit de inicio, datos y parada

El protocolo RS-232 típico envía:

```
css

[Start][Data bits (LSB primero)][Stop]
```

Y en **RS-232 invertido**, el **bit de inicio es un 0 lógico (transmitido como 5 V)** → representado como un 1 en la señal invertida que tú diste.

Así que:

```
arduino

Señal recibida: 1 0 1 1 0 0 1 1 1 0 1 1
                ↑
                bit de inicio
```

Ahora dividimos:

- Bit de inicio: 1 → lógico 0 (correcto)
- 8 bits de datos: 0 1 1 0 0 1 1 1 (en el orden que llega)
- Bit de parada: 0 → lógico 1 (válido)

## ✓ Paso 2: Interpretar los bits de datos

La cadena de datos recibida es:

```
java

Bits recibidos (lógica invertida): 0 1 1 0 0 1 1 1
Bits reales (desinvertidos):      1 0 0 1 1 0 0 0
```

Recordando que RS-232 envía el **LSB primero**, invertimos el orden:

```
go

LSB primero: 0 0 0 1 1 0 0 1 → en binario real
```

## ✓ Paso 3: Convertir a ASCII

```
vbnet

Binario: 00011001
Decimal: 25
```

ASCII: → Código 25 = "EM" (End of Medium), carácter de control

## ✓ Resultado final:

- **Carácter ASCII transmitido:** código **25 decimal**, que corresponde a un carácter de control llamado **EM (End of Medium)**.
- **Baudrate: 20,000 baudios**

¿Quieres que lo represente gráficamente en bits o con diagrama?



explicame eso de los bits reales en : Bits recibidos (lógica invertida): 0 1 1 0 0 1 1 1  
Bits reales (desinvertidos): 1 0 0 1 1 0 0 0 y que significa el caracter em ?

Claro, vamos por partes para explicarlo bien:

## ✓ 1. ¿Qué significa lógica invertida en RS-232?

El protocolo **RS-232** tradicional (especialmente en microcontroladores o PC) suele usar **lógica inversa**, es decir:

Voltaje	Valor lógico real	Bit recibido
0V	1 lógico	0
5V	0 lógico	1

Entonces, cuando tú "ves" un 0 en la señal medida, en realidad se está transmitiendo un 1, y viceversa.

## ✓ 2. Interpretación paso a paso

Tú observaste esta secuencia:

```
java
```

```
Bits recibidos (RS-232 invertido): 0 1 1 0 0 1 1 1
```

Si lo llevamos a su **valor lógico real**, simplemente invertimos cada bit:

```
0 → 1
1 → 0
1 → 0
0 → 1
0 → 1
1 → 0
1 → 0
1 → 0
```

Resultado:

```
java

Bits reales (desinvertidos): 1 0 0 1 1 0 0 0
```

### ✓ 3. Reordenar en formato LSB-first (como se transmite en RS-232)

En RS-232, los bits **se transmiten del menos significativo (LSB) al más significativo (MSB)**, así que debemos leer la secuencia **de derecha a izquierda** para interpretar correctamente el byte.

Entonces, tomamos:

```
1 0 0 1 1 0 0 0
```

y reordenamos (de LSB a MSB):

```
→ 0 0 0 1 1 0 0 1
```

### ✓ 4. Convertir a decimal y ASCII

```
CSS

Binario (MSB a LSB): 00011001
Decimal: 25
```

Este número **25 decimal** en la tabla ASCII corresponde al carácter de control:

 **EM** → *End of Medium*

### ✓ 5. ¿Qué es el carácter ASCII EM (End of Medium)?

- Es un **carácter de control** definido por la norma ASCII.
- Su **código decimal es 25**.
- Su uso original (hoy casi obsoleto) era en **comunicaciones por cinta magnética o terminales de texto**, para señalar el "fin del medio de almacenamiento".

No es un carácter imprimible, como una letra o número, sino un comando de control, por eso no verás una letra visible si lo imprimes, aunque sí se puede interpretar por algunos sistemas.

### ✓ Resumen Final

- **Bits recibidos:** 0 1 1 0 0 1 1 1
- **Bits reales (desinvertidos):** 1 0 0 1 1 0 0 0
- **Orden correcto (LSB first):** 00011001
- **Decimal:** 25
- **ASCII:** EM (*End of Medium*)

¿Quieres hacer otro ejemplo o verificar un carácter diferente?



en la grafica se ve 010011000100 pero yo te di ya invertido 101100111011 entonces cual es caracter ?

Excelente observación. Vamos a aclararlo paso a paso para evitar confusión.

## ✓ 1. Lo que tú diste:

Me diste esta secuencia ya **invertida** (como se vería en la señal lógica medida en RS-232 invertido):

```
101100111011
```

Me dijiste explícitamente que ya **está invertida**, así que esto es lo que **el osciloscopio muestra** (voltajes).

## ✓ 2. Interpretamos los bits reales (desinvertiendo):

Aplicamos la inversión bit a bit:

```
makefile

Inversión:
1 → 0
0 → 1
1 → 0
1 → 0
0 → 1
0 → 1
1 → 0
1 → 0
1 → 0
0 → 1
1 → 0
1 → 0
```

Entonces:

yaml

Bits reales: 010011000100

### ✓ 3. Separación en campos del protocolo RS-232

Recordemos la estructura típica:

css

[Start][8 Data bits][Stop]

Vamos a separar:

pgsql

Start bit: 0 → válido (recuerda que el bit de inicio debe ser un 0 lógico = invertido sería un 1)

Data bits (LSB primero): 1 0 0 1 1 0 0 0

Stop bit: 1 → válido (un bit de parada debe ser 1 lógico)

Entonces los **8 bits de datos** reales son:

1 0 0 1 1 0 0 0

### ✓ 4. Reordenar a MSB first

Ya que se transmite **LSB primero**, reordenamos los bits así:

java

Datos transmitidos (LSB → MSB): 1 0 0 1 1 0 0 0

↑
↑

LSB
MSB

Reordenado (MSB → LSB): 0 0 0 1 1 0 0 1

### ✓ 5. Convertimos a decimal y ASCII

vbnet

Binario: 00011001

Decimal: 25

ASCII: EM (End of Medium)

### ✓ Resultado final

- **Bits reales (desinvertidos):** 010011000100
- **Bits de datos:** 10011000 (LSB primero)
- **Reordenado (MSB primero):** 00011001
- **Decimal:** 25
- **Carácter ASCII:** EM (*End of Medium*)

¿Deseas que lo represente como trama RS-232 completa o ver qué pasaría si ese último bit fuera parte de otro carácter?



dice que con logica inversa 5v= 0 0V= 1 teniendo eso en cuenta yo te paso los valores 101100111011 dime el caracter 5v= 0 0V= 1

Perfecto, gracias por la aclaración final. Vamos a resolverlo con esa base clara:

### ✓ Datos que me diste:

Cadena:

101100111011

Esta es la **señal medida en voltaje**, por tanto:

- 1 → 0 V → representa un 1 lógico
- 0 → 5 V → representa un 0 lógico

Es decir, tú ya estás dando los valores de voltaje que viste, y ahora los vamos a **convertir a bits lógicos reales**.

### ✓ Paso 1: Convertir voltajes a bits lógicos reales

Usamos:

- 1 (0 V) → 1 lógico
- 0 (5 V) → 0 lógico

Entonces:

```
yaml

Voltajes recibidos: 1 0 1 1 0 0 1 1 1 0 1 1
                   ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
Bits reales:       1 0 1 1 0 0 1 1 1 0 1 1
```

### ✓ Paso 2: Separar en [START][DATA][STOP]



Protocolo típico RS-232:

```
css

[Start bit][8 Data bits LSB-first][Stop bit]
```

Usamos los primeros 10 bits (los otros 2 podrían ser otro carácter o ruido):

- **Start bit: 1** → Esto indica **1 lógico**, lo cual es **inválido** en RS-232 (debería ser 0 lógico).

👉 Entonces **no empieza en un start bit válido**.

Vamos a **correr 1 bit a la derecha** para intentar encontrar una trama válida.

### ✅ Reintentar con la siguiente secuencia desplazada 1 bit

```
pgsql

Bits reales ahora:  0 1 1 0 0 1 1 1 0 1 1
                    ↑
                Start bit = 0 (válido)
```

Tomamos:

- **Start: 0** → correcto
- **Data bits (8): 1 1 0 0 1 1 1 0**
- **Stop bit: 1** → correcto

### ✅ Paso 3: Reordenar los 8 bits (LSB first)

Bits de datos recibidos (en orden transmitido, LSB first):

```
1 1 0 0 1 1 1 0
```

Reordenamos a MSB first:

```
0 1 1 1 0 0 1 1
```

### ✅ Paso 4: Convertir a ASCII

```
vbnet

Binario: 01110011
Decimal: 115
ASCII:  '*' 'S' '**
```

### ✅ Resultado final

- Cadena de voltajes: 101100111011
  - Bits reales (lógicos): 1 0 1 1 0 0 1 1 1 0 1 1
  - Trama válida encontrada desplazando 1 bit
  - Carácter ASCII: 's'
  - Código decimal: 115
- 

¿Deseas ahora ver cómo verificarlo en un terminal serial o representarlo en osciloscopio?