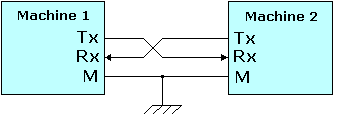
# BUS UART (RS232)

**Ejemplo N ° 1: el enlace serie (RS232)**

El enlace en serie permitirá que se comuniquen 2 dispositivos (computadoras, sensores, etc.). Hay muchos enlaces seriales (RS-232, Universal Serial Bus (USB), Serial ATA, SPI, ...). Vamos a estar interesados ​​en RS-232, que está muy extendido, así como en la versión presente en las placas de microcontroladores Arduino, Rasp Pi  
  
Para comunicarse a través del canal serie, dos dispositivos deben tener 3 cables como mínimo.  


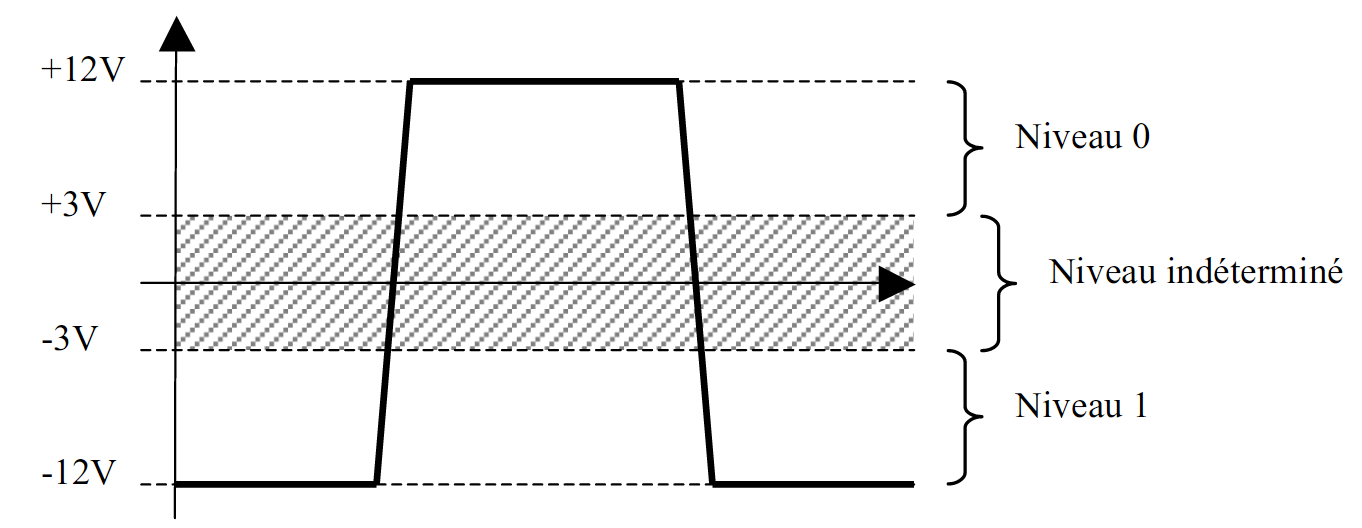
* El primer cable es la referencia eléctrica, comúnmente llamada tierra eléctrica. Esto hace posible tomar las mediciones de voltaje configurando el mismo marco de referencia. En nuestro caso, consideraremos que 0V será nuestro marco de referencia eléctrico común.
* Los otros dos cables permiten la transmisión de datos. Uno se usa para enviar datos para un transmisor, pero también se usa para recibir datos del otro transmisor. Lo mismo para el otro cable. Permite la transmisión de uno y la recepción del otro.

Este es el mínimo indispensable. La norma no prohíbe el uso de otros cables que se utilizan para el control de flujo y el manejo de errores.  
  
La velocidad de transmisión del transmisor debe ser idéntica a la velocidad de adquisición del receptor. Estas velocidades se expresan en BAUDIOS (1 baudio corresponde a 1 bit por segundo). Hay diferentes velocidades estandarizadas: 9600, 4800, 2400, 1200 ... baudios La comunicación se puede hacer en ambas direcciones (dúplex), primero transmisión, luego recepción (semidúplex) o transmisión y recepción simultáneas (full-duplex) Dado que la transmisión es del tipo asíncrono (no hay un reloj común entre el transmisor y el receptor), los bits adicionales son esenciales para la operación: bit de inicio de palabra (inicio), bit (s) de finalización palabra (stop)  
  
La principal ventaja de la comunicación en serie es la cantidad reducida de cables: la comunicación más simple se puede hacer en 3 cables (Tx, Rx y tierra).

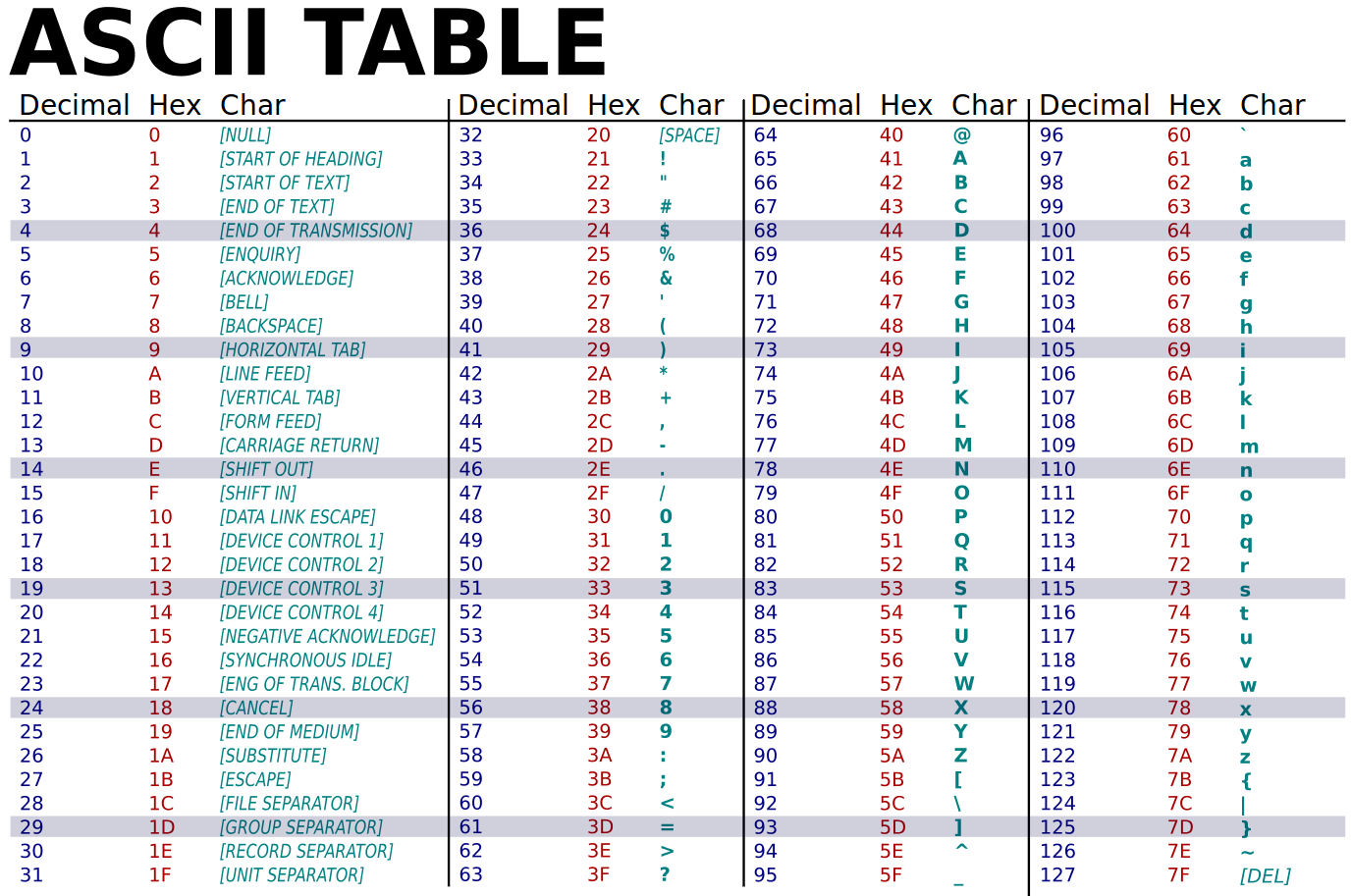
**1.1. La señal eléctrica y el protocolo**  
  
Por encima de todo, debe saber que, para comunicarse, dos dispositivos electrónicos o informáticos utilizan datos en forma de bits. Estos bits son estados lógicos (verdadero o falso, 0 o 1) que se pueden agrupar para formar conjuntos de bits. Cuando estos conjuntos de bits están formados por 8 bits, forman un byte. Hasta ahora nada nuevo!  
  
Los voltajes utilizados  
  
Estos bits son en realidad niveles de voltaje eléctrico. Y el estándar RS-232 define qué voltajes deben usarse (aquí el estándar V28 presente en las computadoras):

|  |  |  |
| --- | --- | --- |
|  | **Nivel lógico 0** | **Nivel lógico 1** |
| Voltaje eléctrico mínimo | + 3V | -3V |
| Tensión eléctrica máxima | + 25V | -25V |

Lo que da: -3v> 1 lógica> -25v y + 3v <0 lógica <+ 25v

En general obtenemos:  
  
  
Notamos que los voltajes entre + 3V y -3V se ignoran porque es en estas áreas donde se encuentran la mayoría de los parásitos. Es un medio para evitar un cierto número de errores de transmisión.  
También es un medio de verificar que el enlace físico (los cables) sea correcto: cuando no hay comunicación en el canal serie, existe lo que se llama un estado de reposo. Es decir un nivel lógico siempre presente. Este es el nivel lógico 1. O un voltaje entre -3V y -25V. Si este estado inactivo no está presente, puede haber un problema con el cableado (corte de cable, etc.).

***Los datos***  
  
Los datos que pasan a través del canal serie se transmiten en forma digital (binaria). Es decir con los niveles lógicos 0 y 1. Tomemos los datos que nos gustaría enviar, por ejemplo, la letra mayúscula "P". Debe saberse que una letra del teclado está codificada en un número de 8 bits, por lo tanto, un byte. De hecho, es de 7 bits que se codifica, pero al agregar un 0 delante de la codificación, se conserva su valor y se puede codificar la letra en 8 bits. Estos códigos se definen de acuerdo con la tabla ASCII. Por lo tanto, para cada carácter del teclado, hay una codificación de 8 bits.

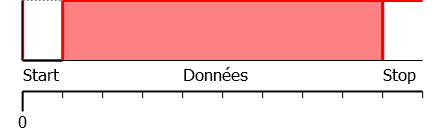
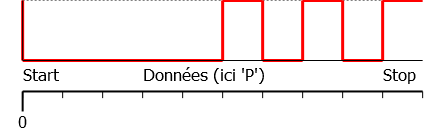
  
Al observar la tabla, caemos en la letra mayúscula "P" y vemos que su correspondencia en decimal es 80 por lo tanto en binario: 01010000.

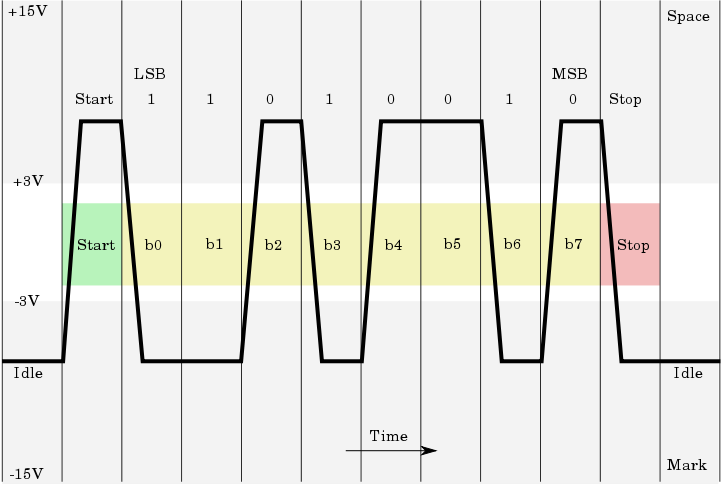
***Orden y delimitadores***

Ahora veremos cómo se transmite un byte en el canal serie enviando nuestro ejemplo, la letra "P".

Para entender sigamos la analogía de una llamada telefónica:

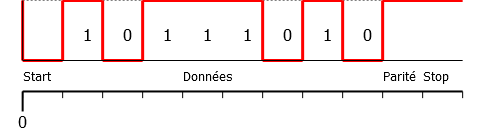
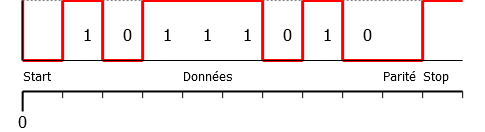
* Cuando realiza una llamada telefónica, generalmente comienza diciendo "Hola" o "Hola". Este comienzo del mensaje le permite abrir la conversación. De hecho, si recibe una llamada y nadie responde después de levantar el teléfono, la conversación no puede tener lugar. En el estándar RS-232, tendremos una apertura de la comunicación gracias a un bit de inicio. Es él quien iniciará la conversación con su interlocutor. En el estándar RS-232, este último es un estado 0.
* Luego comenzará a hablar y dará la información que desea transmitir. Serán los datos. El elemento principal de la conversación (aquí nuestra letra 'P').
* Finalmente, después de ingresar todo lo que tenía que decir, finaliza la conversación con un "Adiós". Esto termina la conversación. Por lo tanto, habrá un bit de finalización o de detención que hará lo mismo en el canal serie. En el estándar RS-232, es un estado 1.

  
  
  
Así es como funciona la comunicación en serie. Además, ¿sabes por qué se llama así al canal serie? De hecho, es porque los datos a transmitir se envían uno por uno, a la cola leu-leu. Exactamente como una conversación entre dos personas: la persona que habla no puede pronunciar varias oraciones al mismo tiempo, ni varias palabras o sonidos. Cada elemento sigue en un orden lógico. La imagen anterior resume la comunicación que acabamos de tener, solo tenemos que completarla para enviar la letra "P".  
  
Entonces está el bit de inicio, nuestra letra P y el bit de parada. Por lo que dijimos, esto daría, en orden, esto: 0 (Inicio) 01010000 (Datos) y 1 (Detener).  
  
Bueno ... eso es casi todo. Excepto que los ingenieros que inventaron este protocolo tuvieron la buena idea de transmitir los datos al revés ... En consecuencia, la respuesta correcta fue: 0 (Inicio) 00001010 (datos) 1 (Parar), es decir, 0000010101. Con un cronograma, observaríamos esto:  
  
  
  
  
  
  
  
***Un poco de vocabulario***  
  
Por lo tanto, los datos se envían al revés. Lo que debe saber es que el bit de datos que viene después del bit de inicio se denomina bit menos significativo o LSB en inglés para bit menos significativo. Es un poco como un número que tiene unidades (a la derecha), decenas, cientos, miles (a la izquierda), etc. Por ejemplo, el número 6395 tiene 5 unidades (a la derecha), 9 decenas, 3 cientos y 6 mil (a la izquierda). Podemos referirnos al bit menos significativo en binario que, por lo tanto, está a la derecha. Cuanto más avanzas, más asciendes hacia ... el bit más significativo o MSB en inglés para el bit más significativo. Y a medida que los datos se envían hacia atrás en el enlace serie, tendremos el bit menos significativo justo después del inicio, así que a la izquierda y el bit más significativo a la derecha.  
Por lo tanto, es esencial saber dónde está el bit menos significativo para poder leer los datos en la ubicación. De lo contrario, terminamos con datos incorrectos.  
  
Para reagrupar un poco todo lo que vimos en el protocolo estándar RS-232, aquí hay una imagen de un marco:

  
Deberías poder averiguar qué personaje se envía en este marco ... ¿entonces? Sugerencia: es una carta ... Leemos los niveles lógicos de izquierda a derecha, es decir 11010010; luego los devolvemos, es decir, 01001011 finalmente nos comparamos con la tabla ASCII y encontramos la letra mayúscula "K". Tenga cuidado con los voltajes negativos que corresponden a la lógica 1 y los voltajes positivos a la lógica 0.  
  
  
***Velocidad***  
El estándar RS-232 define la velocidad a la que se envían los datos. Se expresan en bits por segundo (bit / s). Recomienda velocidades inferiores a 20,000 bits / s. Excepto que en la práctica, es muy común usar velocidades de bits más altas de hasta 115.200 bits / s. Cuando vamos a utilizar el canal en serie, definiremos la velocidad a la que se transfieren los datos. Esta velocidad depende de varias restricciones que son: la longitud del cable utilizado para conectar los dos interlocutores y la velocidad a la que los dos interlocutores pueden entenderse entre sí. Esto es lo que dice el estándar RS-232:

|  |  |
| --- | --- |
| **Velocidad (bit / s) - (baudios)** | **Longitud (m)** |
| 2,400 | 60 |
| 4.800 | 30 |
| 9.600 | 15 |
| 19,200 | 7.6 |
| 38,400 | 3.7 |
| 56,000 | 2.6 |

Cuanto más corto es el cable, mayor es la velocidad de flujo porque se reduce la tensión y se reduce el riesgo de interferencia.

***Manejo de errores***  
  
A pesar de los voltajes impuestos por el estándar, sucede que hay otros parásitos y que ocurren errores de transmisión. Para limitar este riesgo, hay una solución. Consiste en agregar un bit de control llamado bit de paridad. Justo antes del bit de parada, agregaremos un bit que será par o impar. Entonces, respectivamente, un 0 o un 1. La configuración del enlace le permitirá elegir una paridad par o impar.  
  
Si se elige una paridad par, entonces el número de niveles lógicos 1 en los datos más el bit de paridad debe dar un número par. Entonces, en el caso a continuación, donde hay 5 niveles lógicos 1 sin el bit de paridad, este último tendrá que tomar un nivel lógico 1 para que el número de 1 en la señal sea par. O 6 en total:  
  
  
  
  
En el caso en el que uno elegiría una paridad impar, entonces, en la misma señal donde hay 5 niveles lógicos 1, el bit de paridad tendrá que tomar el valor que mantiene un número impar de 1 en la señal. O un bit de paridad igual a 0 en nuestro caso:  
  
  
  
  
después, el receptor verificará si el número de niveles lógicos 1 es realmente igual a lo que indica el bit de paridad. En caso de que se haya producido un error de transmisión, dependerá del receptor tratar el problema y pedirle a la otra parte que lo repita.

***El* conector de*enlace serie RS232***  
El conector utilizado por el estándar RS232 es del tipo DB9:

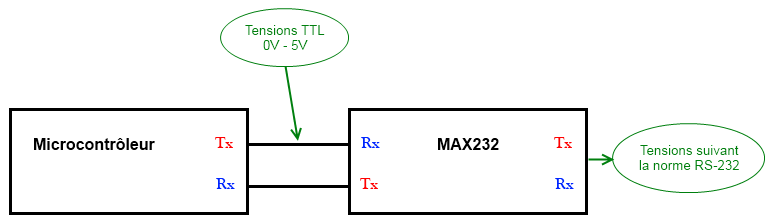
|  |  |  |
| --- | --- | --- |
|  |  |  |
| Tenga en cuenta la presencia de al menos los 3 cables obligatorios: RxD (recibir datos), TxD y masa en TIERRA, pero también otras posibilidades de cables que permiten una gestión más eficiente del enlace serie (no discutiremos su uso en este curso ) | |
|  |  |  |

***El enlace en serie de la placa del microcontrolador Arduino.***  
  
Tome el siguiente ejemplo: el objetivo es conectar dos microcontroladores Arduino para que puedan intercambiar datos.  
  
El voltaje de los microcontroladores:

|  |  |
| --- | --- |
|  | **voltaje** |
| Nivel lógico 0 | 0V |
| Nivel lógico 1 | + 5V |

Al contrario de lo que impone el estándar RS-232, los microcontroladores no pueden usar voltajes negativos. De repente, usan el único voltaje que pueden usar, a saber, 0V y + 5V. Por lo tanto, hay algunos pequeños cambios en la transmisión en serie. Un nivel lógico 0 corresponde a un voltaje de 0V y un nivel lógico 1 corresponde a un voltaje de + 5V. Afortunadamente, como casi todos los microcontroladores usan este estándar, no hay problema en conectar dos microcontroladores entre sí, incluso si no son de la misma marca. Este estándar se llama UART para transmisor receptor asíncrono universal en lugar de RS232. Excepto por los voltajes eléctricos y el conector, ¡es lo mismo!

***Cruce de datos***  
  
Solo deberemos tener cuidado al cruzar los cables. El Tx (pin de transmisión) de un microcontrolador está conectado al Rx (pin de recepción) del otro microcontrolador. Y a la inversa, el Tx del otro al Rx del primero. Y, por supuesto, masa a masa para hacer una referencia común.



El circuito electrónico MAX232 adaptará los voltajes del enlace serie del Microcontrolador al enlace serie (RS232) de la computadora.