Comunicación Serial

1. Conceptos y Términos Generales

¿Qué es la comunicación serial?

La comunicación serial es un protocolo muy común (no hay que confundirlo con el Bus Serial de Comunicación, o USB) para comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora. La mayoría de las computadoras incluyen uno o dos puertos seriales RS-232. La comunicación serial es también un protocolo común utilizado por varios dispositivos para instrumentación; existen varios dispositivos compatibles con GPIB (*General-Purpose Interface Bus/Bus de interfaz de propósito general*) que incluyen un puerto RS-232. Además, la comunicación serial puede ser utilizada para adquisición de datos si se usa en conjunto con un dispositivo remoto de muestreo.

El concepto de comunicación serial es sencillo. El puerto serial envía y recibe bytes de información un bit a la vez. Aún y cuando esto es más lento que la comunicación en paralelo, que permite la transmisión de un byte completo por vez, este método de comunicación es más sencillo y puede alcanzar mayores distancias. Por ejemplo, la especificación *IEEE 488* para la comunicación en paralelo determina que el largo del cable para el equipo no puede ser mayor a 20 metros, por el otro lado, utilizando comunicación serial el largo del cable puede llegar a los 1200 metros.

Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII. Para realizar la comunicación se utilizan 3 líneas de transmisión: (1) **Tierra** (o referencia), (2) **Transmitir**, (3) **Recibir**. Debido a que la transmisión es asincrónica, es posible enviar datos por una línea mientras se reciben datos por otra. Existen otras líneas disponibles para realizar *handshaking*, o intercambio de pulsos de sincronización, pero no son requeridas. Las características más importantes de la comunicación serial son la **velocidad de transmisión**, los **bits de datos**, los **bits de parada**, y la **paridad**. Para que dos puertos se puedan comunicar, es necesario que las características sean iguales.

a. **Velocidad de transmisión *(baud rate)*:** Indica el número de bits por segundo que se transfieren, y se mide en baudios (*bauds*). Por ejemplo, 300 baudios representa 300 bits por segundo. Cuando se hace referencia a los ciclos de reloj se está hablando de la velocidad de transmisión. Por ejemplo, si el protocolo hace una llamada a 4800 ciclos de reloj, entonces el reloj está corriendo a 4800 Hz, lo que significa que el puerto serial está muestreando las líneas de transmisión a 4800 Hz. Las velocidades de transmisión, en bits por segundo, más comunes para las líneas telefónicas son de 14400, 28800, y 33600. Es posible tener velocidades más altas, pero se reduciría la distancia máxima posible entre los dispositivos. Las altas velocidades se utilizan cuando los dispositivos se encuentran uno junto al otro, como es el caso de dispositivos GPIB.
b. **Bits de datos:** Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits. Las cantidades más comunes de bits por paquete son 5, 7 y 8 bits. El número de bits que se envía depende en el tipo de información que se transfiere. Por ejemplo, el ASCII estándar tiene un rango de 0 a 127, es decir, utiliza 7 bits; para ASCII extendido es de 0 a 255, lo que utiliza 8 bits. Si el tipo de datos que se está transfiriendo es texto simple (ASCII estándar), entonces es suficiente con utilizar 7 bits por paquete para la comunicación. Un

paquete se refiere a una transferencia de byte, incluyendo los bits de inicio/parada, bits de datos, y paridad. Debido a que el número actual de bits depende en el protocolo que se seleccione, el término paquete se usar para referirse a todos los casos.

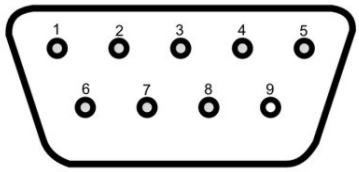
c. **Bits de parada:** Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj, es posible que los dos dispositivos no estén sincronizados. Por lo tanto, los bits de parada no sólo indican el fin de la transmisión sino además dan un margen de tolerancia para esa diferencia de los relojes. Mientras más bits de parada se usen, mayor será la tolerancia a la sincronía de los relojes, sin embargo la transmisión será más lenta.
d. **Paridad:** Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: **par**, **impar**, **marcada** y **espaciada**. La opción de no usar paridad alguna también está disponible. Para paridad par e impar, el puerto serial fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico. Por ejemplo, si la información a transmitir es 011 y la paridad es par, el bit de paridad sería 0 para mantener el número de bits en estado alto lógico como par. Si la paridad seleccionada fuera impar, entonces el bit de paridad sería 1, para tener 3 bits en estado alto lógico. La paridad marcada y espaciada en realidad no verifican el estado de los bits de datos; simplemente fija el bit de paridad en estado lógico alto para la marcada, y en estado lógico bajo para la espaciada. Esto permite al dispositivo receptor conocer de antemano el estado de un bit, lo que serviría para determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos, o si los relojes de los dispositivos no están sincronizados.

¿Qué es RS-232?

RS-232 (Estándar ANSI/EIA-232) es el conector serial hallado en las PCs IBM y compatibles. Es utilizado para una gran variedad de propósitos, como conectar un ratón, impresora o módem, así como instrumentación industrial. Gracias a las mejoras que se han ido desarrollando en las líneas de transmisión y en los cables, existen aplicaciones en las que se aumenta el desempeño de RS-232 en lo que respecta a la distancia y velocidad del estándar. RS-232 está limitado a comunicaciones de punto a punto entre los dispositivos y el puerto serial de la computadora. El hardware de RS-232 se puede utilizar para comunicaciones seriales en distancias de hasta 50 pies.

Pines del conector DB-9

Pin	Nombre	Dirección	Descripción
1	**CD**	**?**	Carrier Detect
2	**RXD**	**?**	Receive Data
3	**TXD**	**?**	Transmit Data
4	**DTR**	**?**	Data Terminal Ready
5	**GND**		System Ground
6	**DSR**	**?**	Data Set Ready
7	**RTS**	**?**	Request to Send
8	**CTS**	**?**	Clear to Send
9	**RI**	**?**	Ring Indicator
Dirección es del DTE (Computador) al DCE (Módem).			



Conector externo de la computadora y expuesto del cable.
Funciones de los pines en RS-232:

- Datos: **TXD** (pin 3), **RXD** (pin 2)
- Handshake: **RTS** (pin 7), **CTS** (pin 8), **DSR** (pin 6), **DCD** (pin 1), **DTR** (pin 4)
- Tierra: GND (pin 5)
- Otros: RI (pin 9)

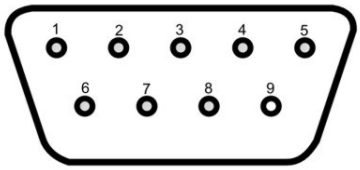
¿Qué es RS-422?

RS-422 (Estándar EIA RS-422-A) es el conector serial utilizado en las computadoras Apple de Macintosh. RS-422 usa señales eléctricas diferenciales, en comparación con señales referenciadas a tierra como en RS-232. La transmisión diferencial, que utiliza dos líneas para transmitir y recibir, tiene la ventaja que es más inmune al ruido y puede lograr mayores distancias que RS-232. La inmunidad al ruido y la distancia son dos puntos clave para ambientes y aplicaciones industriales.

¿Qué es RS-485?

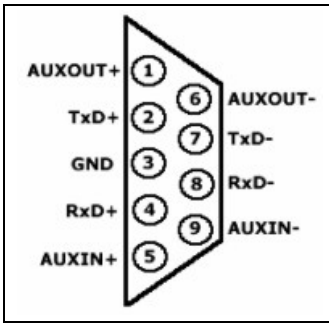
RS-485 (Estándar EIA-485) es una mejora sobre RS-422 ya que incrementa el número de dispositivos que se pueden conectar (de 10 a 32) y define las características necesarias para asegurar los valores adecuados de voltaje cuando se tiene la carga máxima. Gracias a esta capacidad, es posible crear redes de dispositivos conectados a un solo puerto RS-485. Esta capacidad, y la gran inmunidad al ruido, hacen que este tipo de transmisión serial sea la elección de muchas aplicaciones industriales que necesitan dispositivos distribuidos en red conectados a una PC u otro controlador para el conjunto HMI (*Human Machina Interface*), u otras operaciones. RS-485 es un conjunto que cubre RS-422, por lo que todos los dispositivos que se comunican usando RS-422 pueden ser controlados por RS-485. El hardware de RS-485 se puede utilizar en comunicaciones seriales de distancias de hasta 4000 pies de cable (1219.2 mt).

Pines del conector DB-9

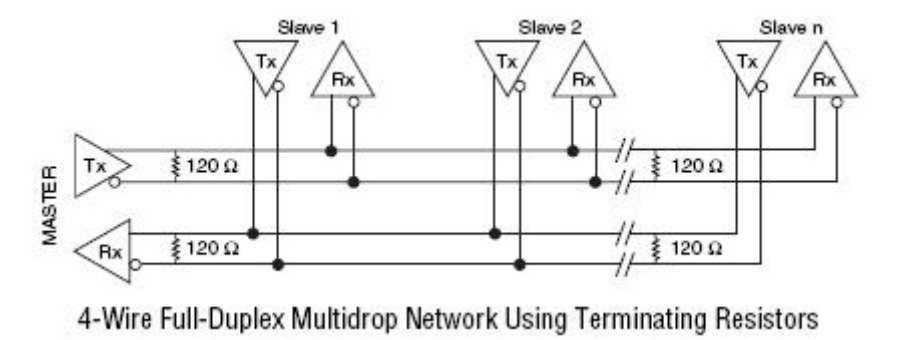


- Conector externo de la computadora y expuesto del cable.
- Funciones de los pines en RS-485 y RS-422:
- Datos: **TXD+** (pin 8), **TXD-** (pin 9), **RXD+** (pin 4), **RXD-** (pin 5)
- Handshake: **RTS+** (pin 3), **RTS-** (pin 7), **CTS+** (pin 2), **CTS**- (pin 6)
- Tierra: **GND** (pin 1)

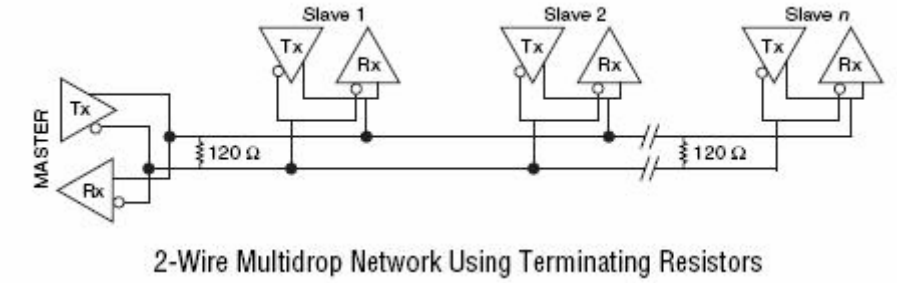
RS 422/485 Female Pinout



Pin	Descripción
1	Auxiliary Output **+**
2	Data Output **+**
3	Signal Ground
4	Data Input **+**
5	Auxiliary Input **+**
6	Auxiliary Output **-**
7	Data Output **-**
8	Data Input **-**
9	Auxiliary Input **-**



4-Wire Full-Duplex Multidrop Network Using Terminating Resistors



2-Wire Multidrop Network Using Terminating Resistors

	RS232	RS422	RS485
Cabling	single ended	single ended	Multi- drop
Number of Devices	1 transmit 1 receive	5 transmitters 10 receivers	32 transmitters 32 receivers
Communication Mode	full duplex	full duplex half duplex	full duplex half duplex
Max. Distance	50 feet at 19.2 Kbps	4000 feet at 100 Kbps	4000 feet at 100 Kbps
Max. Data Rate	19.2 Kbps for 50 feet	10 MBPS for 50 feet	10 MBPS for 50 feet
Signalling	unbalanced	balanced	balanced
Mark (data 1)	-5 V min. -15 V max.	2 V min. (B>A) 6 V max. (B>A)	1.5 V min. (B>A) 5 V max. (B>A)
Space (data 0)	5 V min 15 V max.	2 V min. (A>B) 6 V max. (A>B)	1.5 V min. (A>B) 5 V max. (A>B)
Input Level Min.	+/- 3 V	0.2 V difference	0.2 V difference
Output Current	500 mA	150 mA	250 mA

¿Qué es *handshaking* o intercambio de pulsos de sincronización?

El método de comunicación usado por RS-232 requiere de una conexión muy simple, utilizando sólo tres líneas: Tx, Rx, y GND. Sin embargo, para que los datos puedan ser

transmitidos correctamente ambos extremos deben estar sincronizados a la misma velocidad. Aún y cuando este método es más que suficiente para la mayoría de las aplicaciones, es limitado en su respuesta a posibles problemas que puedan surgir durante la comunicación; por ejemplo, si el receptor se comienza a sobrecargar de información. Es en estos casos cuando el intercambio de pulsos de sincronización, o *handshaking*, es útil. En esta sección se describirán brevemente las tres formas más populares de *handshaking* con RS-232: handshaking por software, handshaking por hardware, y XModem.

Handshaking por software: Esta forma de sincronización utiliza bytes de datos como caracteres de control, de manera similar a como GPIB utiliza las cadenas de caracteres como comandos. Las líneas necesarias para la comunicación siguen siendo Tx, Rx, y GND, ya que los caracteres de control se envían a través de las líneas de transmisión como si fueran datos. La función SetXMode permite al usuario habilitar o deshabilitar el uso de dos caracteres de control: XON y XOFF. Estos caracteres son enviados por el receptor para pausar al transmisor durante la comunicación.

A manera de ejemplo, asúmase que el transmisor comienza a enviar datos a alta velocidad. Durante la transmisión, el receptor se da cuenta que el búfer de entrada se está llenando debido a que el CPU está ocupado con otras tareas. Para pausar temporalmente la transmisión, el receptor envía XOFF (cuyo valor es típicamente 19, o 13 hex) hasta que el búfer se vacíe. Una vez que el receptor está preparado para recibir más datos envía XON (cuyo valor es típicamente 17, u 11 hex) para continuar la comunicación. LabWindows[1] enviará un XOFF cuando el búfer de entrada se encuentre a la mitad de su capacidad. Además, en caso que la transmisión inicial de XOFF haya fallado, LabWindows enviará de nuevo un XOFF cuando el búfer alcance un 75% y 90% de su capacidad. Para que funcione correctamente, es necesario que el transmisor esté utilizando el mismo protocolo.

La mayor desventaja de este método es además lo más importante a considerar: los números decimales 17 y 19 son ahora los límites para la transmisión. Cuando se transmite en ASCII, esto no importa mucho ya que estos valores no representan carácter alguno. Sin embargo, si la transmisión de datos es en binario, lo más probable es que estos valores sean transmitidos como datos regulares y falle la comunicación.

Handshaking por hardware: El segundo método de *handshaking* utiliza líneas de hardware. De manera similar a las líneas Tx y Rx, las líneas RTS/CTS y DTR/DSR trabajan de manera conjunta siendo un par la entrada y el otro par la salida. El primer par de líneas es RTS (por sus siglas en inglés, *Request to Send*) y CTS (*Clear to Send*). Cuando el receptor está listo para recibir datos, cambia la línea RTS a estado alto; este valor será leído por el transmisor en la línea CTS, indicando que está libre para enviar datos. El siguiente par de líneas es DTR (por sus siglas en inglés, *Data Terminal Ready*) y DSR (*Data Set Ready*). Estas líneas se utilizan principalmente para comunicación por módem, permiten al puerto serial y módem indicarse mutuamente su estado. Por ejemplo, cuando el módem se encuentra preparado para que la PC envíe datos, cambia la línea DTR a estado alto indicando que se ha realizado una conexión por la línea de teléfono. Este valor se lee a través de la línea DSR y la PC comienza a enviar datos. Como regla general, las líneas DTR/DSR se utilizan para indicar que el sistema está listo para la comunicación, mientras que las líneas RTS/CTS se utilizan para paquetes individuales de datos.

En LabWindows, la función SetCTSMode habilita o deshabilita el uso de *handshaking* por hardware. Si el modo CTS está habilitado, LabWindows aplica las siguientes reglas:

[1] La versión 5 de LabWindows podrá descargarse de la página: http://www.rootshell.be/~wcruzy/cd

- **Cuando la PC envía datos**: La librería de RS-232 debe de detectar que la línea CTS se encuentra en estado alto antes de enviar datos.
- **Cuando la PC recibe datos**: Si el puerto está abierto y el búfer de entrada puede contener más datos, la librería envía a RTS y DTR a estado alto.
  - Si el búfer de entrada está al 90% de su capacidad, la librería manda a estado bajo RTS pero mantiene DTR en alto.
  - Si el búfer de entrada está casi vacío, la librería manda a estado alto RTS y mantiene DTR en alto.
  - Si el puerto se cierra, la librería manda a estado bajo a RTS y DTR.

Handshaking por XModem: El último modo de *handshaking* es el protocolo de transmisión de archivos XModem. Este protocolo es muy común en comunicación por módem. Aún y cuando es más utilizado para comunicación por módem, el protocolo XModem puede ser utilizado directamente entre otros dispositivos. En LabWindows, la implementación de XModem se mantiene oculta para el usuario. Mientras la PC se conecte a otro dispositivo que utilice el protocolo XModem, se pueden utilizan las funciones de LabWindows para transferir datos de un lado a otro. Estas funciones son XModemConfig, XModemSend, y XModemReceive.

XModem utiliza un protocolo basado en los siguientes parámetros: *start_of_data*, *end_of_trans*, *neg_ack*, *ack*, *wait_delay*, *start_delay*, *max_tries*, *packet_size*. Estos parámetros deben de ser comunes en ambos lados de la comunicación, y el estándar XModem contiene la definición estándar de éstos; sin embargo, se pueden modificar utilizando la función XModemConfig de LabWindows para cumplir cualquier otro requerimiento. Los parámetros en XModem funcionan de la siguiente manera: el receptor envía el carácter "*neg_ack*". Esto indica al transmisor que ya está listo para recibir datos. El receptor continuará enviado el carácter "*neg_ack*" en intervalos de tiempo de duración de "*start_delay*" hasta que iguale la cuenta de "*max_tries*" o reciba "*start_of_data*" del transmisor. Si el receptor intenta comunicarse con el transmisor la misma cantidad de veces que "*max_tries*", informará al usuario que no fue posible comunicarse con el transmisor. Si el receptor sí recibe el "*start_of_data*" del transmisor, leerá el paquete de información que sigue. Este paquete contiene el número de paquete, el complemento del número de paquete para fines de verificación de errores, el paquete actual de datos con una cantidad de bytes igual a "*packet_size*", y un checksum para más verificación de errores. Después de recibir el paquete, el receptor mandará llamar el "*wait_delay*", y luego enviará el "*ack*" al transmisor. Si el transmisor no recibe el "*ack*", intentará de reenviar el paquete de datos una cantidad de veces igual a "*max_tries*" o hasta que reciba el "*ack*". Si nunca recibe el "*ack*", informará al usuario que hubo un fallo al momento de querer transferir el archivo.

Los datos deben de ser enviados en paquetes con una cantidad de bytes igual a "*packet_size*". Debido a esto, cuando se está enviando el último paquete y no se tiene la cantidad suficiente de información válida para llenarlo, el protocolo llenará el paquete con el carácter ASCII Null (0). Esto puede causar que el archivo recibido sea más grande que el original. Es importante recordar que no hay que usar XON/XOFF con el protocolo XModem, ya que el número de paquete durante la transferencia por XModem se incrementará conforme se envían los caracteres XON/XOFF, lo que puede causar una falla en la comunicación.

2. Prueba Loopback para verificar la operación de su dispositivo

Existen tres programas para verificar la operación del puerto serial: LabVIEW, HyperTerminal y LabWindows/CVI. Estas opciones ejecutan una prueba Loopback en el puerto serial al poner en corto circuito los pines de Envío y Recibo de información en el cable conectado a ese puerto. El primer paso describe el proceso para conectar en corto circuito los pines; este es el primer paso para los tres programas mencionados. Los procedimientos de HyperTerminal y LabWindows/CVI están incluidos al final del documento.

a. **Conecte un cable a su puerto serial**. El puerto más común es RS-232 con un conector de 9-pin ó 25-pin (DB-9 ó DB-25). En el cable, conecte las líneas 2 y 3 una con otra. Esto conectará la línea de Envío que viene de la computadora a la línea de Recibo que va a la computadora. Cuando los tenga conectados, la fila de arriba de su cable DB-9 debe verse de la manera siguiente [1 2=3 4 5].

   En un puerto RS-485, los voltajes usan señales eléctricas diferenciales. Por lo tanto, debe conectar TXD+ con RXD+ y TXD- con RXD- (en el conector DB-9 esto equivale a conectar el pin 4 con el 8 y el pin 5 con el 9; en un conector Combicon conecte el pin 1 con el 5 y el pin 2 con el 4; en un conector modular jack conecte el pin 2 con el 6 y el pin 3 con el 7). Asegúrese que su software esté configurado en el modo de 4-alambres o 4-wires.

   Nota: Refiérase al Apéndice A del "Manual de Usuarios de Windows para Hardware y Software Serial" (archivo NISerial en http://www.rootshell.be/~wcruzy/cd) para más información acerca de los diferentes tipos de conectores.

b. Para **HyperTerminal**, siga las instrucciones siguientes.

La siguiente prueba escribe y lee desde el mismo puerto serial (asegúrese que tiene puesta la configuración loopback)

1. Seleccione **Start>>Programs>>Accessories>>Hyperterminal**.
2. Doble clic en el icono de **Hypertrm**
3. Si le pregunta si desea instalar el módem, responda **No**.
4. De un nombre a su sesión
5. En el menú Pop-Up, escoja **direct to COMx** donde x es el numero del puerto COM que está probando
6. Ponga el control de flujo a **None**. Puede dejar los otros parámetros tal como están, o incrementar la velocidad si lo desea
7. Seleccione **File » Properties** en la barra de menú, seleccione la pestaña **Settings**, y pulse el botón **ASCII Setup**.
8. Verifique la opción para **Echo typed characters locally**.
9. Comienze a escribir. Si ve doble caracteres (esto es. Ve dos caracteres por cada character que tipea), el puerto serial está trabajando correctamente

c. Para **LabWindows/CVI**, siga las instrucciones siguientes.

La siguiente prueba escribe y lee desde el mismo puerto serial.

Abrir el ejemplo en LabWindows/CVI que se encuentra en **x:\sample\rs232\serial.prj**, donde **x** es el directorio CVI. Conectar su cable modificado al puerto serial que desea probar, seleccione ese puerto en el ejemplo, y escribir entonces para leer en ese puerto. Si la prueba es un éxito, su puerto fue configurado correctamente

3. Arquitectura de Software para Instrumentos Virtuales (VISA)

VISA es una librería de interfaz simple para controlar VXI, GPIB, RS-232, y otros tipos de instrumentos. VISA es el estándar utilizado por la Alianza de Sistemas de VXI plug & play, que incluye más de 35 de las compañías más grandes en la industria de instrumentación. El estándar VISA unifica la industria para hacer software que pueda ser interpretado y reusado por más tiempo, sin importar el tipo de operación de su instrumento.

4. Localización de Averías Avanzadas para Data del Puerto COM

Usuarios avanzados que desean analizar los bits exactos que son transferidos a través del puerto serial para detectar diferencias entre programas, pueden usar un programa llamado **Portmon**. Portmon es provisto por Sysinternals[2], quienes mantiene una página web con utilidades avanzadas, información técnica, y código relacionado con Windows 9x y Windows NT/2K internos.

[2] http://www.microsoft.com/technet/sysinternals/utilities/portmon.mspx