Diseño de

Sistemas Informáticos Industriales

Apuntes de

Comunicación serie.
RS-232 y RS-485

Ingeniería Técnica Industrial (Electrónico)
Departamento de Informática de Sistemas y Computadores
Escuela Técnica Superior de Ingeniería del Diseño

4-comunicaciones_serie_r3.doc

Contenido

4. Comunicaciones industriales. Comunicación serie (nivel físico). 3
4.1 La conexión serie 3
4.1.1 Serie us paralelo 3
4.1.2 Transmisión asíncrona y síncrona 3
4.2 Conexión serie RS-232 5
4.2.1 Especificaciones 5
4.2.2 Control de flujo 8
4.2.3 Conexión estándar entre un computador y un modem 8
4.2.4 Conexión null-modem 9
4.2.5 Uso del RS-232 del PC. Hardware y software 10
4.2.6 Aplicación general a dispositivos industriales 14
4.2.7 Ejemplo: Visor de peso G6 18
4.3 Ejemplo: VISOR DE PESO IE-21 20
4.4 Conexión serie RS-485 22
4.4.1 Especificaciones y características 22
4.4.2 Uso del RS-485 con el PC 24
4.4.3 Aplicación a un producto industrial:los módulos NuDAM de ADLink 26
4.4.3.1 Introducción 26
4.4.3.2 Configuración de la red. 27
4.4.3.3 Módulo conversor RS-232 a RS-485 NuDAM-6520. 28
4.4.3.4 Módulo de E/S digital. NuDAM-6050 30
4.5 Bibliografía 35

4 COMUNICACIONES INDUSTRIALES. COMUNICACIÓN SERIE (NIVEL FÍSICO)

4.1 LA CONEXIÓN SERIE

El objetivo de este tema es saber qué es una conexión serie y poder aprovechar las normas RS-232 y RS-485 que emplean muchos sistemas industriales desde el punto de vista del conexionado físico y de su aprovechamiento desde una aplicación para computador.

Las normas RS-232 y RS-485 aquí descritas corresponde al nivel físico (1) de referencia del estándar OSI. Sobre dichas normas se pueden construir nuevas capas que aporten otras funcionalidades.

4.1.1 SERIE US PARALELO

La transmisión serie de información digital consiste en el envío/recepción de secuencias de bits uno a uno. Como contraposición, la transmisión paralela permite el envío/recepción de más de un bit a la vez.

Como características destacables de la transmisión serie están:

- Más barata, al necesitar menos conductores.
- No sufre grandes problemas de autoinducción entre líneas, pues usa pocas líneas.
- Menor velocidad de transmisión que la paralela.

Estas características la hacen adecuada para comunicaciones a distancias superiores a unos metros, donde el coste del cableado o su tamaño son factores a considerar.

Las transmisiones en paralelo son las más adecuadas para alcanzar velocidades de transferencia altas al nivel de placas de circuito impreso o de circuito integrado.

4.1.2 TRANSMISIÓN ASÍNCRONA Y SÍNCRONA

Supóngase el siguiente escenario en una transmisión serie:

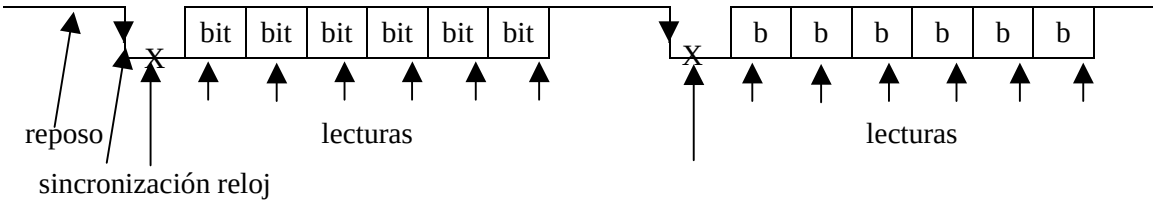
- Imagínese que un emisor emite una secuencia de bits a 10 millones de bits por segundo

  10 Mbps, --> un bit dura $1/10^7 = 0´1\mu s$.
- Un receptor, para recibirlos debe muestrear la línea a la misma velocidad, es decir, cada $0´1\mu s$.
- Cada uno basa la medida del tiempo en su propio reloj, que pueden no estar sincronizados con precisión.

- Si, por ejemplo, la diferencia de velocidad entre uno y el otro es del 1%, entonces el primer muestreo estará desplazado 0´001µs.
- Tras 200 muestras, los relojes estarán desplazados 200*0´001µs=0´2µs por lo que no se estará recogiendo adecuadamente la información.

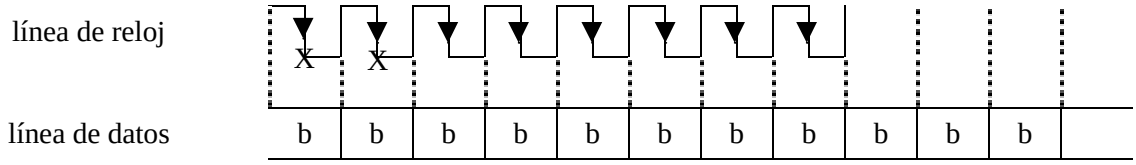
Para resolver el problema ilustrado es necesario introducir técnicas de ***sincronización*** que permiten ponerse de acuerdo en las temporizaciones. Hay dos grupos de soluciones: utilizar transmisión ***asíncrona*** o utilizar transmisión ***síncrona***.

La ***transmisión asíncrona*** intenta evitar el problema enviando secuencias de bits que no sean muy largas y sincronizando los relojes al principio de cada secuencia.



En la ***transmisión síncrona*** los relojes se mantienen sincronizados usando una línea de reloj o una codificación de datos "*autorreloj*".

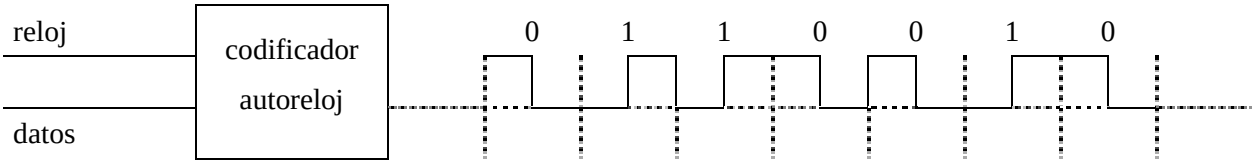
Usando línea de reloj,



Por ejemplo el bus I2C utiliza dos líneas, una para datos y otra para reloj

Usando una codificación *autorreloj*

0110010



Por ejemplo, la codificación Manchester empleada en el protocolo CSMA/CD usado en redes tipo Ethernet.

La forma en que se sincroniza la información en ambos tipos de transmisión da lugar a los conceptos de sincronismo de bit, de carácter y de bloque. (?ampliar)

4.2 CONEXIÓN SERIE RS-232

EIA RS-232 es un conjunto de normas del nivel físico de la terminología OSI para la conexión DTE-DCE con velocidades de hasta 20 Kbits/s.

Inicialmente se desarrolló para conectar un ordenador con un módem, pero su aceptación ha hecho que se aplique a infinidad de equipos industriales y domésticos.

Este tipo de interfaz física tiende a desaparecer con la aparición de nuevos interfaces serie de alta velocidad (USB, FireWire, etc.), pero los principios aquí descritos son igualmente aplicables.

FICAR FOTO D'EQUIP

Permite transmisión serie asíncrona y síncrona entre equipos. El modo de transmisión asíncrono es el más implantado y el que se verá aquí.

Son tres especificaciones:

- Mecánica: ISO 2110
- Funcional: norma CCITT V.24
- Eléctrica: norma CCITT V.28

4.2.1 ESPECIFICACIONES

Especificaciones mecánicas

La norma original utiliza un conector tipo DB-25 mostrado en la Figura 4-1. La Tabla 4-1 muestra el nombre de los pines más importantes para nuestro trabajo.

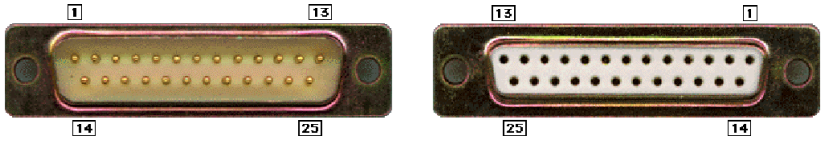


Figura 4-1. Conectores macho y hembra tipo DB-25

Pin	Señal
1	PGND (Protective Ground)
2	TXD (Transmit Data)
3	RXD (Receive Data)
4	RTS (Ready to Send)
5	CTS (Clear to Send)
6	DSR (Data Set Ready)
7	SG (Signal Ground)
8	CD (Carrier Detect)
20	DTR (Data Terminal Ready)
22	RI (Ring Indicator)

Tabla 4-1. Nombre de señal asignado a los pines del conector tipo DB-25

La desvirtuación de la especificación ha llevado a simplificar la interfaz y es muy habitual encontrarnos dispositivos con interfaz RS-232 con muchas líneas eliminadas y con un conector tipo DB-9. Hay desvirtuaciones mayores que llegan a dejar solo la linea de referencia (SG) y YXD y RXD (teléfonos móviles, camaras fotográficas digitales, GPS, etc.).

Figura 4-2. Conectores macho y hembra tipo DB-9

Pin	Señal
1	CD (Data Carrier Detect)
2	RD (Recive Data)
3	TD (Trasmit Data)
4	DTR (Data Terminal Ready)
5	SG (Signal Ground)
6	DSR (Data Set Ready)
7	RTS (Request To Send)
8	CTS (Clear To Send)
9	RI (Ring Indicator)

Tabla 4-2. Nombre de señal asignado a los pines del conector tipo DB-9

Especificaciones funcionales

Las especificaciones funcionales indican el significado y funcionalidad de cada señal. Se resumen a continuación las más interesantes para nuestro trabajo.

Señales para establecimiento de conexión:

- DTR (Data Terminal Ready), DTE operativo.
- DSR (Data Set Ready), DCE operativo.
- RI (Ring Indicator), el teléfono está sonando.

Señales para el control de flujo de datos:

- RTS (Request To Send, Ready To Send), DTE indica al DCE cuando tiene un carácter a transmitir.
- CTS (Clear To Send), DCE está listo para aceptar un carácter del DTE.
- CD (Carrier Detect), DCE indica que la conexión remota está activa.

Señales de transmisión de datos:

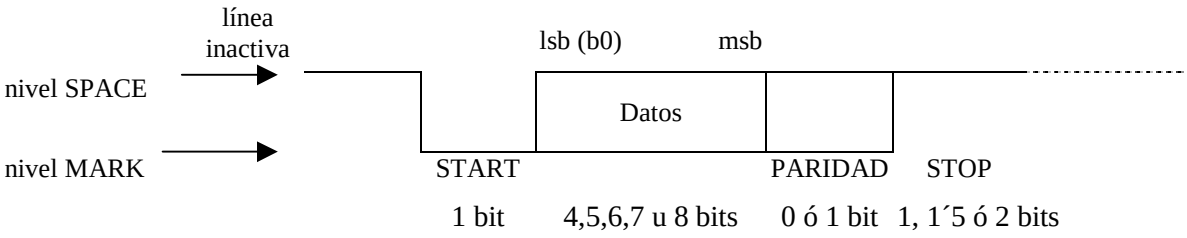
- TXD envío de datos en el DTE, recepción en el DCE.
- RXD recepción de datos en el DTE, envío en el DCE.

Otras:

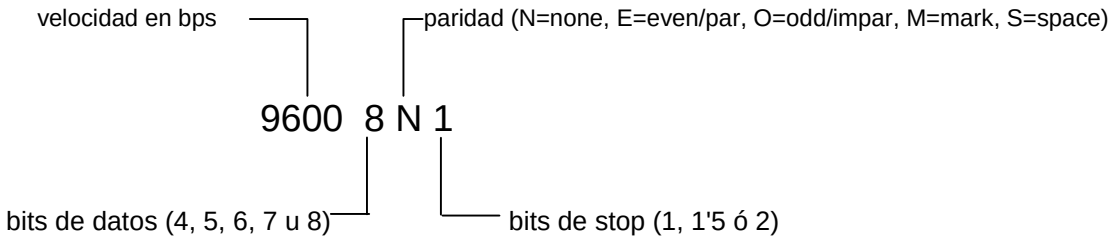
- SG (Signal Ground), referencia de las señales.

- PGND (Protective Ground), masa de protección.

El formato de los datos que circula por los pines TXD y RXD tiene el siguiente formato:



Una forma habitual de expresar la configuración de datos a usar y su velocidad consiste en una cadena de la forma:



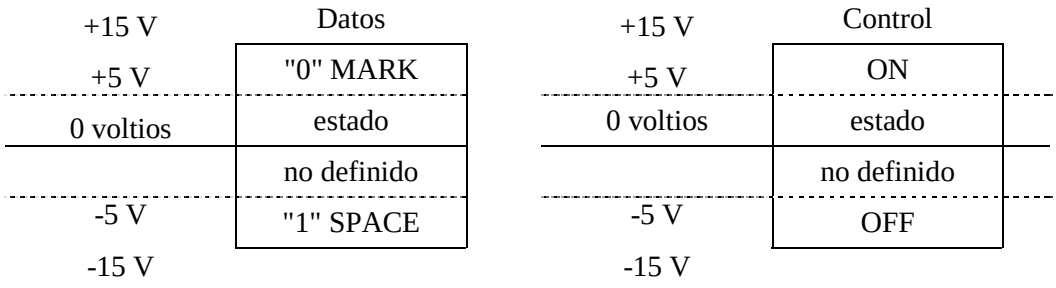
Los dos equipos que comunican deben ser configurados con el mismo formato de datos. A partir de ese momento, los datos transmitidos entre los dos extremos deberá seguir ese formato.

Algunas velocidades de comunicación habituales son 150, 300, 600, 1200, 2400, 4800, 9600, ... baudios.

Especificaciones eléctricas

Define los niveles de tensión, corriente, etc. a emplear. Destacar:

- Transmisión no balanceada (referencia de tensión común a todas las señales).
- Limitación de corriente a 0,5 A.
- Capacidad máxima de conductor de 2.500 pF
- Tiempo de flanco de señal de un mínimo del 4% del tiempo de bit
- Longitud máxima recomendada de 15 metros
- Niveles de tensión según diagrama



Actividad.

Dibuja el cronograma que resulta de enviar la cadena "Hola" por una conexión RS-232 utilizando la configuración 2400 8N1. Calcula el tiempo mínimo que se necesitará para transmitir la cadena.

Actividad.

Suponiendo que un modem transmite al DTE a una velocidad máxima de 56 kbps 8N1 por una conexión RS-232. ¿Cuántos bytes útiles se transmiten realmente por segundo?

¿Cuánto tiempo tardará en transmitir 1 Mbyte como mínimo?

4.2.2 CONTROL DE FLUJO

La interfaz serie RS-232 permite control de flujo half-duplex y full-duplex. Para ello se pueden emplear protocolos hardware, basados en el uso de las señales asignadas a los pines; o software, basados en el envío de códigos especiales que controlan el flujo.

Para los protocolos hardware, los más extendidos son:

- RTS-CTS. Por CTS se indica que se está listo para aceptar datos.
- DTR-DSR. Lo mismo, pero por DSR.

Para los software, los más extendidos son:

- XON (11h), XOFF (13h). Basado en enviar caracteres especiales para indicar que se pueden enviar o no datos.
- ENQ (?5h), ACK (?6h). Basado en solicitud y contestación.

La ventaja de los protocolos software es su compatibilidad con todos los equipos, pero a costa de un menor rendimiento.

En algunos casos no se emplea ningún tipo de control de flujo, debiéndose asegurar de que no se van a perder datos o de que las pérdidas no afectan al funcionamiento de la aplicación.

4.2.3 CONEXIÓN ESTÁNDAR ENTRE UN COMPUTADOR Y UN MODEM

El diseño original de la norma estaba pensado para conectar un computador a un módem. En la Figura 4-1 se indican las conexiones físicas que habrá entre un módem y un conector tipo DB-25 disponible en cualquier computador que cumpla la norma.

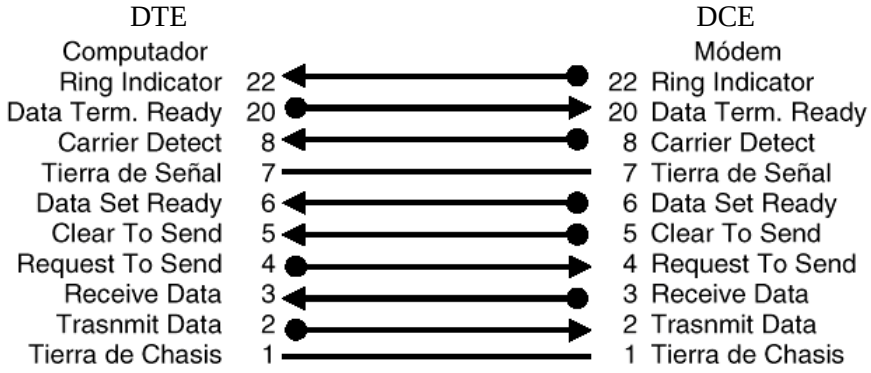


Figura 4-1. Conexió RS-232 con conector DB-25 entre un computador y un modem.

Es posible emplear un adaptador DB-9 a DB-25 siguiendo el conexionado mostrado en la Tabla 4-1.

Señal	9 Pin DTE	25 Pin DCE	Fuente DTE o DCE
CD (Data Carrier Detect)	1	8	Desde el módem
RD (Receive Data)	2	3	Desde el módem
TD (Transmitted Data)	3	2	Desde el Comp. /Term.
DTR (Data Terminal Ready)	4	20	Desde el Comp. /Term.
SG (Signal Ground)	5	7	Desde el módem
DSR (Data Set Ready)	6	6	Desde el módem
RTS (Request to Send)	7	4	Desde el Comp. /Term.
CTS (Clear to Send)	8	5	Desde el módem
RI (Ring Indicator)	9	22	Desde el módem

Tabla 4-1. Conversión DB-9 a DB-25

4.2.4 CONEXIÓN NULL-MODEM

La norma RS-232 se aplica a una gran cantidad de productos comerciales. Dichos productos pueden ser diseñados como DTE o como DCE.

En muchos casos puede ser interesante interconectar DTE-DTE o DCE-DCE. Para ello es necesario utilizar un cable que "engañe" a las partes para que parezca una conexión DTE-DCE. Para esto se utilizan los denominados cables ***null-modem*** (anulador de módem).

Un cable mínimo NULL-MODEM consiste en cruzar las señales TXD y RXD y conectar las referencias de señal, tal como se muestra en el esquema de la Figura 4-1. Dicho cable no permitirá control de flujo hardware, por lo que, si es necesario un control de flujo, este será obligatoriamente software.

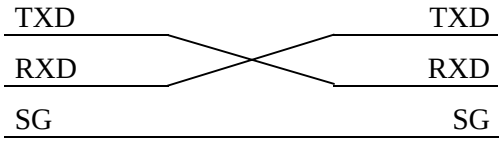


Figura 4-1. Conexión NULL-MODEM mínima

La interpretación y simplificación de la norma ha hecho que no exista un cable null-modem universal, por lo que es imposible dar un esquema universal para este tipo de cable. La Figura 4-2 muestra algunas posibilidades para conexión DB-25.

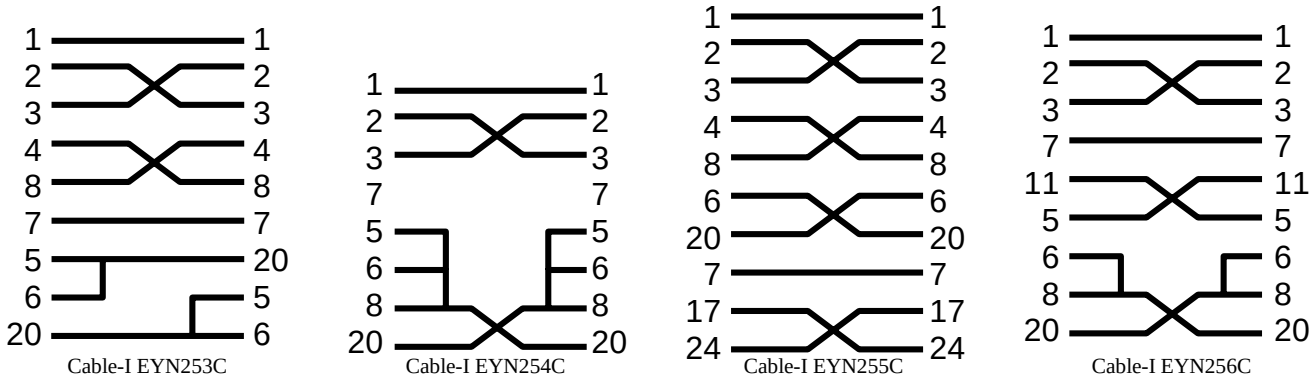


Figura 4-2. Diferentes modelos de conexión NULL-MODEM para DB-25

4.2.5 USO DEL RS-232 DEL PC. HARDWARE Y SOFTWARE.

El PC suele disponer de 1 ó 2 conexiones serie tipo RS-232. El conector exterior suele ser del tipo DB-9 macho, pero también puede ser un conector DB-25 macho.

La funcionalidad hardware interna se implementa mediante circuitería UART (??Universal Asynchronous Receiver Transmiter) compatible INTEL 8250 o INTEL 16550A. Para usar este hardware se dispone de unos registros de entrada/salida y de una línea de interrupción hardware para cada canal serie.

Para que un software pueda hacer uso directo de estos circuitos, deberá hacer directamente las lecturas/escrituras de estos puertos y aportar una rutina de servicio para la interrupción. Dicha interrupción se disparará cuando se reciban datos y cuando se termine de transmitir un dato.

Desde el punto de vista de los sistemas operativos MS-DOS y WINDOWS, los puertos serie se nombran como COM1, COM2, COM3… COMn. Estos sistemas operativos ofrecen servicios que permiten usar más fácilmente estos puertos.

Una de las formas estándar de tratar los puertos serie es como unos ficheros en los que las órdenes de escritura envían caracteres y las de lectura permiten recoger los caracteres recibidos. Tanto MS-DOS/WINDOWS como Unix proporcionan este tipo de servicio.

Con C++ Builder podemos usar la propuesta anterior o conseguir mejores característicass accediendo a los servicios del API (??Application Program Interface) de WINDOWS para uso del puerto serie. La programación de las características básicas es sencilla, pero el máximo aprovechamiento se consigue sólo con conocimientos más avanzados de programación multi-hilo. En la web de la asignatura hay ejemplos de como usar el puerto serie directamente desde el hardware y desde Linux.

Para sacar el máximo partido al puerto serie sin necesidad de conocimientos de programación avanzados se puede recurrir a un componente prefabricado. En C++ Builder no se incluye un componente para el uso de puerto serie, pero se puede recurrir al de una tercera compañía (una posibilidad es usar el componente ActiveX que incorpora Visual Basic).

Por ejemplo, el componente gratuito ComPort (http://comport.sourceforge.net) permite acceder a prácticamente todas las características del puerto serie con las ventajas que nos ofrecen la programación orientada a eventos.

Para hacerse una idea de su funcionalidad, supóngase el objeto `ComPort1` instancia de la clase `TComPort`, establecer la configuración del puerto podría ser:

```
ComPort1->Port = "COM2";            // usar com2
ComPort1->BaudRate = br9600;        // velocidad 9600
ComPort1->DataBits = dbSeven;       // bits de datos
ComPort1->Parity->Bits = prOdd;     // paridad impar
ComPort1->StopBits = sbTwoStopBits; // dos bits de stop
```

Antes de "usar" el puerto hay que "abrirlo". Se debe "cerrar" si se deja de usar o se quiere cambiar la configuración. Este mecanismo permite al S.O. proporcionar a una tarea el uso exclusivo de un puerto.

```
ComPort1->Open();     // intentar abrir el puerto
```

Ahora ya se puede enviar algo al puerto. Por ejemplo, se puede enviar cualquier secuencia de datos,

```
unsigned char datos[]={54,97,10,234};

ComPort1->Write (datos, 4);
```

Y la secuencia 54, 97, 10, 234 se envía por el serie.

En el caso de que la configuración del puerto no admita la totalidad de bits del dato, entonces los bits más significativos son descartados. Por ejemplo, si elegimos 6 bits de datos, se descartan los 2 bits de mayor peso de cada dato.

Es importante destacar que el programa no espera a que se haya enviado la información físicamente por el cable. Ésta es colocada en un "buffer" y es el S.O. el encargado de ir enviándolo mediante el mecanismo de interrupciones. Así se consigue que nuestro programa continúe haciendo otras tareas.

Para obtener en una variable la información recibida por el canal serie se podría hacer:

```
char buffer[1000];
```

```
int num_datos;

num_datos = ComPort1->Read(buffer, 10);
```

que indica que se saquen 10 datos del tampón de entrada. Si al llamar a la funcin no estn disponibles 10 datos, entonces se esperar a que lo esten.

Es muy habitual que lo enviado por el canal serie sean datos correspondientes a cadenas de caracteres, por lo que se suele facilitar el envío y recepción de caracteres. Por ejemplo, se puede usar la siguiente sintaxis,

```
ComPort1 -> WriteStr("Hola");
```

y la secuencia ´H´, ´o´, ´l´,´a´, se envía por el serie.

Para obtener en una variable de cadena la información recibida por el canal serie se podría hacer:

```
AnsiString a;
int num_datos;
num_datos = ComPort1 -> ReadStr(a,10);
```

En realidad no existe diferencia alguna entre enviar/recibir caracteres y datos, pues se trata en los dos casos de datos binarios de n bits.

(parlar de com configurar els protocols)

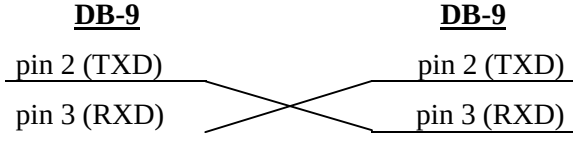
La mejor forma de "recibir" información sin utilizar técnicas de encuesta (polling) que desperdicien el tiempo de la CPU es aprovechar el evento `OnRxChar`, que se genera cuando han llegado datos por el puerto serie.

```
... OnRxChar (TObject Sender, int Count)
{
   ...
   ComPort1->ReadStr(Label1 -> Caption, Count);
}
```

Ejemplo:

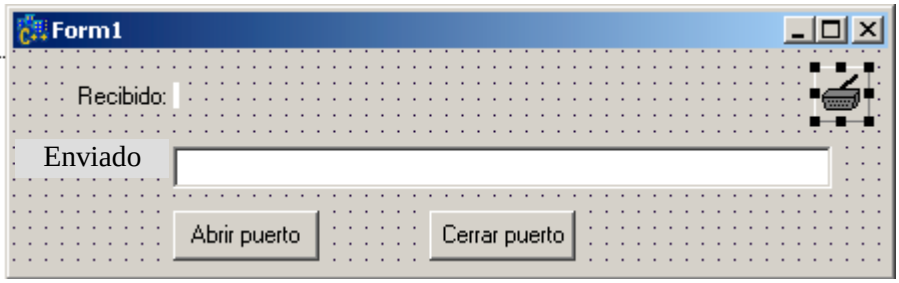
Se desea conectar dos PC por el puerto serie y desarrollar una aplicación de consola de manera que lo que se teclee en uno de los ordenadores aparezca en el otro.

- En primer lugar hay que interconectar los dos ordenadores. Si tienen conectores tipo DB-9 se puede realizar un cable NULL-MODEM básico según el siguiente esquema.



pin 5 (SG) pin 5 (SG)

- El siguiente programa sirve para enviar caracteres por el serie utilizando el componente.




```
//--------- PROGRAMA PARA ENVIAR ----------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "CPort"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
        ComPort1->Port = "COM1";     // 9600, 8N1
        ComPort1->BaudRate = br9600;
        ComPort1->Parity->Bits = prNone;
        ComPort1->DataBits = dbEight;
        ComPort1->StopBits = sbOneStopBit;
        ComPort1->Open();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
        ComPort1->Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char &Key)
{
        ComPort1->Write(&Key,1);
}
//---------------------------------------------------------------------------
```

- El siguiente listado permite recibir caracteres por el puerto serie y mostrarlos por pantalla.

```
//-------------- PROGRAMA PARA RECIBIR ----------------------------------
```

```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "CPort"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
        //ComPort1->Port = "COM1";
        ComPort1->BaudRate = br9600;
        ComPort1->Parity->Bits = prNone;
        ComPort1->DataBits = dbEight;
        ComPort1->StopBits = sbOneStopBit;
        ComPort1->Open();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
        ComPort1->Close();
}
//---------------------------------------------------------------------------
char buffer[1000]="";   //buffer vacio
void __fastcall TForm1::ComPort1RxChar(TObject *Sender, int Count)
{
        AnsiString a;
        ComPort1->ReadStr(a,Count);
        strcat(buffer,a.c_str());
        Label2->Caption = buffer;
}
//---------------------------------------------------------------------------
```

Actividad.

Modificar los programas anteriores para que en ambas partes se puedan teclear y recibir datos.

4.2.6 APLICACIÓN GENERAL A DISPOSITIVOS INDUSTRIALES

La norma RS-232 es utilizada en una gran variedad de dispositivos de dos maneras no excluyentes:

- **Configuración del dispositivo**. Para establecer el comportamiento del dispositivo.
- **Trabajo normal**. El dispositivo trabajará en colaboración con otro dispositivo, recibiendo y/o enviando información durante el trabajo normal.

Un ejemplo típico del primer caso sería la programación de un autómata programable, donde, en una primera fase, se transfiere a través de una conexión serie el programa de control que ejecutará el autómata.

Un ejemplo del segundo caso sería un módulo GPS conectado a través de su salida serie a un sistema de seguimiento de vehículos.

Para la configuración de dispositivos suele ser habitual utilizar un ***terminal serie***. Un terminal serie es un equipo constituido por una pantalla, un teclado y una conexión serie que permite representar en pantalla los datos que se reciben por el serie como códigos ASCII, mientras que los grafos que se pulsan en el teclado son enviados codificados en el correspondiente código ASCII. Los terminales disponen de microinterruptores que permiten configurar los parámetros de la conexión serie (bits de datos, paridad, bit de stop, velocidad, etc.).

Actualmente suele ser muy habitual sustituir el terminal por un programa ejecutándose sobre un computador que lo emula, y aportando ventajas añadidas como mayor flexibilidad en la configuración y posibilidad de transferencia de ficheros.

(Ficar fotos de terminal y ordinador amb hyperterminal).

El uso habitual de los terminales hace que muchas veces la información transferida entre los dos equipos sea textual y con significado fácilmente comprensible por el usuario. De lo dicho se deduce que hay dos modalidades de transferir datos en RS-232: ***información binaria*** o ***información en forma de cadenas de texto*** (en realidad la segunda es un subconjunto de la primera).

Desde el punto de vista del programador, ambas aproximaciones son equivalentes, la diferencia es que, en el segundo caso, la interpretación humana del significado es más sencilla.

El aprovechamiento de la conexión serie consistirá entonces en saber generar las secuencias a enviar por el canal serie y en interpretar dichas secuencias, aspecto que se trata en este apartado.

Partiendo de que el puerto serie del computador está abierto y configurado adecuadamente, supóngase que se tienen dos funciones generales que permiten usar la conexión serie y cuyos prototipos son

```
void envia_rs232(unsigned char *datos, int cant_datos);

void recibe_rs232(unsigned char *datos, int *cant_datos);
```

La función `envia_rs232()`, deposita en el buffer de transmisión serie del sistema operativo (es decir, no espera a que termine la transmisión de datos) una secuencia de `cant_datos` datos que serán transmitidos uno a uno a través del serie. Como resultado devuelve si los datos han sido correctamente volcados en el buffer. En función de la configuración del número de bits, los bits de mayor peso de los bytes son descartados.

Por ejemplo, para transmitir los datos 125, 33, 40, 100 en ese orden se podría hacer:

```
unsigned char info[] = {125, 33, 40, 100};

envia_rs232( info, 4);
```

Para transmitir la cadena “Hola” se podría hacer:

```
char *cad = ”Hola”;
envia_rs232( cad, strlen(cad));
```

Obsérvese que realmente no hay diferencia entre las dos posibilidades.

La función `recibe_rs232()` copia en la dirección indicada por el puntero los datos recibidos a través del serie. En `cant_datos` se deposita la cantidad de datos copiada, pudiendo valer 0 si no se ha recibido nada (ello permite comprobar si se ha recibido algo sin bloquear el programa que utiliza dicha función aplicando una técnica de encuesta o polling).

Para recoger la información recibida del serie, independientemente de si es texto o datos, se podría hacer,

```
unsigned char buffer[1000];  //deposito datos
int n_datos;

recibe_rs232( buffer, &n_datos);
```

Si los datos son textuales, podríamos mostrarlos por pantalla de la siguiente forma,

```
buffer[n_datos] = '\0'; //para que sea cadena C
printf("Se ha recibido %s\n", buffer);
```

En el caso de información textual, se deberá tener especial precaución con los 'retornos de carro' para que sean compatibles con el sistema utilizado. Como recomendación, nunca se deberá utilizar '\n' para evitar la traslación CR+LF de MS-DOS/Windows.

Estas funciones son lo suficientemente genéricas como para que puedan aplicarse a cualquier software que utilice el serie, incluido el componente elegido.

Como estrategia de aprendizaje y afianzamiento del uso de programas que aprovechen el puerto serie se propone la resolución de las siguientes actividades mediante el método del descubrimiento guiado.

Actividad.

Se desea controlar una red industrial basada en RS-485 (suponemos ahora que es lo mismo que RS-232) usando un PC. Cada módulo conectado a la red tiene un identificador diferente y se desea acceder a un módulo de salidas digitales que acepta el formato de mensaje abajo mostrado.

Diseñar una función para C++Builder que haga uso de la función `envia_rs232()` y construya un mensaje y lo envíe por el serie a partir de los parámetros que se le pasan como argumento.

Formato mensaje:

#mmssvccc
mm → identificador de módulo, número decimal de 2 cifras
ss → salida a modificar, número decimal de 2 cifras
v → valor de la salida ("0" o "1")
ccc → suma de comprobación, la suma de mm, ss y v con 3 cifras decimales

Ejemplo: #07011009 → módulo 7, salida 1, valor 1, suma de comprobación 9

La función a rellenar es

```
void maneja_modulo(int num_modulo, int num_salida, int va-
lor) {

}
```

Y un ejemplo de uso:

```
maneja_modulo(7,1,1);
```

Siguiendo con la metodología del descubrimiento se propone ahora una actividad que pretende enseñar las bases del tratamiento de la información que se recibe por una conexión serie.

En este caso se supone la llegada de un flujo continuo de información. Dado que el programa "abre" el puerto en un momento indefinido, el flujo de información puede encontrarse en cualquier punto de la secuencia.

Para interpretar la información se sigue el siguiente planteamiento:

1. Acumular fragmento recibido en un buffer.
2. Comprobar la completitud del mensaje.
   - SI: Interpretar y eliminar mensaje.
   - NO: Continuar con 1.

Actividad.

Un módulo industrial de entradas digitales emite continuamente la secuencia de caracteres mostrada abajo, donde el carácter numérico de más a la derecha corresponde a la entrada 1 y el de más a la izquierda a la entrada 8.

Los caracteres "H" y "L" representan un nivel alto y bajo respectivamente en la entrada correspondiente.

Suponiendo que se conecta el módulo a una entrada serie de un PC y se usa la función `recibe_rs232()`, desarrollar el programa que analiza los caracteres recibidos y, en función del valor de la entrada 7, poner la propiedad Visible a true (para un 1) o a false (para un 0) de un objeto llamado "Puerta".

...*HLLHHLHL...

Actividad.

Adapta las actividades anteriores al componente TComPort.

En el segundo caso se deberá emplear el evento de recepción de caracteres.

4.2.7 EJEMPLO: VISOR DE PESO G6

La empresa Graviton (www.graviton.es) produce distintos equipos para pesaje industrial. Una familia de equipos producidos son los visores de peso, que son dispositivos que se conectan a las células de carga (sensores de peso configurados normalmente en puente) y permiten mostrar el peso sobre la plataforma de pesaje.

Un de los visores de peso producidos es el modelo G6 (ver Figura 4-1). La Tabla 4-1 resume sus características.

Figura 4-1. Visor de peso G605SP

Tensión de célula	+5V
Sensibilidad	0,5 mV/V a 3,5 mV/V
Resolución convertidor	24 bits (16.777.215 puntos)
Resolución final	167.772 puntos
Frecuencia de conversión	10, 20, 30, 40, 50 Hz
Precisiones	3000, 6000, multirango, libre
Fondo de escala (F.E.)	1 a 150000 Kg
Máxima no linealidad	15 ppm de F.E.
Máxima no repetibilidad	10 ppm de F.E.
Deriva de ganancia	<5 ppm/ºC
Deriva de cero	<15 ppm/ºC
Peso muerto	75 % F.E.
Tara sustractiva	100 % F.E.
Tiempo de recuperación del cero	5 seg
Células conectables	8 de 360 Ohms
Alimentación	18-24 V DC; 13-22 V AC
Dimensiones (mm)	96x48x125
Relés	3 Relés reed (Vmax. 48 V , Imax 200 mA)
Consumo	5 W

Tabla 4-1. Características del visor G605SP

El visor posee una salida estándar RS-232 que permite su conexión al computador. En el manual del visor se incluye la siguiente información para el aprovechamiento del serie:

Com1	Bauds	Selecciona la velocidad del RS232.Permite seleccionar desde 300 bauds a 9600 baudios.
	Palabra	Selecciona el formato de la palabra del RS232.Se selecciona una palabra de 3 caracteres en la que el primer carácter de la izquierda indica el bit de paridad, n= no paridad, o= paridad impar, e= paridad par. El carácter central es una cifra que indica el número de bit de la palabra y puede ser 7 u 8.El carácter de la derecha indica el número de stops bits y puede ser 1 ó 2.
	Nombre	Programa el nombre del terminal.Para comunicación vía RS 232 si no es indispensable se aconseja no dar nombre al terminal para agilizar la comunicación.
	Test	Manda un carácter [Z] por el RS232. Si ha realizado un puente entre los terminales 2 y 3 del conector RS232, y se recibe bien el carácter, el visor indicará "bien" en el display en caso contrario indicará "mal".
	Edita	Muestra en pantalla todos los caracteres recibidos por el RS232.

Protocolo de comunicaciones (I)

Se puede comunicar el visor de peso desde un programa standard de emulación de terminal como el HyperTerminal (Hilgraeve). Este programa viene incluido con la instalación de Windows 95 (Microsoft). Una vez realizadas las conexiones el terminal contestará a los siguientes comandos:

Conectar [Nombre]	Conecta el terminal
Bruto	Bruto
Cero	Adquisición de cero
Leds	X, Signo, Cero, Tara, Bruto, Estable, X, X
Reles	X, Grueso, Fino, Vaciado, X, X, X, X
Help	Listado de comandos

Se puede configurar el visor de manera que envíe una secuencia fija de caracteres que contienen información sobre el peso medido e información adicional. El formato de la cadena es:

```
.nkk000004BRUTO000000
```

donde los 6 últimos dígitos corresponden al peso en kilos medido.

4.3 EJEMPLO: VISOR DE PESO IE-21

El visor de peso industrial modelo IE-21 de la empresa Microgram Instruments Española S.A. posee una salida de comunicación serie RS-232 configurable en velocidad, bits de datos, bits de stop, y paridad. El formato de la transmisión de datos, puede variar entre 7 modalidades distintas, configurables externamente.

Gracias a la gran flexibilidad de la configuración del visor, se puede conectar a cualquier plataforma o sistema de pesaje (tolvas, silos, basculas aéreas, etc.).

El visor posee un conversor analógico/digital de 16 bits, lo que supone mas de 65000 subdivisiones del peso. El peso se muestra a través de 6 displays de 7 segmentos con punto decimal.

Internamente, se puede configurar el modo de funcionamiento (valor de la división, valor de fondo de escala, transmisión bruto / neto, nº de decimales, brillo de los displays, etc.…) y ajustar la ganancia del peso, y los parámetros para la conversión.

Tiene un pulsador para tarar el peso, situado en ese momento sobre la plataforma, que desaparece automáticamente cuando se detecta una pesada estable superior al peso tarado seguida de una bajada del peso hasta el nivel cercano al cero. Otro pulsador sirve para bloquear la tara, y que esta se mantenga hasta que esta se desbloquee mediante una nueva pulsación del mismo.

También muestra información adicional mediante 5 leds, con los que informa cuando el peso es estable, cero, o esta dentro de la zona de pesada mínima. Dos de ellos sirven para señalar cuando actúa la tara y el bloqueo de tara.

Figura 4-1. Visor de peso modelo IE-21.

Para interconectar el visor con un PC a través de la conexión serie se debe tener en cuenta que los dos lados actúan como DTE, por lo que el cable de conexión es una variante null-modem con un conector DB-9 hembra para el PC y un conector DB-9 macho para el lado del visor. Las señales RXD y TXD van cruzadas al igual que las señales RTS y CTS. En el conector del lado del ordenador hay un puente entre las señales DTR y DSR

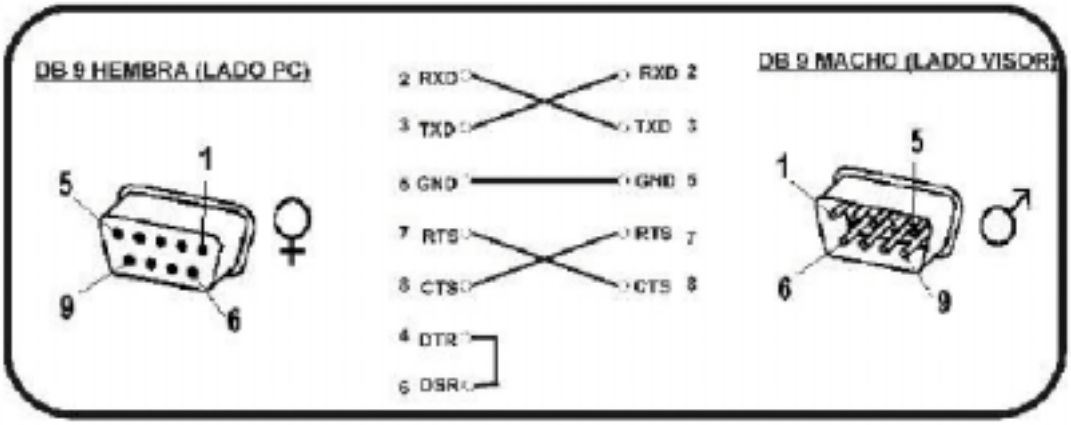


Figura 4-2. Esquema del cable de conexión RS-232 de PC / IE-21.

La configuración por defecto de la salida serie del visor, es la siguiente:

- Velocidad: 1200 Baudios (9600).
- Paridad: Impar.
- Bits de datos: 7.
- Bits de stop: 2.
- Transmisión por petición.

El formato de la transmisión de datos, es en modo ASCII, y la siguiente forma:

"SD - 335.0 kg"

El carácter de inicio de trama es una **"**S". El siguiente carácter, nos da información sobre el estado del peso; si este es una **"**D", quiere decir que el *peso no es estable*, si es un **"espacio**", quiere decir que el *peso es estable*. En el caso de que este carácter sea una **"**I", nos esta indicando que el peso esta *fuera de rango*, y en el siguiente carácter nos indicara por donde se sale de la escala:

- **"+"** El peso se pasa de la escala.
- **"-"** El peso no llega al mínimo de la escala. (por debajo de cero)

Con lo qué la trama queda de la siguiente forma: **SI+**

Si el peso esta dentro del rango, el tercer carácter siempre es un **"espacio**". El cuarto carácter será un espacio si el peso es positivo, y un *"-" si es negativo*.

Los 9 caracteres siguientes, representan el valor del peso en formato ASCII, incluido el punto decimal. Seguidamente a estos, viene la trama **"espacio**" seguida de **"kg**". Por último el carácter final de trama es un retorno de carro.

4.4 CONEXIÓN SERIE RS-485

Como actualizaciones de la norma RS-232 se propusieron nuevas normas, por ejemplo:

- RS-422. Especificación eléctrica diferencial.
- RS-423. Conexión sistemas RS-422 y RS-232.
- RS-485. Especificación eléctrica multimaestro.

Las limitaciones de la norma RS-232 se pueden superar con las mejoras que aportan estas normas posteriores.

Gracias al uso de una transmisión balanceada, RS-485 aporta como beneficio inmediato una mayor velocidad de transmisión, una longitud de línea del orden de kilómetros y una buena inmunidad al ruido.

FICAR FOTO, UN OMROM AMB RS_485 i PROFIBUS ESTARÍA BÉ.

4.4.1 ESPECIFICACIONES Y CARACTERÍSTICAS

La norma RS-485 engloba características de propuestas posteriores a la RS-232. Las siguientes características y especificaciones son las más destacables:

- Red multipunto, con hasta 32 emisores/receptores
- Medio físico
  - Par trenzado (mejor si se apantalla).
  - Terminadores de 120 Ω.
  - Opcionalmente, resistencias pull-up y pull-down.
- Tensiones
  - Voltaje en modo común de –7 a +12 V.
  - Histéresis de 200 mV.
  - "1">= +0.2 V, "0" <= -0.2 V.
- Formato de los datos idéntico a RS-232
- Control de flujo half-duplex.

La distancia máxima que se puede alcanzar depende de la velocidad de transmisión. Por ejemplo:

- a 10 Mbps → 12 metros.
- a 100 Kbps → 1200 metros

La Figura 4-1 muestra un diagrama de una conexión tipo RS-485. Las estaciones transmisoras se conectan, en cualquier punto, a los conductores de un único par trenzado. A los extremos

del par se conectan resistencias terminadoras de 120 Ohms. En la Figura 4-2 se muestra en más detalle la circuitería implicada en la transmisión y recepción de datos. Como el medio se comparte entre todas las estaciones, la transmisión solo podrá ser half-duplex.

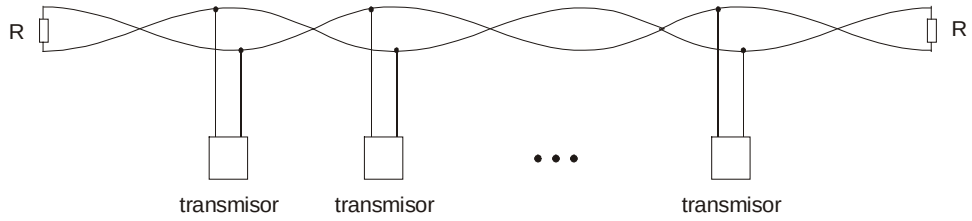


Figura 4-1. Esquema de una conexión RS-485

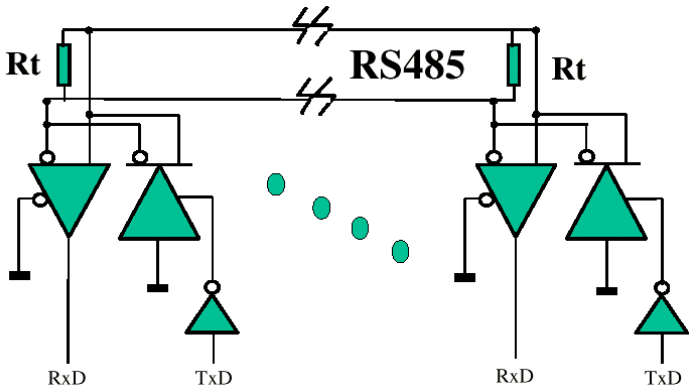


Figura 4-2. Circuitería del emisor/receptor

El estándar RS-485 está ampliamente extendido en la industria, utilizándose para los siguientes tipos de comunicación:

- Punto a punto.
- Maestro-esclavos.
- Multimaestro.

En función del tipo de comunicación seleccionado aparecerán diferentes problemáticas a la hora de compartir el medio físico. La problemática a este nivel la debe resolver el software que se desarrolle en el nodo de transmisión.

La Figura 4-3 muestran una configuración maestro-esclavo. En esta configuración, el maestro es el único autorizado a iniciar una transmisión, que serán básicamente órdenes a los esclavos para que hagan determinadas cosas o respondan con su estado (por ejemplo, la lectura de un sensor).

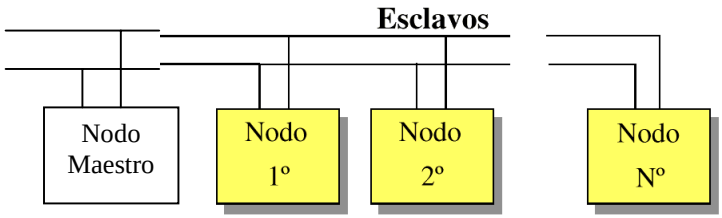


Figura 4-3. Configuración maestro-esclavos (DRETS)

La Figura 4-4 muestra una configuración multimaestro. Al compartirse un medio físico y ser la transmisión half-duplex, es necesario establecer un reparto del uso del medio por parte de los maestros. Una opción empleada suele ser crear un anillo lógico, que consiste en un mecanismo de paso de turnos entre los maestros mediante el intercambio de un testigo.

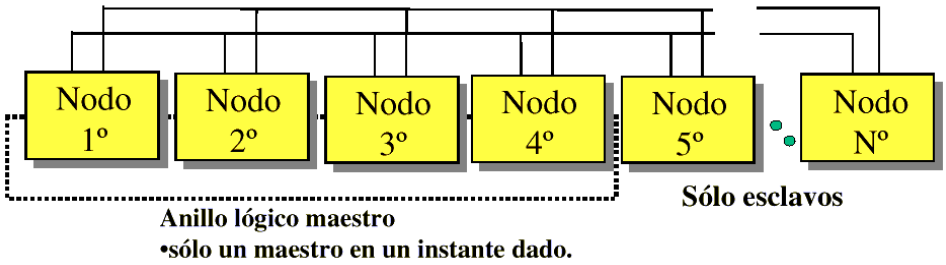


Figura 4-4. Configuración multimaestro (DRETS)

La Tabla 4-1 resume las características de algunas normas RS.

NORMAS	RS-232	RS-422	RS-485
Modo	Simple	Diferencial	Diferencial
Número transmisores	1	1	32
Número receptores	1	10	32
Longitud máxima	15 m	1200 m	1200 m
Velocidad máxima	20 Kbp	10 Mbps	10 Mbps
Salida transmisor	± 5 V min ± 15 V max	± 2 V min	± 1.5 V min
Carga al transmisor	3K a 7K	100 min	60 min Ω
R de entrada al receptor	3K a 7K	4K min	12 K min
Sensibilidad del receptor	± 3 V	± 200 mV	± 200 mV

Tabla 4-1. Comparación de distintas normas RS (DRETS)

4.4.2 USO DEL RS-485 CON EL PC

El hecho de que el formato de datos utilizados por RS-232 y RS-485 capacita al PC a usar fácilmente una conexión RS-485. Para ello se utilizan los adaptadores de señal disponibles en el

mercado que, básicamente, adaptan los niveles de tensión y el modo de transmisión de la conexión RS-232 del PC a la norma RS-485.

Con uno de estos adaptadores capacitaremos al PC para comunicarse, por ejemplo, con sensores y actuadores industriales, PLCs, reguladores, etc. En la Figura 4-1 puede verse una configuración típica.

Para alcanzar mayores velocidades que el RS-232 específico del PC será necesario adquirir una tarjeta RS-485 para instalarla en una ranura de expansión del PC. Desde el punto de vista del programador, las conexiones serie se verán en Windows como COMx adicionales.

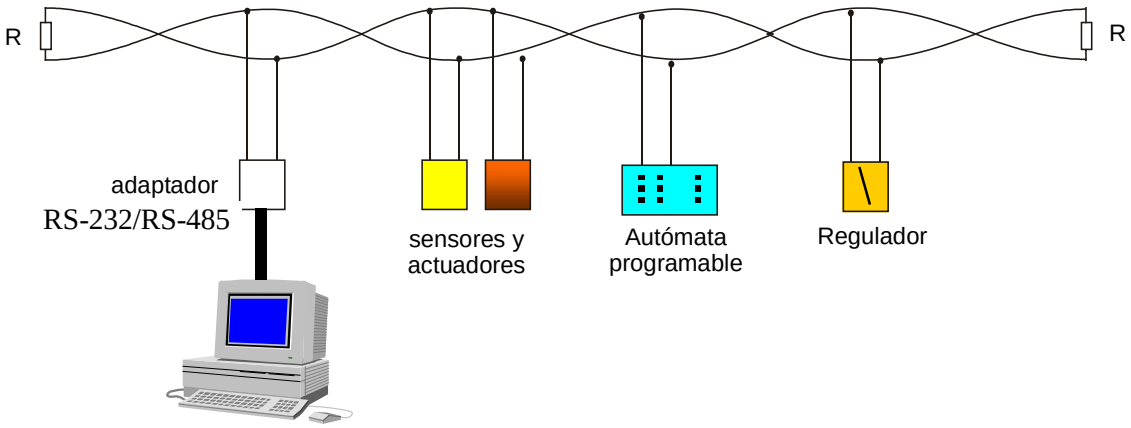


Figura 4-1. Configuración típica con un PC conectado a un RS-485

Actividad

Localizar información en Internet sobre aplicaciones del RS-485 y sus precios. Antes de empezar el trabajo se deberá decidir sobre que se quiere localizar información y comentarlo con el profesor. Al final de la actividad, cada grupo expondrá lo más interesante que ha encontrado.

Si tenéis interés en algún tipo de producto específico, por ejemplo, autómatas y reguladores:

- www.omrom.com, autómatas y reguladores
- www.embedded.com, de todo sobre informática industrial

Se puede recurrir también a las siguientes direcciones para localizar información:

- www.adlink.co.tw, módulos NuDam que se estudiarán y usarán más adelante
- www.advantech.com, módulos ADAM similares a los anteriores
- www.qnv.com, suministrador de equipos informáticos industriales en España, dispone de lista de precios.

Un buscador también es una buena opción

4.4.3 APLICACIÓN A UN PRODUCTO INDUSTRIAL:LOS MÓDULOS NUDAM DE ADLINK

4.4.3.1 Introducción

NuDAM son una colección de módulos de adquisición de datos de la empresa ADlink Technologies (www.adlink.com.tw) que proporcionan una solución muy completa para sistemas distribuidos de control y adquisición datos en configuración maestro-esclavos.

El sistema de interconexión de los módulos se basa en el estándar RS-485. Se pueden interconectar hasta 256 módulos remotos utilizando un par trenzado y cubrir una distancia de hasta 1200 metros (4000 pies). Se puede ampliar la red utilizando repetidores que unan segmentos.

Los módulos se pueden controlar utilizando un sencillo protocolo de órdenes y respuestas ASCII a través del RS-485. Cada módulo tiene una dirección identificativa única en la red que permite distinguir a unos de otros.

Algunos módulos disponibles son:

- De comunicación:
  - NuDAM-6520, conversor RS-232 a RS-485.
  - NuDAM-6510, repetidor RS-485.
- Entrada analógica:
  - NuDAM-6011, entradas analógicas de alta ganancia con E/S digital.
  - NuDAM-6012, entradas analógicas.
  - NuDAM-6013, 3 canales RTD.
  - NuDAM-6017, 8 canales de entrada analógica.
  - NuDAM-6018, 8 canales para termopar.
- Salidas analógicas:
  - NuDAM-6021, salida analógica.
- E/S digital:
  - NuDAM-6050, 7 entradas digitales y 8 salidas digitales.
  - NuDAM-6052, entradas digitales lejanas.
  - NuDAM-6060, salidas de relés y entrada digital.
- Contadores:
  - NuDAM-6080, entrada de contador/frecuencia.

Para conectar el sistema a un computador con conexión RS-232 se podrá emplear el conversor NuDAM 6520.

4.4.3.2 Configuración de la red.

Para montar una red se pueden usar hasta 256 módulos y un máximo de 128 módulos por segmento. En la Figura 4-1 se muestra un esquema de la conexión de un PC a un segmento. Obsérvese que cada segmento del bus se debe acabar con una resistencia terminadora de 120 Ω.

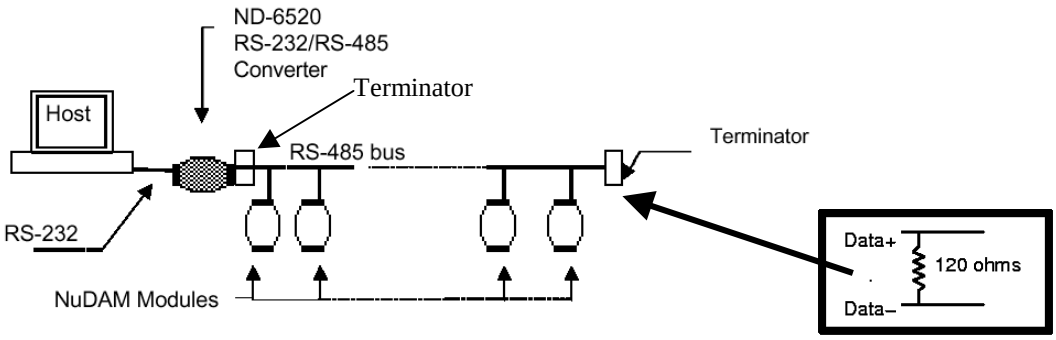


Figura 4-1. Configuración sencilla con módulos Nu-DAM

En el caso de necesitar extender más la red o crear bifurcaciones se pueden emplear repetidores, tal y como se muestra en la Figura 4-2 y en la Figura 4-3.

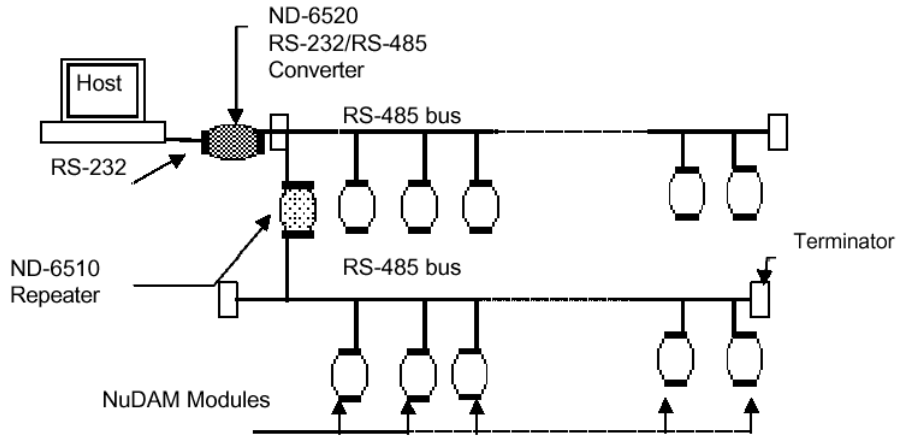


Figura 4-2. Configuración ramificada

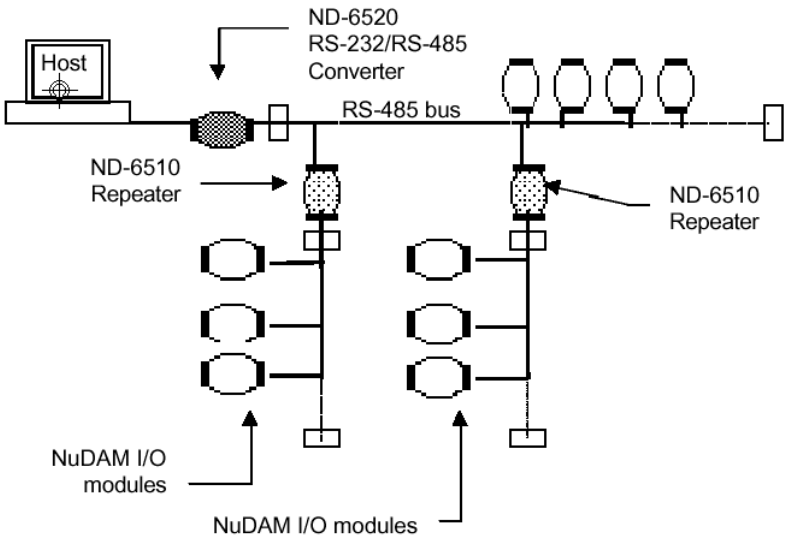


Figura 4-3. Configuración libre

Para el cableado se usará par trenzado, y se recurrirá a par trenzado apantallado en caso de interferencias, teniendo especial precaución en cuidar la conexión a tierra. La Figura 4-4 muestra una recomendación para la derivación a tierra.

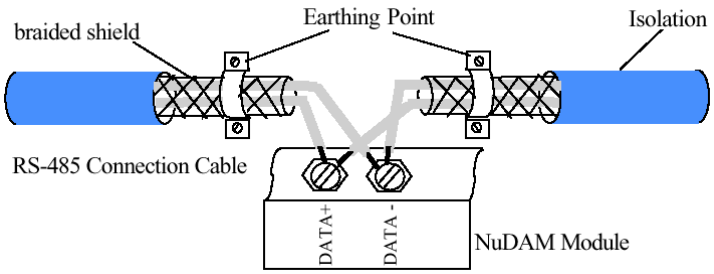


Figura 4-4. Conexión a tierra

4.4.3.3 Módulo conversor RS-232 a RS-485 NuDAM-6520.

Para poder aprovechar una red con estos módulos desde un PC es necesario disponer de una conexión RS-485, para ello se puede recurrir a tarjetas de expansión disponibles en el mercado, que, una vez instalada en el computador, se comportarán como una conexión serie cualquiera, identificándose como un COMx más y configurándose de igual forma.

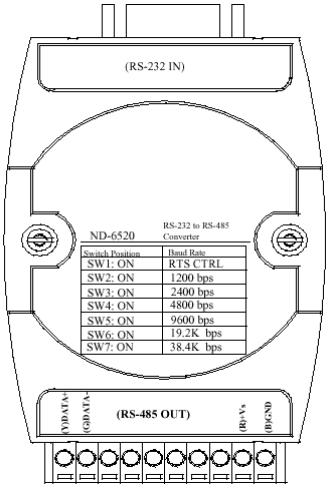
En el caso de que no se desee instalar una nueva tarjeta de expansión es posible recurrir a un conversor RS-232 a RS-485. La empresa ADLink proporciona el módulo NuDAM 6520 para este menester.

Para utilizar el módulo basta con conectarlo con un cable serie DTE-DCE (sin ningún cruce de cables) al PC, alimentarlo y conectarlo al par trenzado. En la Tabla 4-1 se pueden comprobar las especificaciones.

- **Input**
  - Interface : standard RS-232 9 pin D-type connector
  - Speed (bps) : 1200, 2400, 4800, 9600, 19.2K, 38.4K, RTS
  - Data Format : 8 bits, 9 bits, 10 bits, or 11 bits
- **Output**
  - Interface : RS-485, differential, 2 half-duplex wires
RS-422, differential, 4 duplex wires
  - Speed (bps) : 1200, 2400, 4800, 9600, 19.2K, 38.4K, RTS
  - Max distance : 4000 ft. (1200m)
- **Isolation**
  - Isolation voltage : 3000 V DC
- **Bus**
  - Max Loading : 128 NuDAMs on a bus
  - Max modules : 256 NuDAMs with one ND-6510 repeater
- **Power**
  - Power Supply : +10V to +30V
  - Power Consumption : 1.2 W

Tabla 4-1. Especificaciones del módulo NuDAM 6520

La Figura 4-1 muestra el aspecto del módulo y el significado de cada una de las conexiones disponibles.



Pin #	Signal Name	Description
1	(Y)DATA+	RS-485 transmission line, positive
2	(G)DATA-	RS-485 transmission line, negative
9	(R)+VS	NuDAM power supply, +10V~+30V
10	(B)GND	NuDAM ground
--	RS-232 IN	9-pin RS-232 connector

Figura 4-1. Aspecto del módulo NuDAM 6520 y significado de las conexiones

En el módulo se deberá configurar previamente la velocidad en bps a la que se desea transmitir.

Para su conexión al PC se deberá respetar el interconexionado mostrado en la Figura 4-2 Como ya se ha dicho, se deduce de él que se trata de una conexión DTE-DCE.

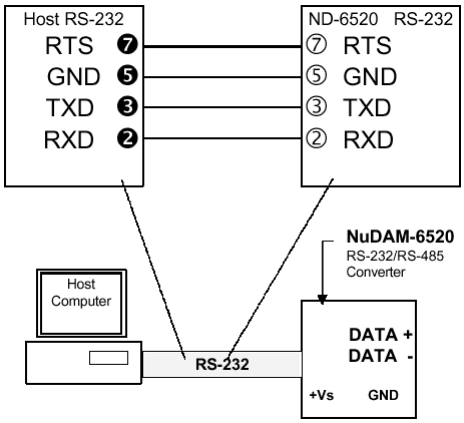


Figura 4-2. Conexión física entre un DB-9 en el PC y el DB-9 del módulo

4.4.3.4 Módulo de E/S digital. NuDAM-6050

Todos los módulos NuDAM siguen un esquema de uso muy similar. Para describir la forma de aprovecharlos a través de la conexión RS-485 se tomará como referencia el módulo de entradas y salidas digitales NuDAM-6050.

Este módulo tiene una serie de entradas digitales que permiten monitorizar señales activas TTL o interruptores ON/OFF pasivos gracias a una resistencia PULL-UP interna. Las salidas digitales consisten en transistores en colector abierto que pueden drenar una corriente máxima de 30 mA. La Tabla 4-1 resume sus especificaciones.

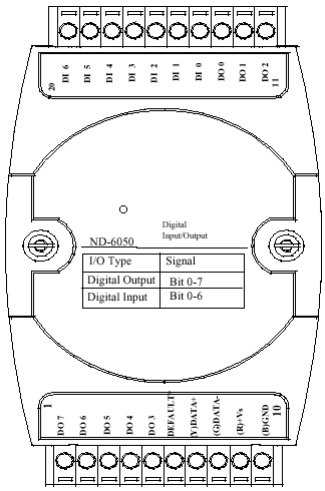
- **Interface**
  - Interface : RS-485, 2 wires
  - Speed (bps) : 1200, 2400, 4800, 9600, 19.2K, 38.4K
- **Digital Input**
  - Channel numbers : 7
  - Logical level 0 : +1V maximum
  - Logical level 1: +3.5V~30V
  - Pull up resister : 10KΩ
  - Maximum current : 0.5mA
- **Digital Output**
  - Channel numbers : 8
  - Output characteristic : open collector transistor
  - Maximum current sink : 30mA
  - Max. power dissipation : 300mW

✧ **Watchdog Function**
- Module internal watchdog timer: 1.6 sec (Firmware Rev. 1.x)
  150 ms (Firmware Rev. 2.x)
- Power failure threshold : 4.65 V
- Safety value : 8 output channels
- Host programmable watchdog :
  53.3 ms ~ 13.645 sec (Firmware Rev. 1.x)
  100 ms ~ 25.500 sec (Firmware Rev. 2.x)

✧ **Power**
- Power supply : +10V to +30V
- Power consumption : 0.4 W

Tabla 4-1. Especificaciones del módulo NuDAM 6050

La Figura 4-1 muestra el aspecto del módulo y la función asignada a cada conexión. La referencia de las entradas y salidas digitales corresponde al pin GND.



Pin #	Signal Name	Description
1	DO 7	Digital output channel 7
2	DO 6	Digital output channel 6
3	DO 5	Digital output channel 5
4	DO 4	Digital output channel 4
5	DO 3	Digital output channel 3
6	Default*	Initial state setting
7	(Y) DATA+	RS-485 series signal, positive
8	(G) DATA-	RS-485 series signal, negative
9	(R) +Vs	Power supply, +10V~+30V
10	(B) GND	Ground
11	DO 2	Digital output channel 2
12	DO 1	Digital output channel 1
13	DO 0	Digital output channel 0
14	DI 0	Digital input channel 0
15	DI 1	Digital input channel 1
16	DI 2	Digital input channel 2
17	DI 3	Digital input channel 3
18	DI 4	Digital input channel 4
19	DI 5	Digital input channel 5
20	DI 6	Digital input channel 6

Figura 4-1. Aspecto del módulo NuDAM 6050 y significado de las conexiones

Para manejar todos los módulos de emplean órdenes con el formato:

(Leading Code)(Addr)(Command)[Data]**[Checksum]**<CR>

When checksum is enable then **[Checksum]** is needed, it is 2-character.

(Leading Code)	Leading Code is the first characteristic of the NuDAM command. All NuDAM commands need a command leading code, such as %,$,#,@,...etc. **1- character**
(Addr)	Module' s address ID, the value is in the range of 00 - FF (Hexadecimal) if no specified in the following. **2- character**
(Command Variable)	Items indicate command codes or value of variables. **Variable length**
[Data]	Some output command need data. **Variable length**
[Checksum]	Checksum in brackets indicate optional parameter, only checksum is enable then this field is required. **2- character**
< >	Identifies a control code character, such as <CR> for carriage return, its value is 0x0D. **1- character**

Por ejemplo,

```
User Command    $012<CR>
:
Response        !01400600<CR>
:
```

$: LeadingCode
01 : Address
2 : Command (Read Configuration)
<CR> : Carriage return 0x0D

Note : 1. There is no spacing between characters.
2. At end of command need a <CR> carriage return 0x0D.
3. Checksum is optional parameter.

Note : Under the following conditions, there will have **no response** message.
1. The specified address ID is not exist.
2. Syntax error.
3. Communication error.
4. Some special commands does not have response .

Para realizar entrada digital se usará:

3. 14. Digital Input

@Description

Read the digital input channel value and readback the digital output channel value.

@Syntax

$(Addr)6<CR>

$	Command leading code.
(Addr)	Address ID
6	Digital data input command.

@Response

ND-6050 module response :

!(DataOut)(DataIn)00<CR>

or

?(Addr)<CR>

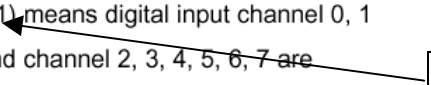
!	Command is valid.
?	Command is invalid.
(DataOut)	Value of digital output channel. **(2-character)**
(DataIn)	Value of digital input. **(2-character)**

@Example

Example for NuDAM-6050 :

User command: $306<CR>
Response: !321100<CR>

!	Command is valid.
32	32 (00110010) means digital output channel 1, 4, 5 are ON, channel 0, 2, 3, 6, 7 are OFF.
11	11 (00000011) means digital input channel 0, 1 are HIGH and channel 2, 3, 4, 5, 6, 7 are LOW.
00	No used



Cuidado, el manual está mal, 11h es 00010001b

Y para la salida digital

3. 8. Digital Output

@Description

Set digital output channel value at specified address. This command is only available to modules involving the digital output function, such as NuDAM-6050, NuDAM-6060 and NuDAM-6063.

@Syntax

#(Addr)(ChannelNo)(OutData)<CR> (6050,6060,6063 Only)

#	Command leading code. **(1-character)**
(Addr)	Address ID **(2-character)**
(ChannelNo)	00 : Set value to all channels 1X : Set value to single channel First character is 1, Second character is channel number. **(2-character)**
(OutData)	Set value to all channels : Each bit is mapping to each channel number Set value to single channel : First character is 0, second character is set to value 0 or 1. **(2-character)**

@Response

><CR>

or

?(Addr)<CR>

>	Command is valid
?	Command is invalid.
(Addr)	Address ID.

@Example

User command:	#300003<CR>
Response:	><CR>

30	Address ID
00	Set output to all channels
03	03 (00000011), Channel 0 and 1 are set ON other channels are set to OFF

User command:	#2F1201<CR>
Response:	><CR>

2F	Address ID
12	**1** : Set output to single channel **2** : Output single channel is channel 2
01	Set single channel to ON

Actividad.

Escribe la cadena que pone a “1” la salida digital 5 del módulo A7h.

Escribe la cadena que pone a “1” las 4 salidas digitales de más peso y a “0” el resto

Actividad.

Realizar una función para el módulo NuDAM 6050 cuyo prototipo sea `int salida_digital(int modulo, int salida, int estado);` a la que, pasándole como parámetros el identificador del módulo, la salida a manipular y el valor que se quiere dar a la salida, cree el mensaje necesario y lo envíe a través del RS-232 utilizando la función genérica `envia_rs232()`.

Actividad.

Amplía la función de manera que compruebe que la operación de salida se realice correctamente. Utiliza la función genérica `recibe_rs232()`. La función devolverá 0 si la operación ha ido bien y un número negativo en caso contrario.

¿Qué problemas presenta la solución propuesta? ¿Cómo se pueden solucionar?

Actividad.

Analizar un módulo NuDAM, indicando:

- Referencia numérica y propósito.
- Precio aproximado.
- Características y especificaciones más destacables.
- Conexionado al proceso.
- Formato de las órdenes y respuestas.

E/S digital: 6052, 6056, 6060 (6053, 6054, 6058, 6063)

E analógica: 6011, 6013, 6017, 6018, (6012, 6014)

S analógica: 6021, 6024

Contador: 6080

4.5 BIBLIOGRAFÍA

manual visor

manual Nudam

...