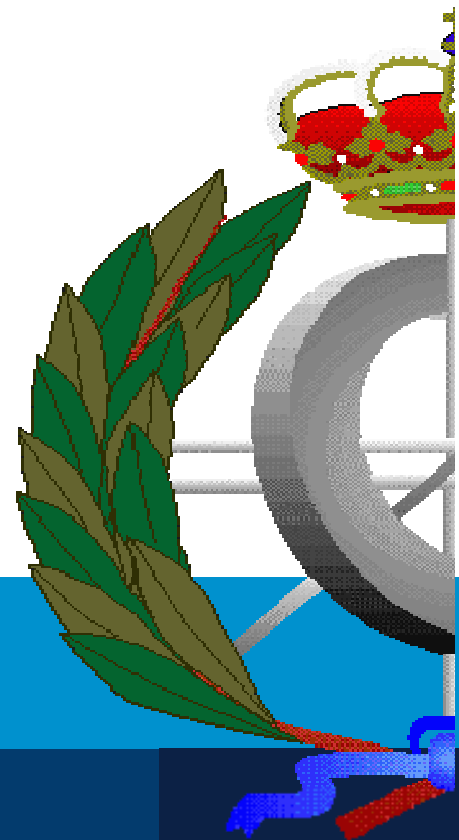
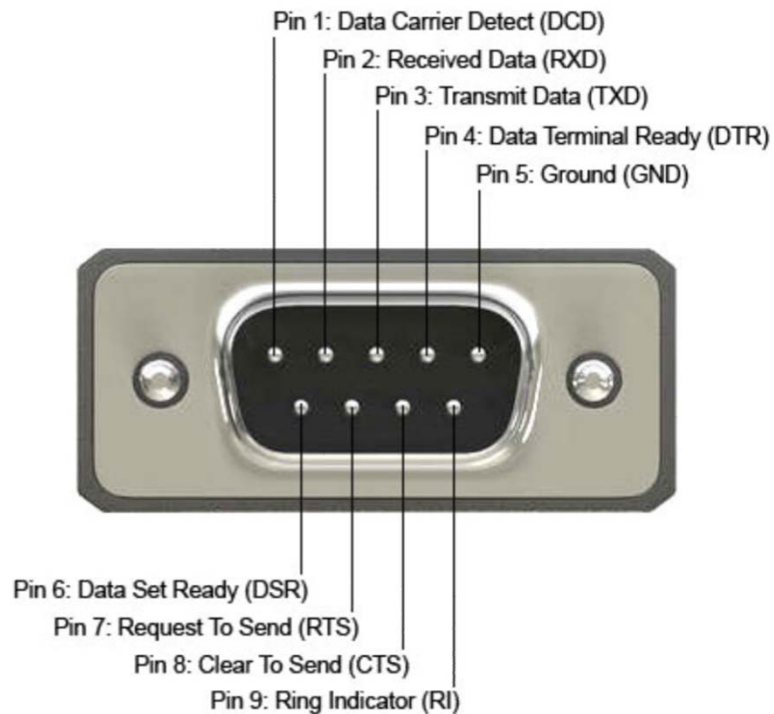




<http://informatica.usal.es/gii>
<http://informatica.usal.es/gii>

Prácticas: “El puerto Serie”

RS232 Pinout





Contenido

- Objetivos
- Introducción
- Normalización de la interfaz
- Norma EIA RS-232-C
 - Características mecánicas, eléctricas, funcionales y de procedimiento
 - Adaptación para conexión ETD-ETD
- Hardware de comunicaciones asíncronas
 - Funciones
 - Diagramas de bloques del transmisor y receptor
- Comunicaciones por puerto serie
 - Funciones de Windows
 - Funciones de UNIX
- Otras aplicaciones





Objetivos

- Hacer un cable de módem nulo para probar la comunicación ETD-ETD
- Aprender a gestionar la comunicación por el puerto serie
- Verificar cómo influyen los parámetros que definen una comunicación por el puerto serie
- Familiarizarse con la interfaz de programación del puerto serie en distintas plataformas





Introducción (I)

- Las comunicaciones serie se utilizan para enviar datos a través de largas distancias, ya que las comunicaciones en paralelo exigen demasiado cableado para ser operativas
- Los datos serie recibidos desde un modem u otros dispositivos son convertidos a paralelo gracias a lo cual pueden ser manejados por el bus del PC
- El puerto serie del PC es un dispositivo asíncrono
 - Un bit de comienzo identifica el **inicio**
 - Los bits de **datos** son enviados al receptor después del bit de comienzo
 - El bit de menos peso es transmitido primero
 - Un carácter de datos suele consistir en 7 ó 8 bits
 - Si en la configuración se ha seleccionado **paridad** se envía un bit de paridad
 - Se utiliza para detectar errores en los caracteres de datos
 - Finalmente se envían 1, 1.5 ó 2 bits de parada que identifican el **final**
- El puerto serie del PC es compatible con el estándar RS-232C





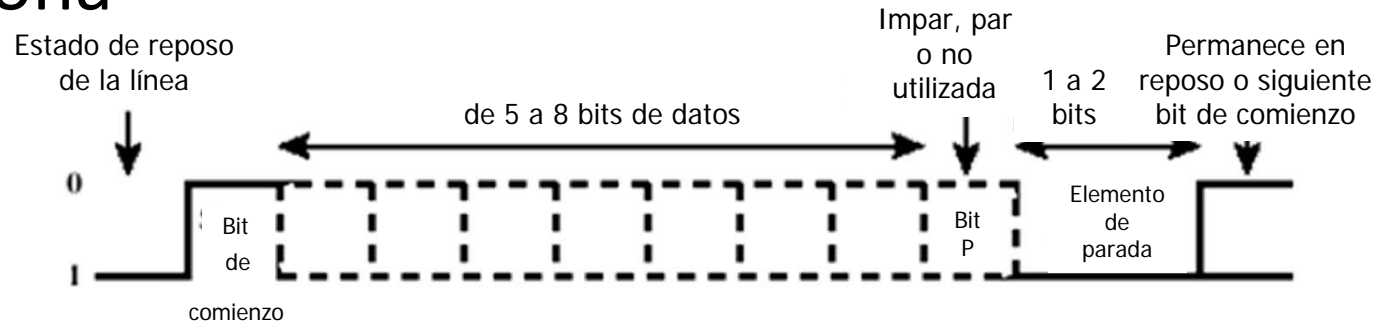
Introducción (II)

- Transmisión de datos
 - Problema
 - Para determinar el valor binario en la recepción de datos digitales se realiza un muestreo de la señal por cada bit recibido
 - Para que el receptor muestre los bits recibidos correctamente debe conocer el instante de llegada así como la duración de cada bit
 - Solución
 - Sincronización entre el emisor y el receptor
 - Dos métodos
 - **Transmisión asíncrona**
 - Los datos se transmiten enviándolos carácter a carácter
 - » De 5 a 8 bits
 - La temporización se debe mantener durante la duración del carácter
 - Resincronización al principio de cada nuevo carácter
 - **Transmisión síncrona**
 - Se transmite un bloque de bits sin códigos de comienzo o parada
 - Los relojes han de estar sincronizados

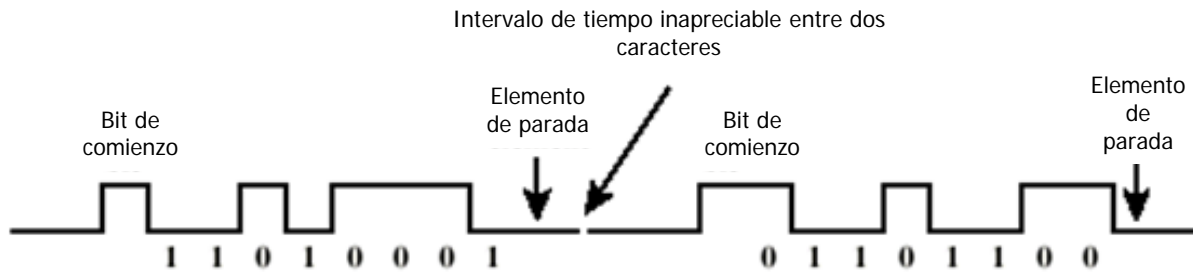


Introducción (y III)

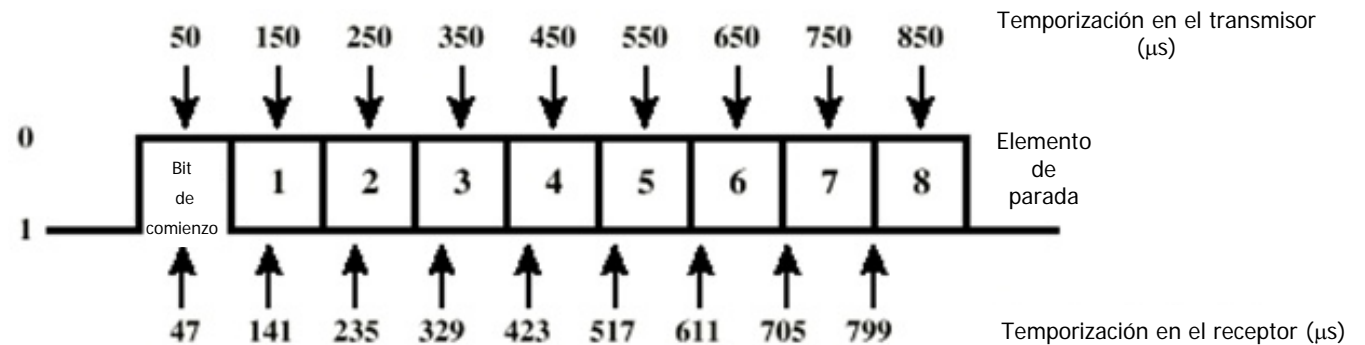
- Transmisión asíncrona



(a) Formato de un carácter



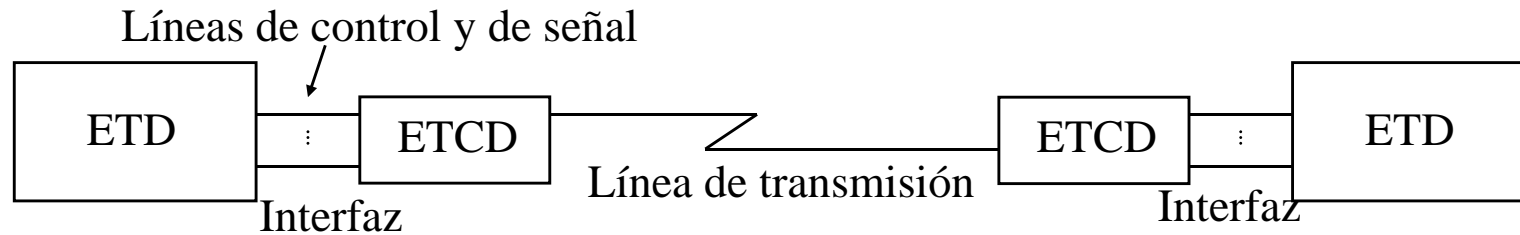
(b) Cadena de caracteres de 8 bits asíncronos



(c) Efecto de un error en la temporización



Normalización del interfaz (I)



⊗ Planteamiento inicial

- ETD utiliza el medio de transmisión a través del ETCD
- La línea de transmisión no está disponible para el usuario y no dispone de las características adecuadas

⊗ Etapas en la Transmisión de Datos

- *Establecimiento de la conexión* si el circuito de datos no es exclusivo entre ellos (Red conmutada)
- *Iniciación* para adaptar los ETCD a la línea de transmisión
- *Transmisión de la información* hasta que uno de los ETD decida finalizar
- *Liberación del circuito*

⊗ Para la realización de estas etapas se necesita un intercambio de señales entre ETD y ETCD a través de unos circuitos (control y señal)





Normalización del interfaz (y II)

- Normalización del interfaz
 - Distintas normas del UIT-T y EIA (Asociación de Industrias Electrónicas) especifican la naturaleza de la interfaz
 - Describen las características:
 - mecánicas
 - define el tipo de conector y sus dimensiones, el número de contactos metálicos, su distribución y dimensiones, etc.
 - eléctricas
 - relacionadas con los niveles de voltaje y su temporización
 - ETD y ETCD deben usar el mismo código, los mismos niveles de voltaje y la misma duración de los elementos de señal (determina la velocidad y la distancia)
 - funcionales
 - define la función que tiene encomendada cada uno de los contactos (pines)
 - de procedimiento
 - especifican la secuencia de eventos que se debe producir en la transmisión de datos, basándose en las características funcionales del interfaz





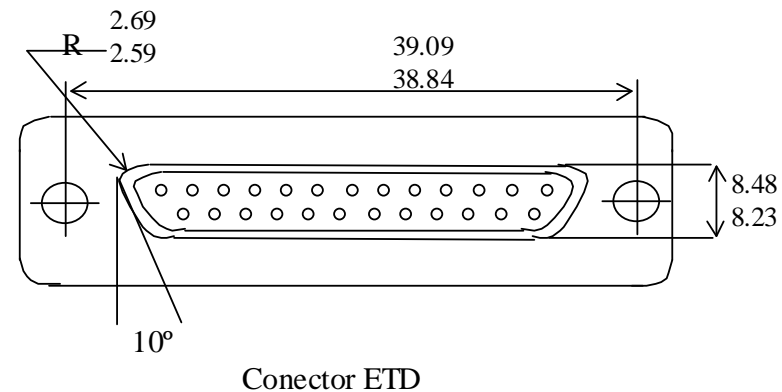
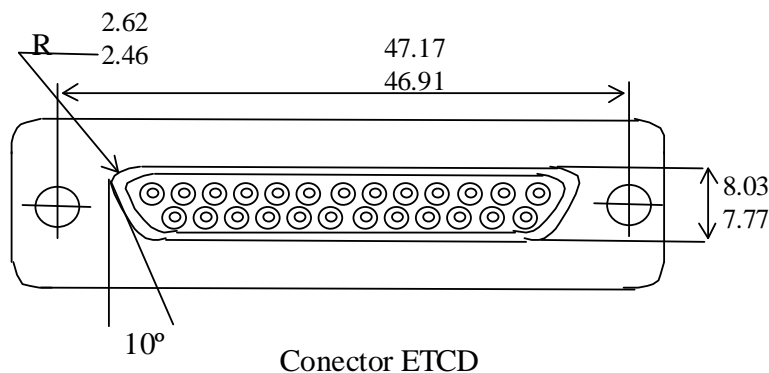
Norma EIA RS-232-C

- Antecedentes
 - En los 60s la EIA establece la norma RS-232 para la conexión ente ETD-ETCD
 - Posteriormente después de varias revisiones se convierte en la RS-232-C
 - Equivalente a normas UIT-T ISO 2100 (mecánicas) + V.28 (eléctricas) + V.24(funcionales)
 - Norma más utilizada para la conexión de terminales a MODEMS
 - Presenta problemas que tratan de resolver recomendaciones posteriores (EIA232E 1991)
- Características
 - Interfaz bipolar ($\pm 15V$) no equilibrado para la transmisión de señales digitales en banda base en el que todas las señales utilizan hilos separados para su envío y un retorno común
- Limitaciones
 - Velocidad máxima 20 Kbps y longitud máxima del cable de 15 m
- Aplicaciones
 - Líneas punto a punto, multipunto, red conmutada y privadas, a dos o cuatro hilos, en modo síncrono o asíncrono y en half o full-duplex



Norma RS-232: características mecánicas

- Conexión física ETD/ETCD (ISO 2110)
 - Cable terminado en cada extremo por conectores tipo *Cannon* de 25 contactos (DB-25) o 9 contactos (DB-9)
 - Consta de dos filas de terminales: una con 13 y otra con 12
 - Es asimétrico → impide su incorrecta conexión
 - ETD conector tipo "macho"
 - Modem conector tipo "hembra"
 - Los conectores del cable serán los complementarios



Norma RS-232: características eléctricas

- Voltaje máximo bipolar $\pm 15V$ en cualquier circuito

- Entradas de datos

- “1” si $V < -3V$
- “0” si $V > +3V$

- Entradas de control

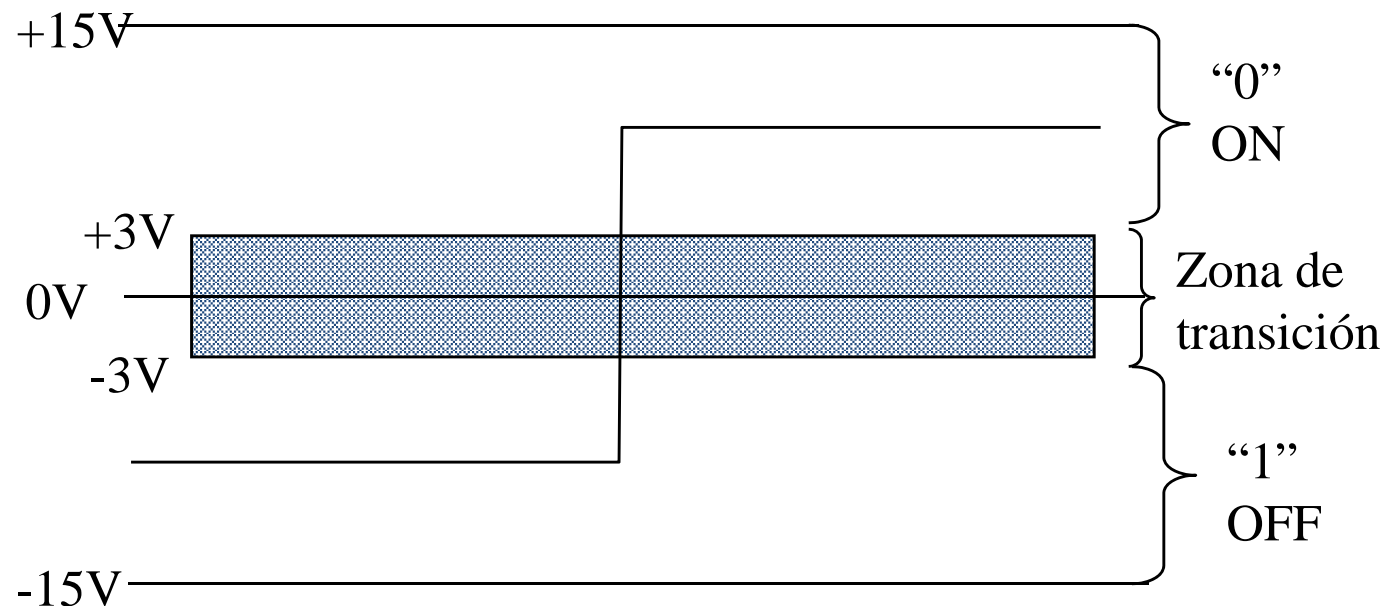
- ON si $V > +3V$
- OFF si $V < -3V$

- Salidas de datos

- “1” si $V < -5V$
- “0” si $V > +5V$

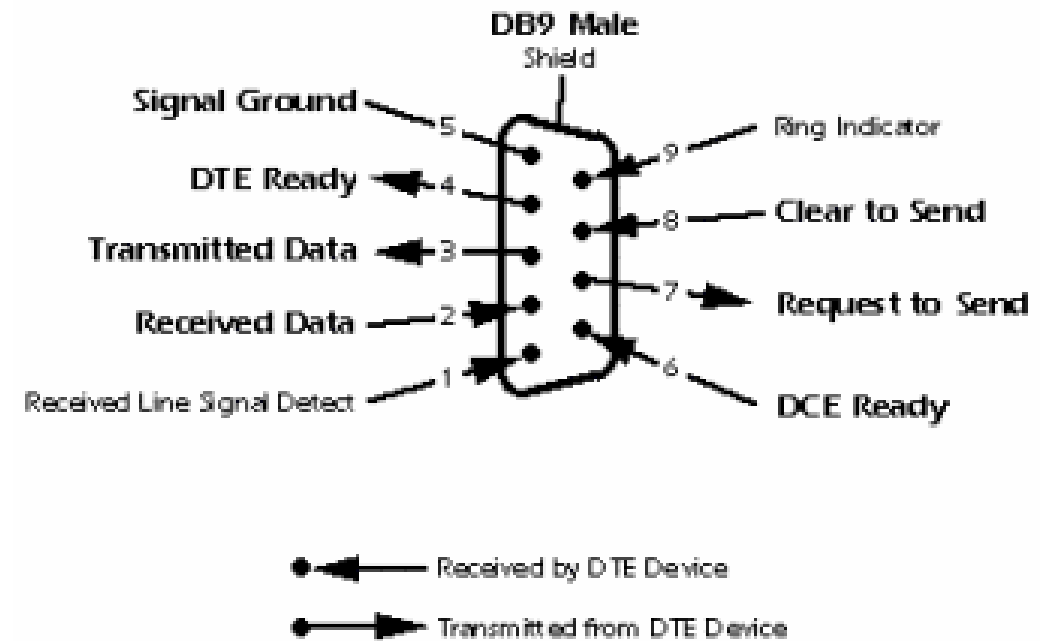
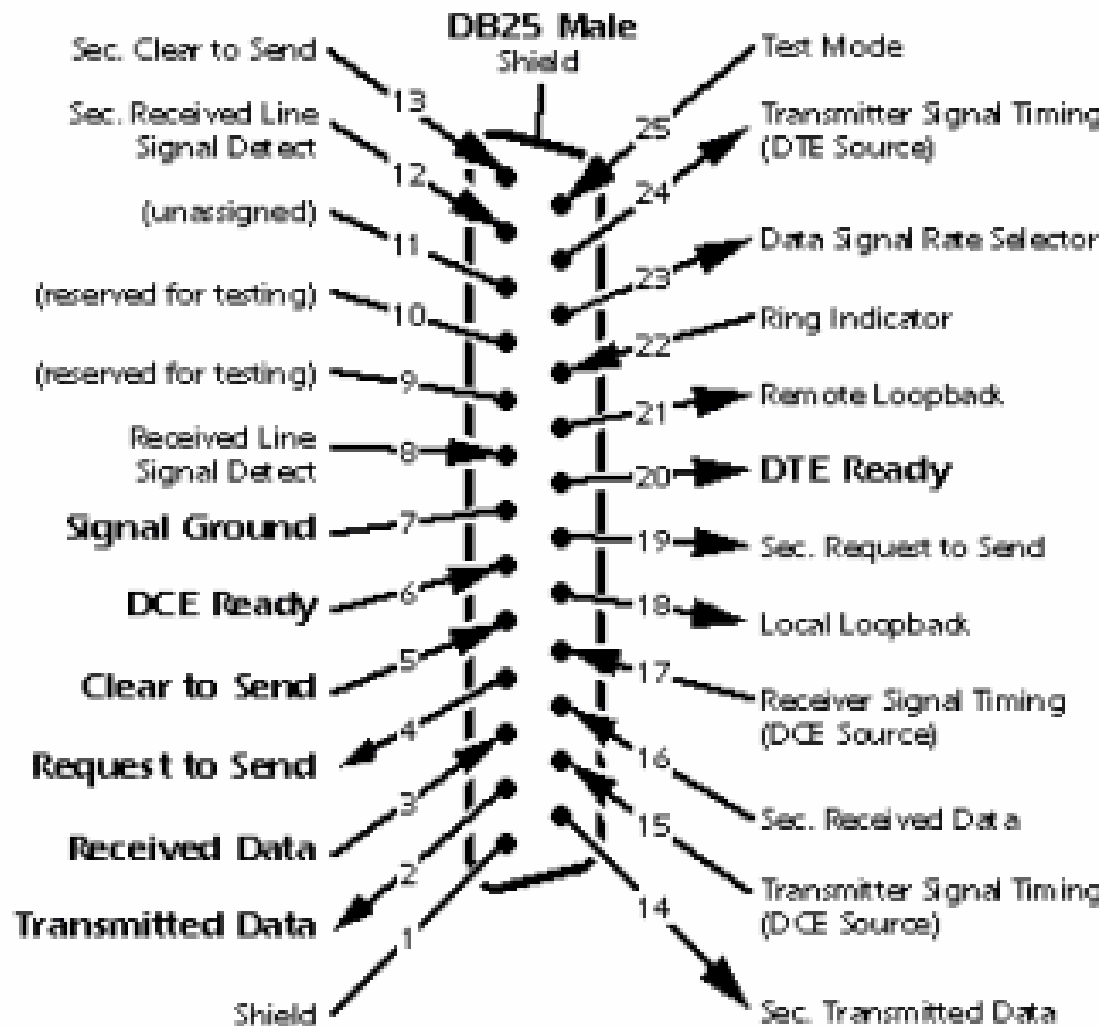
- Salidas de control

- ON si $V > +5V$
- OFF si $V < -5V$



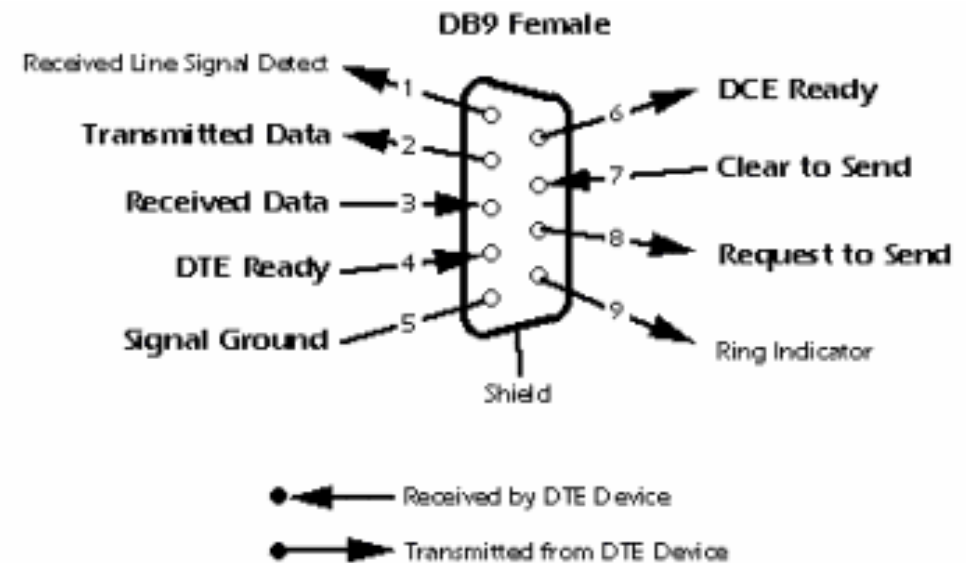
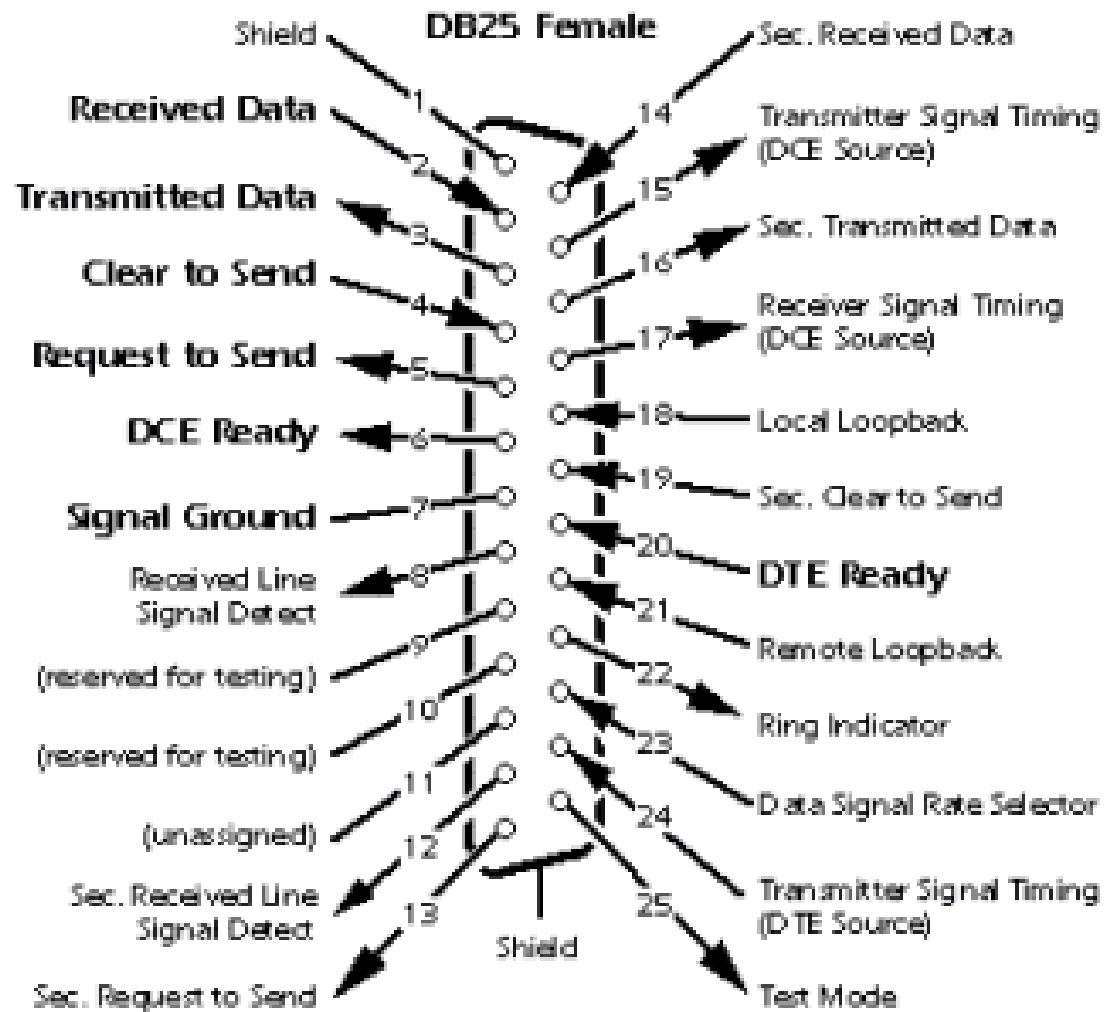
Norma RS-232: características funcionales (I)

Looking Into the DTE Device Connector



Norma RS-232: características funcionales (II)

Looking Into the DCE Device Connector



Norma RS-232: características funcionales (III)

DB-25 Pin #	DB-9 Pin #	SENTIDO DTE-DCE	FUNCIÓN
1	-		Protective_Ground
7	5		Signal_Ground (SG)
2	3	→	Transmitted_Data (TD)
3	2	←	Received_Data (RD)
4	7	→	Request_to send (RTS)
5	8	←	Clear_to send (CTS)
6	6	←	Data set_ready (DSR)
20	4	→	Data terminal_ready (DTR)
22	9	←	Ring_detector (RI)
8	1	←	Carrier_detect (CD)
21	-	←	Signal quality_detect (SQ)
23	-		Data signal_rate selector
24	-	→	Transmitter signal_timing (DTE) (TC)
15	-	←	Transmitter signal_timing (DCE) (TC)
17	-	←	Receiver signal_timing





Norma RS-232: características funcionales (IV)

- CIRCUITOS DE TIERRA /RETORNO

- **Protective ground**

- Conectado eléctricamente a la tierra del conector de alimentación de red
 - Conexión a un extremo de la protección si se usa cable protegido
 - Minimiza interferencias en entorno con alto nivel de ruido
 - La protección no debe conectarse a ambos extremos del cable

- **Signal ground**

- Constituye el hilo de vuelta común de todas las demás señales
 - Debe conectarse en ambos extremos

- CIRCUITOS DE DATOS

- **Transmitted data**



- Se transfieren las señales de datos generadas por el ETD para ser enviadas

- **Received data**



- Se reciben los datos originados en el ETCD



Norma RS-232: características funcionales (V)

- CIRCUITOS DE CONTROL

- **Data Terminal Ready**



- Indica que el ETD está activo y hace que el ETCD se conecte a la línea
 - Debe activarse antes de que el ETCD active **DSR**

- **Data Set Ready**



- El ETCD indica que está conectado a la línea (esta en modo de transmisión, no en modo de marcado o test) y esta listo para el intercambio de señales

- **Request To Send**



- Originado por el ETD, cuando está activo, hace que el ETCD pase al modo de transmisión y estado inactivo al de no transmisión

- **Clear To Send**



- El ETCD indica al ETD que está listo para recibir datos y enviarlos a la línea

- **Ring Indicator**



- El ETCD le dice al ETD que ha recibido una llamada del terminal remoto
 - Después el ETD le dice al ETCD que conteste activando **DTR**





Norma RS-232: características funcionales (y VI)

- CIRCUITOS DE CONTROL

- **Data Carrier Detect**



- El ETCD indica que la línea telefónica ha sido activada por el extremo remoto y que pueden llegar datos en cualquier momento

- **Signal quality**



- El ETCD indica, cuando está activo que no se detectan errores en la línea y cuando está inactivo que la tasa de errores es superior a la nominal

- Proceso de intercambio de señales

- Establecimiento de la conexión

- ETD activa DTR y después el MODEM activa DSR

- Inicialización

- ETD activa RTS y si no hay ningún problema el MODEM activa CTS



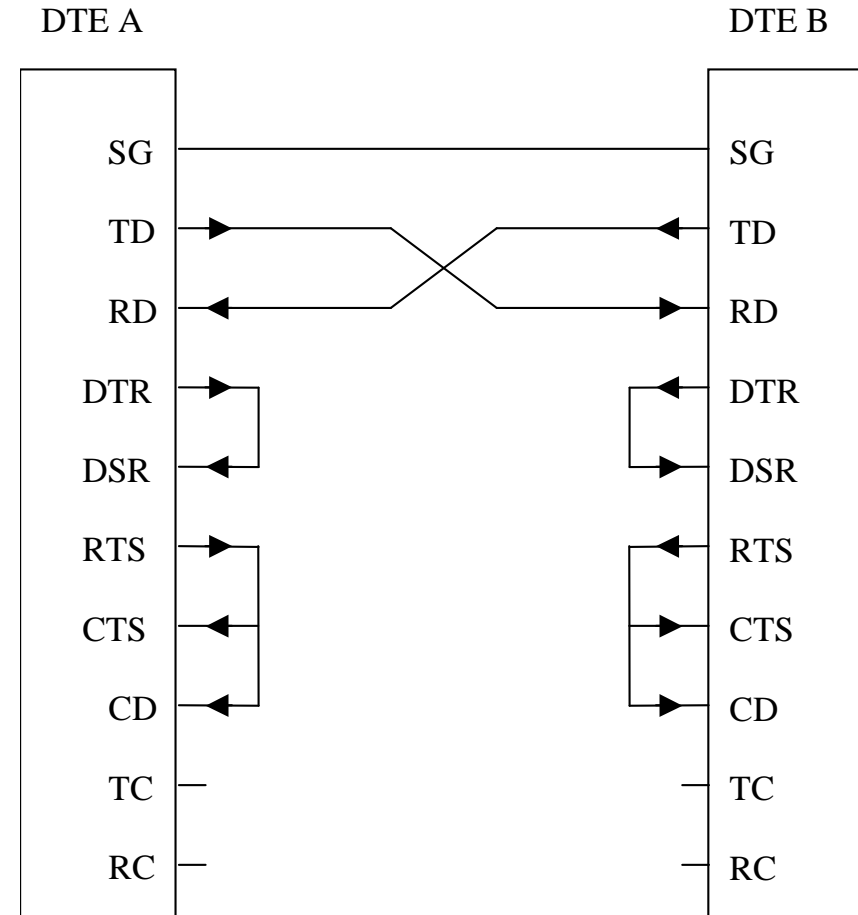
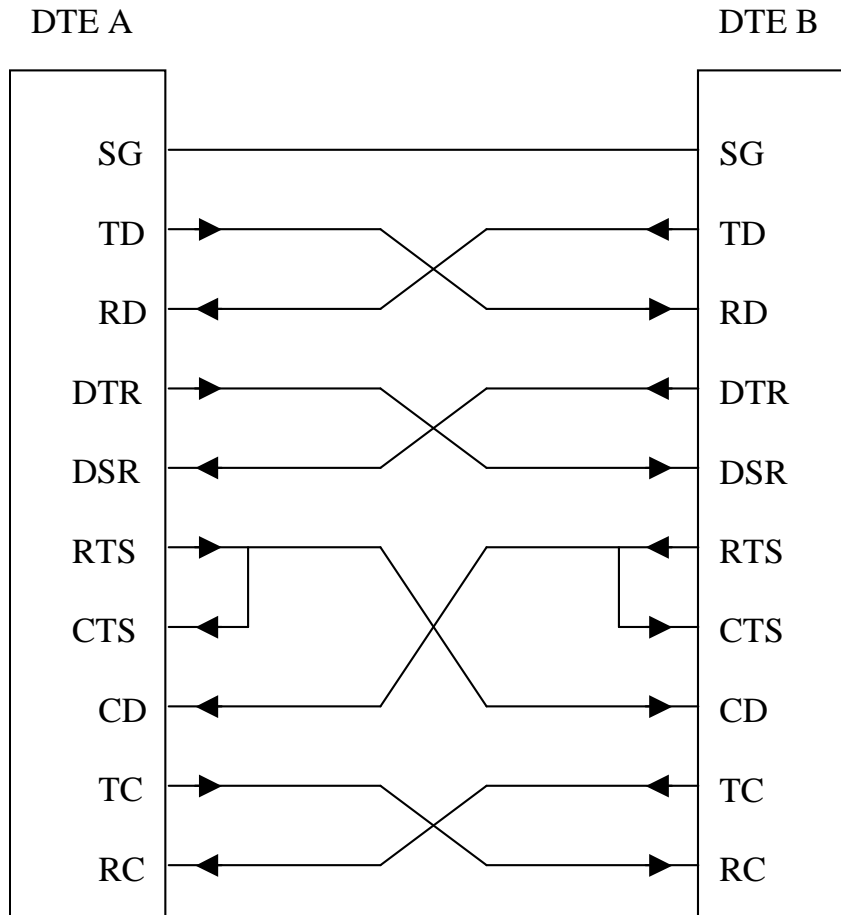


Adaptación para la conexión ETD-ETD (I)

- Planteamiento
 - Norma RS-232-C pensada para conectar ETD (ordenador personal) con MODEM (ETCD)
 - No esta pensada para conectar dos ETD (ordenadores) entre sí
- Se necesita un cable especial (Modem Nulo)
 - Se cruzan determinados hilos para que los dos ETDs trabajen como si tuvieran un ETCD en el otro extremo
 - Cruzar los terminales RD y TD para que lo que uno considera datos transmitidos el otro los considere recibidos
 - Cruzar los terminales DTR y DSR
 - Puentear RTS y CTS
 - Cuando el ETD A desea transmitir activa la línea RTS, que se devuelve a sí mismo como CTS para simular la presencia del modem
 - Conectar RTS a CD
 - Cuando el ETD A desea transmitir activa la línea RTS, por lo que se activa la línea CD del ETD B que lo interpreta como que le van a llegar datos
 - En transmisión síncrona también se cruzan la señales de reloj
- La solución no es única

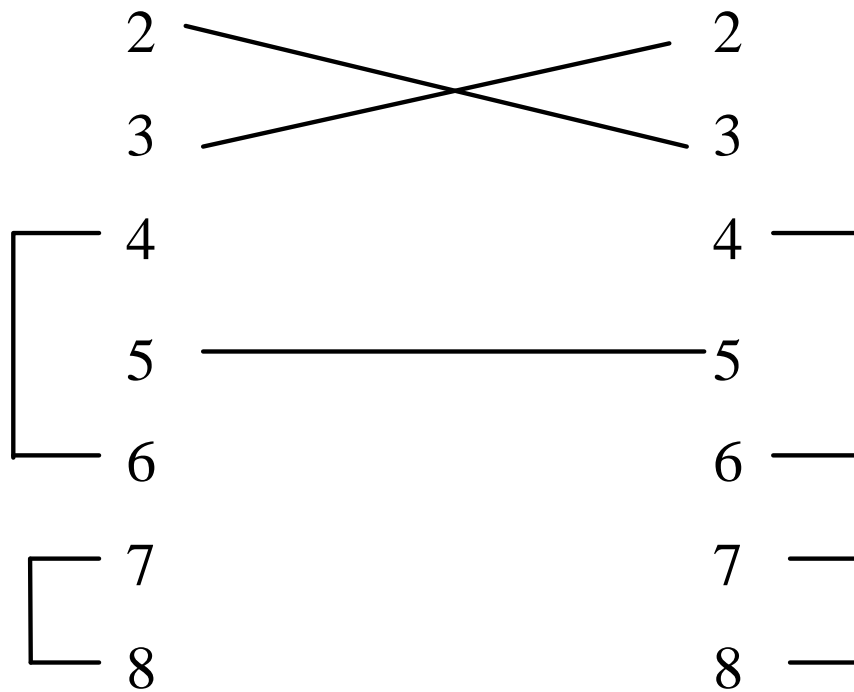


Adaptación para la conexión ETD-ETD (II)

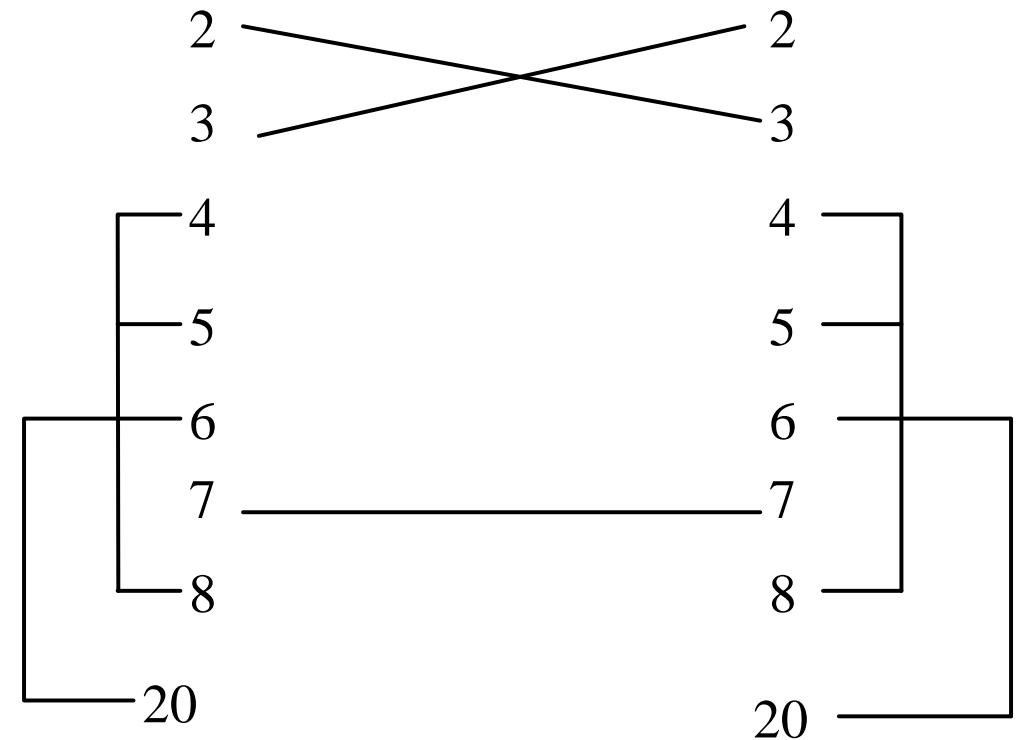




Adaptación para la conexión ETD-ETD (y III)



Conectores hembra 9 pines. Configuración mínima.

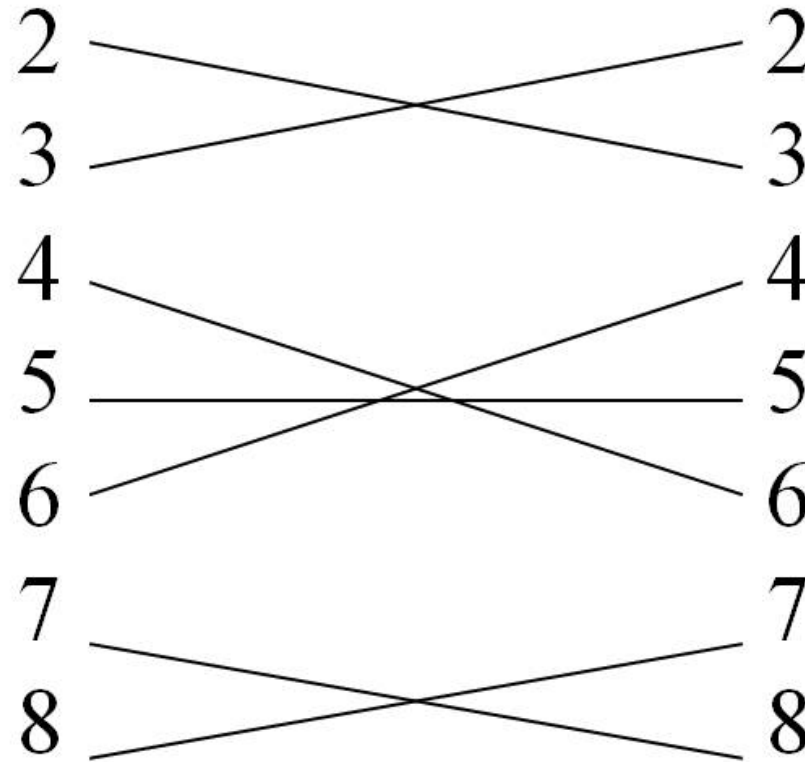


Conectores hembra 25 pines





Adaptación para la conexión ETD-ETD (y IV)



Conectores hembra 9 pines. Configuración recomendada.





Hardware de comunicaciones asíncronas

- UART (Transmisor/Receptor Asíncrono Universal)
 - Circuito integrado que convierte los datos de paralelo a serie y viceversa
 - La UART típica para un PC es el Intel 8251A
 - Puede ser programado para realizar comunicaciones serie síncronas o asíncronas
- Antecedentes
 - Hardware gestiona las funciones de recepción y transmisión de datos y varias de control
 - Programador liberado de realizar software para transmisión
 - Procesador se descarga de la realización de tareas de transmisión
 - Primeras UART surgieron en 1973 para simplificar los programas de comunicaciones
 - Programador liberado de la temporización de E/S asíncrona, des/ensamblar bytes, etc.
 - Procesador realiza otras tareas mientras se envían y reciben datos
 - Recepción/transmisión de la información se reduce a operaciones de E/S sobre puertos o memoria





Funciones de UART (I)

- Transmisión
 - Capturar el carácter que se desea enviar
 - Construir la **Unidad de Datos Serie (SDU)**
 - Bit de comienzo (o *Start*)
 - Bits de datos
 - Bit de paridad
 - Bits de parada (o *Stop*)
 - Serializar
 - Enviar a la línea
- Recepción
 - Detección del bit de *Start*, de paridad, de Stop
 - Detección de errores en la transmisión
 - Formación del carácter (paralelización)
 - Entrega del carácter





Funciones de UART (y II)

- Generales
 - Reloj
 - Ajuste/selección de:
 - Velocidad en baudios, nº de bits de un carácter, nº de bits de stop, paridad
 - Funciones de RS-232
 - Funciones de Interrupción (activación/desactivación)
 - de transmisión
 - buffer de transmisión vacío
 - buffer de retención vacío
 - de recepción
 - dato recibido disponible
 - dato recibido montado
 - de RS-232
 - de error de recepción o *break*
 - paridad
 - desbordamiento
 - error de formato





Diagrama de bloques del transmisor (I)

- *Configuración* para establecer el formato de los datos
 - bits de datos 5, 6, 7 y 8
 - bits de stop 1, 1.5 y 2
 - paridad habilitada (par o impar) deshabilitada
- *Capturar los datos* del bus del sistema
 - En el buffer de transmisión o registro de retención de transmisión se almacenan temporalmente los datos que se desean transmitir
 - Mientras se está enviando un dato se pueden escribir bytes en el registro de retención → aumenta la eficiencia de la transmisión.
- *Formación de la SDU* dependiendo de la configuración
- *Serialización* mediante un registro de desplazamiento de transmisión
 - Bits de SDU se cargan en paralelo en registro desplazamiento de transmisión
 - Se desplazan en las transiciones negativas del reloj de transmisión
 - Al finalizar la transmisión de una SDU se carga la siguiente



Diagrama de bloques del transmisor (II)

- Dos *flags* para informar del estado de la transmisión
 - TBE (*transmitter buffer empty*) buffer de transmisión vacío
 - Se activa cuando el dato del buffer de transmisión se carga en el registro de desplazamiento, indicando que la UART puede aceptar otro byte
 - TXE (*transmitter empty*) transmisor vacío
 - Cuando se han desplazado todos los bits excepto el de STOP final y el buffer de transmisión está vacío, en el siguiente ciclo de reloj), se activa TXE indicando que el registro de desplazamiento está vacío
 - En transmisiones full-duplex comprobar el estado de TBE
 - El software debe asegurarse de que el buffer está vacío antes de enviar otro byte a la UART
 - En transmisiones semi-duplex comprobar TBE y TXE
 - Antes de que el software ordene a un modem transmisor convertirse en receptor, se debe comprobar si están vacíos el buffer de transmisión y el registro de desplazamiento, para no perder bits de la última SDU
- Errores en la transmisión
 - Sobrescritura del transmisor (*transmitter overwrite error*)
 - Se intenta escribir en un buffer no vacío



Diagrama de bloques del transmisor (y III)

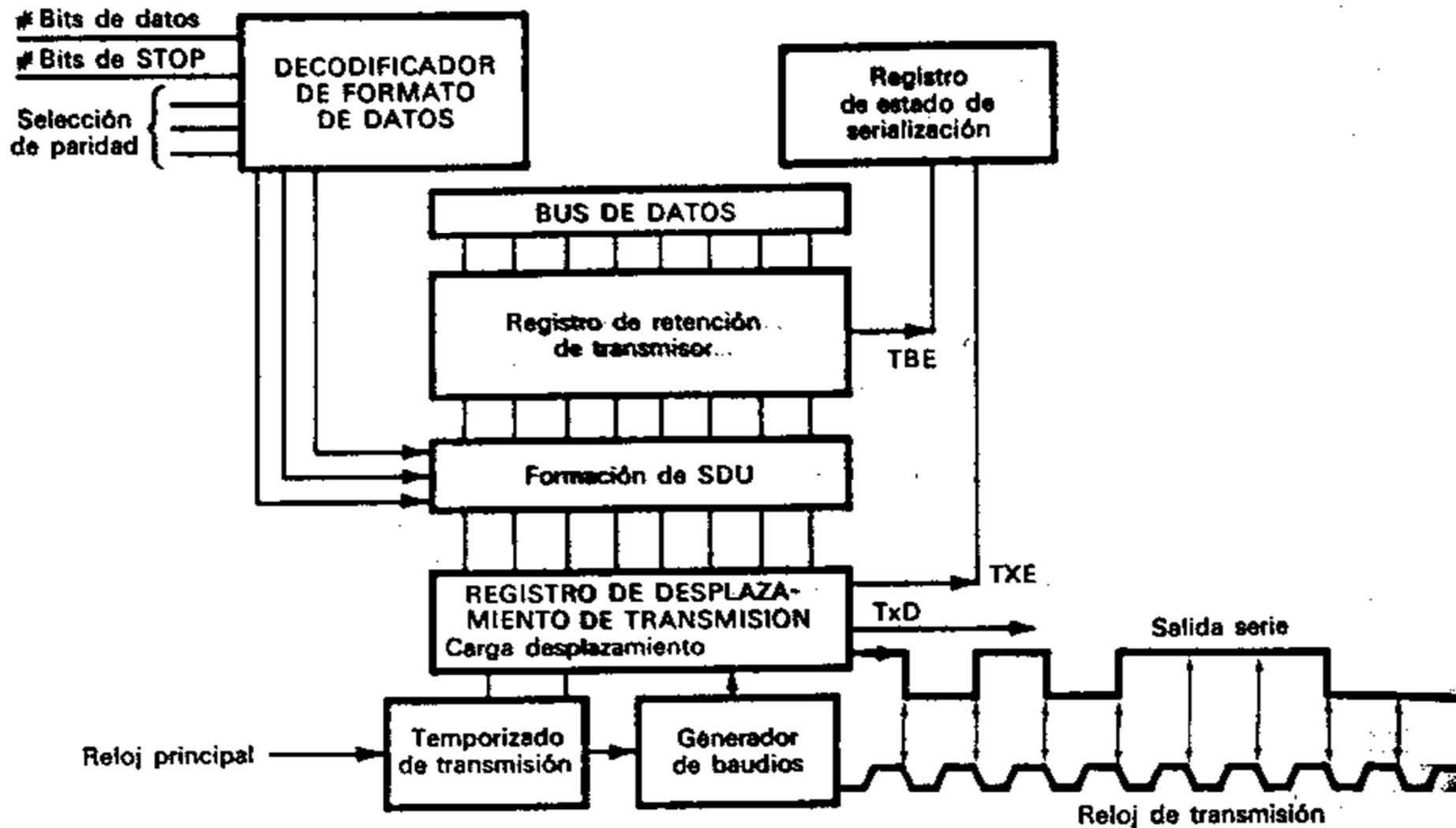




Diagrama de bloques del receptor (I)

- *Configuración* para establecer el formato de los datos
- *Muestreo de la línea* serie de entrada de forma continua
- *Paralelización* del dato
 - Cuando se detecta el bit de START, se van desplazando los bits siguientes al registro de desplazamiento
- *Análisis del dato*
 - De acuerdo con el formato establecido, se extrae de la SDU los bits de información, analizando si se ha producido un error en la recepción
- *Llenado del buffer de recepción*
 - Una vez construido el byte de datos se pasa a un buffer FIFO
 - En ese momento se activa el flag RxRDY y permanece activo hasta que todos los elementos de la FIFO han sido extraídos
 - Por cada elemento en la FIFO hay un registro de estado de serialización (*Serialization Status Register*).



Diagrama de bloques del receptor (II)

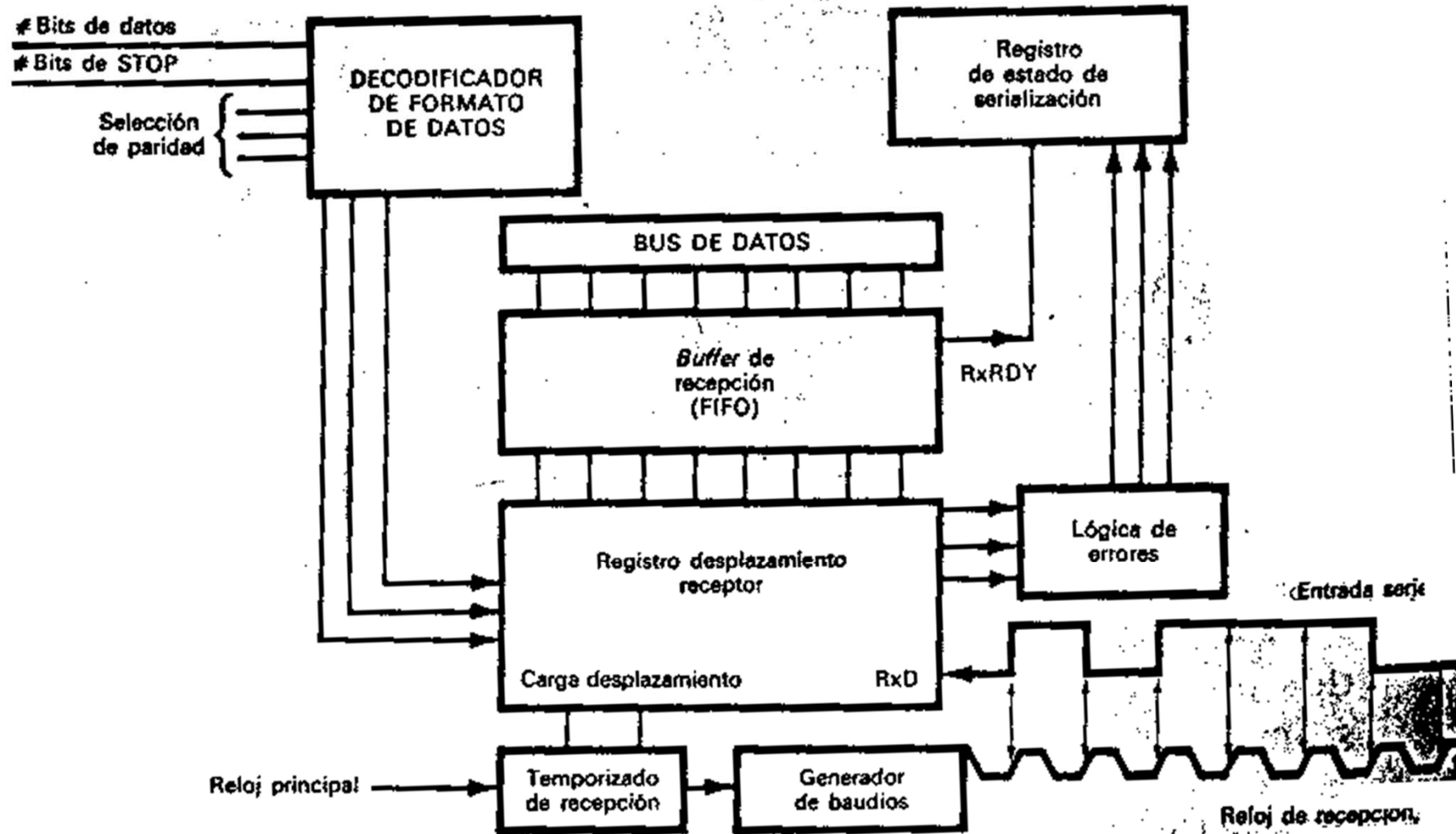




Diagrama de bloques del receptor (y III)

- *Errores en la recepción*
 - Sobrescritura del receptor (*receiver overrun*)
 - Cuando los bytes llegan más rápidamente de lo que se pueden leer se llena el buffer, escribiendo los nuevos bytes sobre los anteriores
 - Error de paridad
 - El bit de paridad no coincide con el esperado según los registros de formato de datos
 - Error de trama (*framing error*)
 - Al ensamblar el byte recibido su bit de STOP es incorrecto
 - Condición de BREAK
 - La Línea está a 0 durante un período mayor de una SDU





Comunicaciones por puerto serie

- Programación muy similar a la operación con ficheros tanto en Windows como en UNIX
 - En esencia el puerto serie se gestiona como cualquier otro dispositivo de E/S , por lo que será necesario el uso del API para abrir, escribir, leer y cerrar ficheros.
 - Diferencias con los ficheros normales
 - Es necesario configurarlo previamente para establecer valores como la velocidad y el modo de funcionamiento.
 - Se requiere de un control de errores para las comunicaciones.
- Modelos de programación
 - Síncrono
 - Emisor y receptor sincronizados de tal forma que cuando uno envía el otro recibe y viceversa.
 - Asíncrono
 - Mientras se esperan datos se pueden estar haciendo otras tareas.
 - Se requiere de algún mecanismo de programación asíncrono como por ejemplo señales (habitualmente en sistemas Unix) o eventos (habitualmente sistemas Windows).





Funciones de Windows (I)

- Los nombres que se dan a los dispositivos serie en Windows son:
 - COM1 (primer puerto serie nativo), COM2 (segundo puerto serie nativo), ...
 - En caso de usar un convertidor USB-Serie, Windows hace corresponder dicho controlador USB con un COM libre (p. ej.: COM9).
- Funciones básicas del API de ficheros
 - [CreateFile](#), [WriteFile](#) y [ReadFile](#): Obtiene un manejador para operar con el puerto serie así como para escribir y leer datos por él.





Funciones de Windows (II)

- Estructura relacionada con la configuración del puerto serie
 - *Device-Control Block* (DCB): permite especificar valores para los parámetros de configuración del puerto serie: velocidad, bits de datos, paridad, bits de parada, control de flujo, etc.
- Funciones básicas para configurar el puerto serie
 - [BuildCommDCB](#): Rellena una estructura DCB según una cadena con un formato determinado. También [BuildCommDCBAndTimeouts](#) que permite además especificar los *timeouts* de lectura y escritura.
 - [GetCommState](#) y [SetCommState](#): Recupera y actualiza respectivamente la configuración del puerto serie en base a la estructura DCB.
 - [GetCommTimeouts](#) y [SetCommTimeouts](#): Recupera y actualiza respectivamente los *timeouts* de lectura y escritura.





Funciones de Windows (III)

- Funciones para la gestión de los eventos del puerto serie
 - [SetCommMask](#): Actualiza el conjunto de eventos que van monitorizarse.
 - [WaitCommEvent](#): Comprueba si se ha notificado algún evento de los que se están monitorizando.
 - [GetCommMask](#): Recupera el valor del conjunto de eventos que se han notificado (útil cuando se están monitorizando más de un evento). Una vez recuperado deberá compararse la máscara con cada evento para ver cuál se ha notificado.





Funciones de Windows (y VI)

- Funciones para el control de errores
 - [ClearCommError](#): Recupera información sobre el error producido en la comunicación junto con el estado actual del dispositivo. Limpia el *flag* de error y habilita las operaciones de E/S sobre el dispositivo en cuestión.
 - [SetCommBreak](#): Suspende la transmisión de caracteres para un dispositivo de comunicaciones y pone la línea a cero (*break state*).
 - [ClearCommBreak](#): Restaura la transmisión de caracteres para un dispositivo de comunicaciones y levanta la línea (*nonbreak state*).





Funciones de UNIX

- Los nombres que se dan a los dispositivos serie en Linux/Unix son:
 - `/dev/ttyS0` (primer puerto serie nativo, equivalente al COM1),
`/dev/ttyS1` (segundo puerto serie nativo, equivalente al COM2), ...
 - `/dev/ttyUSB0` (convertidor USB-Serie 1), `/dev/ttyUSB1` (convertidor USB-Serie 2), ...
- Llamadas al sistema para operar con ficheros
 - *open* y *close*: obtiene y libera respectivamente un descriptor asociado (en este caso) al puerto serie.
 - *read* y *write*: lee y recibe datos respectivamente del descriptor indicado.





Funciones de UNIX (y II)

- Estructura relacionada con la configuración del puerto serie
 - *struct termios*: permite especificar valores para los parámetros de configuración del puerto serie: velocidad, bits de datos, paridad, bits de parada, control de flujo, etc.
- Funciones básicas para configurar el puerto serie
 - *cfsetospeed* y *cfsetispeed*: actualizan la velocidad de entrada y salida.
 - *tcflush*: limpiar los buffers de entrada (TCIFLUSH) y salida (TCOFLUSH).
 - *tcsetattr*: configura el puerto serie acorde con la estructura *termios*.



Otras aplicaciones del cable de modem nulo

- Puerto de gestión en equipos de redes como conmutadores, enrutadores, cortafuegos, etc

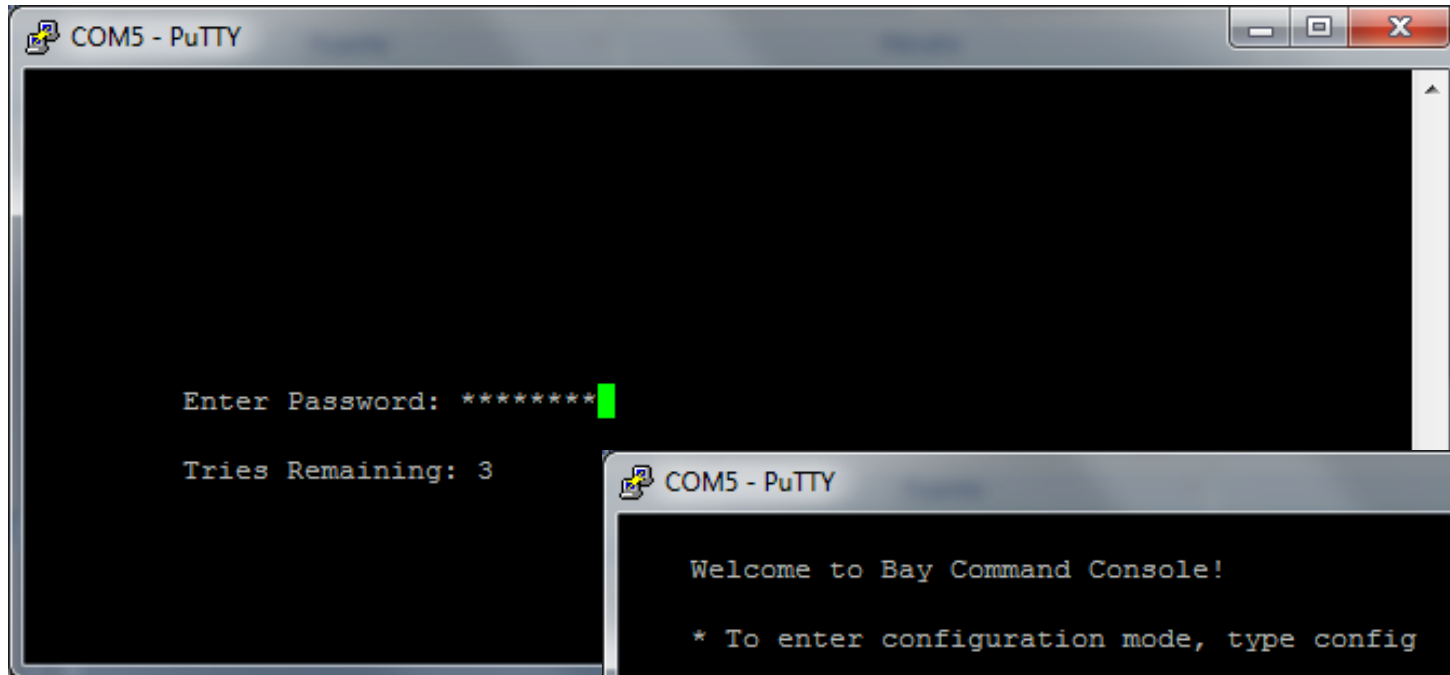


Router Nautica 200
Bay Networks, 1997



HP Procurve 24-Port Switch
Hewlett-Packard, 2007

Acceso a un *router* por el puerto de gestión (I)

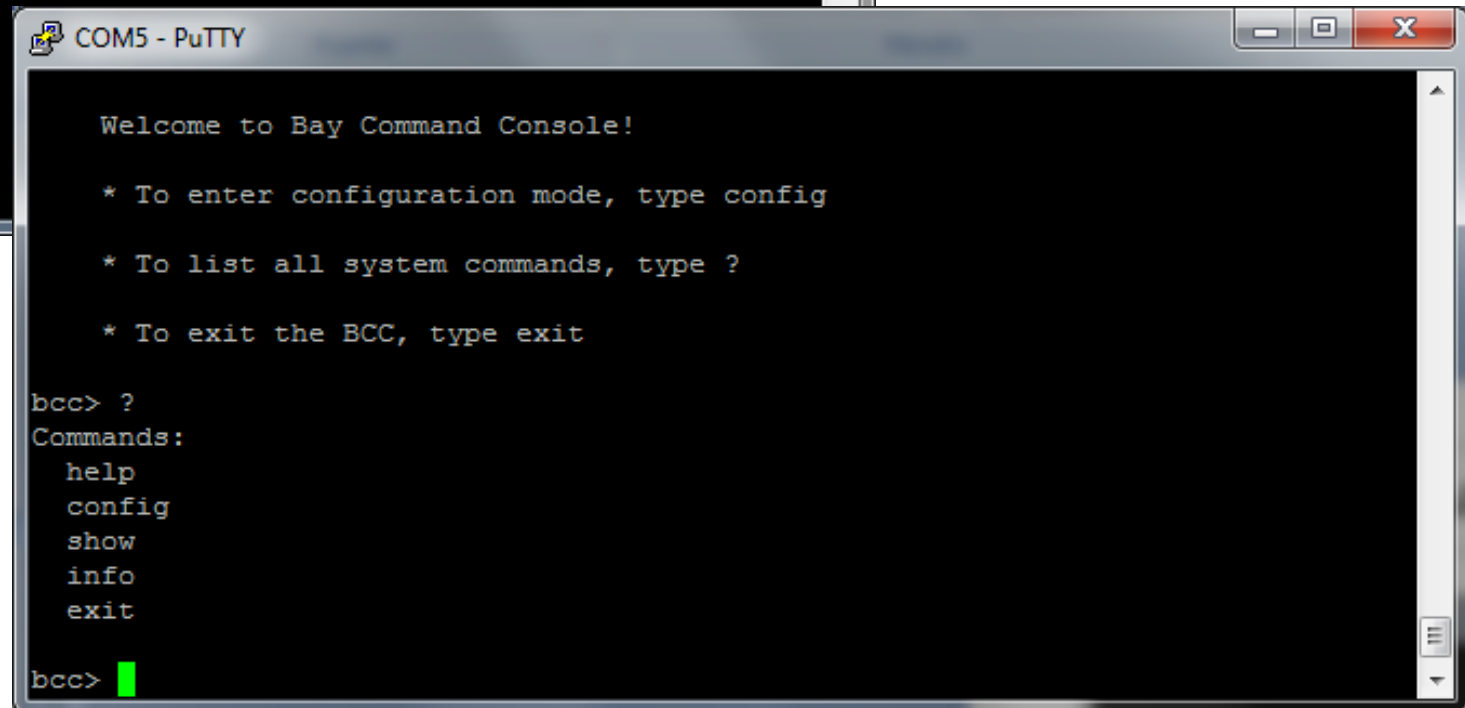


```
COM5 - PuTTY

Enter Password: *****
Tries Remaining: 3
```

El *router* se conecta al ordenador por medio de un cable de modem nulo desde el puerto de consola a un puerto serie libre del ordenador.

Es necesario pasar un proceso de autenticación antes de acceder al menú de administración del router. Cada router dispone de una interfaz de administración a medida.



```
COM5 - PuTTY

Welcome to Bay Command Console!

* To enter configuration mode, type config
* To list all system commands, type ?
* To exit the BCC, type exit

bcc> ?
Commands:
  help
  config
  show
  info
  exit

bcc>
```



Acceso a un *router* por el puerto de gestión (y II)

```
COM5 - PuTTY
bcc> show config
  box
    system
      name "ppay"
      password \
BC42814E048119D7E65A2FBE870857B8386B\
21FA0DE62AB5548E00F17A85285EB3835E9B
mgr-timeout 300
blacklist-timeout 20
secure-unit no
advanced-manager no
html-manager no
pptp-enable no
dhcp-enable no
bootp-relay-enable no
bootp-relay-server 255.255.255.255
syscontact "NoContact"
syslocation "NoLocation"
```

Nombre del dispositivo (ppay), contraseña de administración (encriptada y codificada), ...

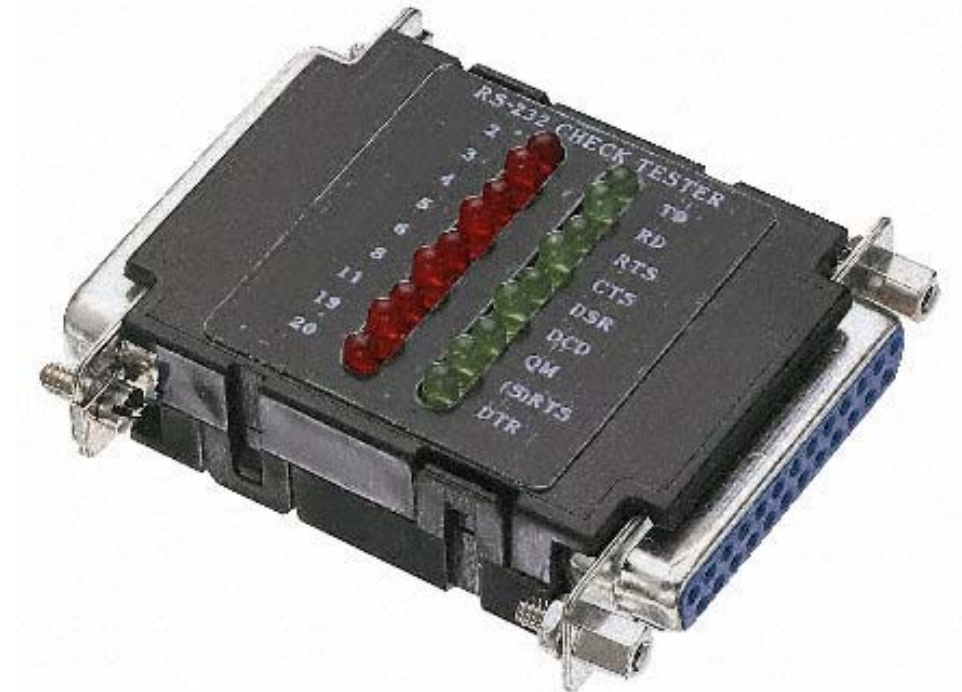
Dirección de red para la interfaz ethernet LAN01 (89.5.0.150), máscara de subred (24 bits), ...

```
COM5 - PuTTY
path type eth name "LAN01"
  ipaddr 89.5.0.150
  ipmasklen 24
  ip-rip-direction none
  rip-support rip1compat
  rip-type splithorizon
  enable yes
  bootp-relay no
  iplearn no
  dns-use-if-up no
  dns-addr1 0.0.0.0
  dns-addr2 0.0.0.0
  dns-addr3 0.0.0.0
CWC ..
path type ppp name "aypp"
  isdn-number "1/923280990"
  isdn-number "2/"
Type: <space> to page; <return> advance 1 line; Q to quit
```



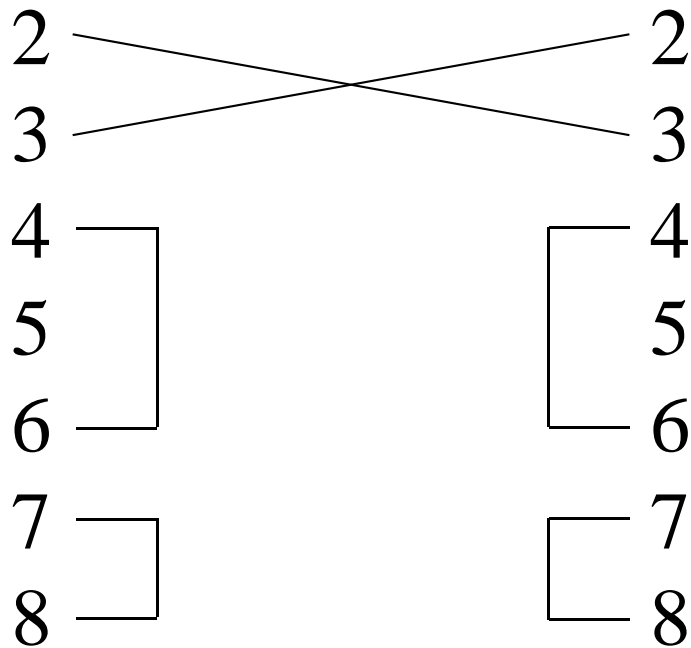
RS232C Check: Introducción

- RS-232 Monitor (tester)
 - Comprueba el estado de cada PIN ante posibles problemas de conectividad en la interfaz EIA RS-232C (V24)
 - Columnas separadas de LEDs rojos y verdes para asegurar una correcta monitorización
 - Rojos cuando existe un voltaje positivo
 - Verdes cuando el voltaje negativo

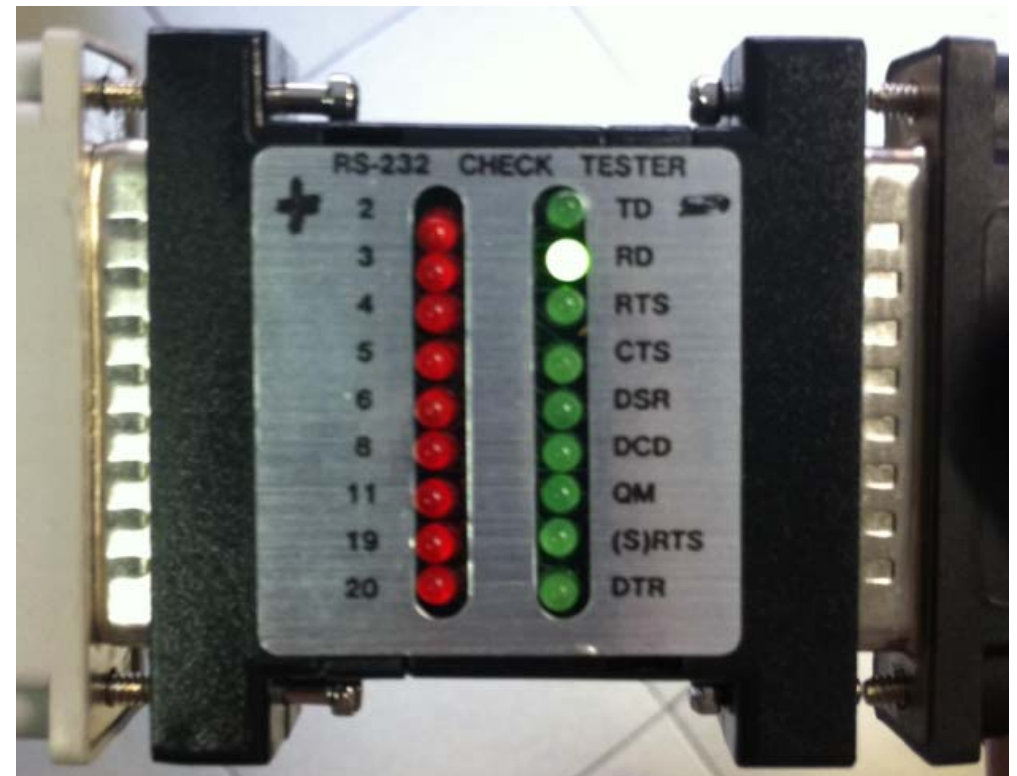


RS232C Check: Comprobar el cable (I)

- Versión reducida del cable de módem nulo



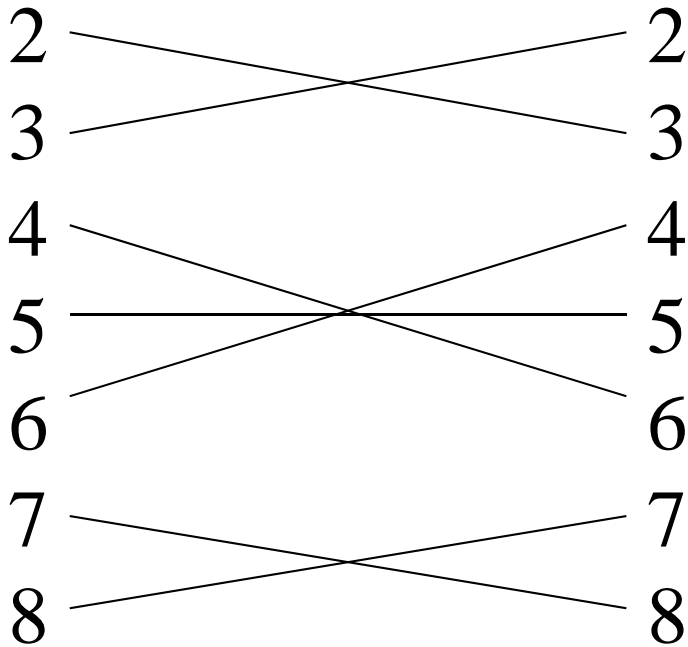
Cable de modem nulo



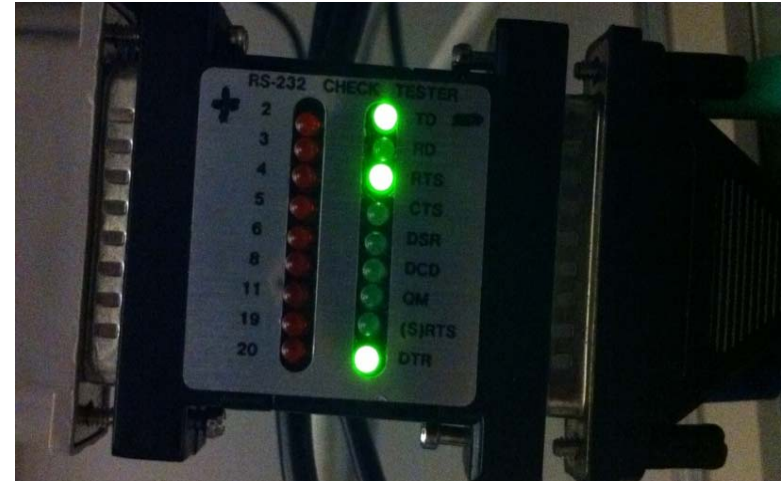
L-Izq desconectado y L-Dcho conectado

RS232C Check: Comprobar el cable (II)

- Versión ampliada del cable de módem nulo



Cable de modem nulo



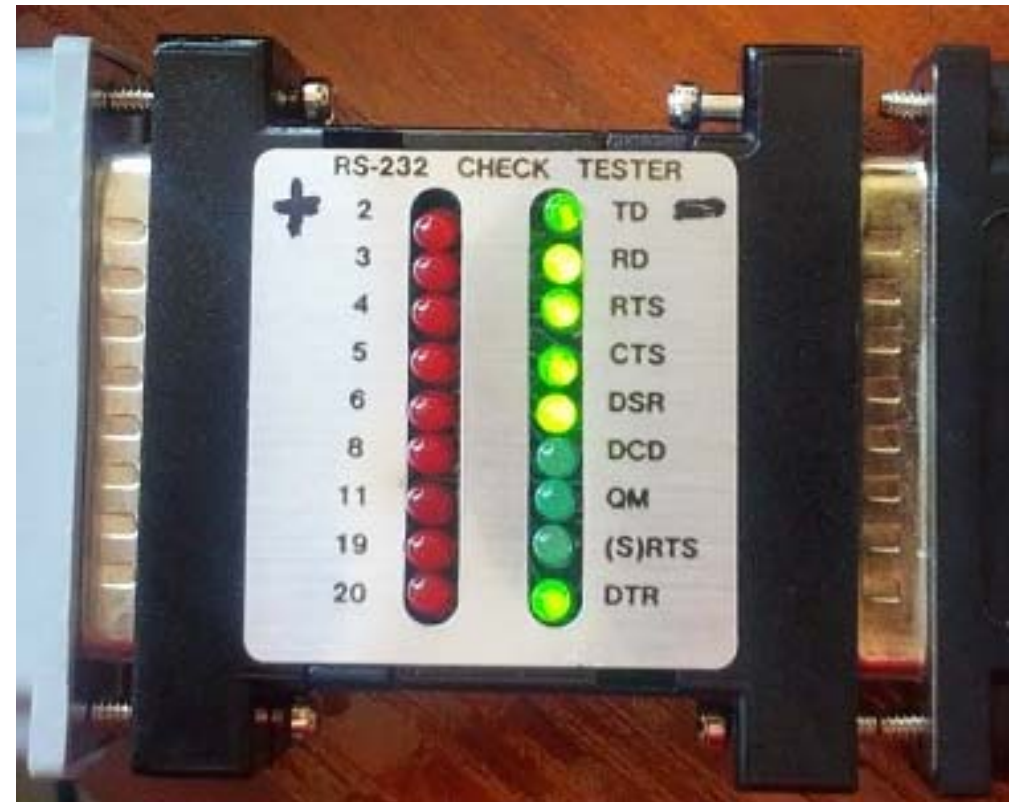
L-Izq conectado y L-Dcho desconectado



L-Izq desconectado y L-Dcho conectado

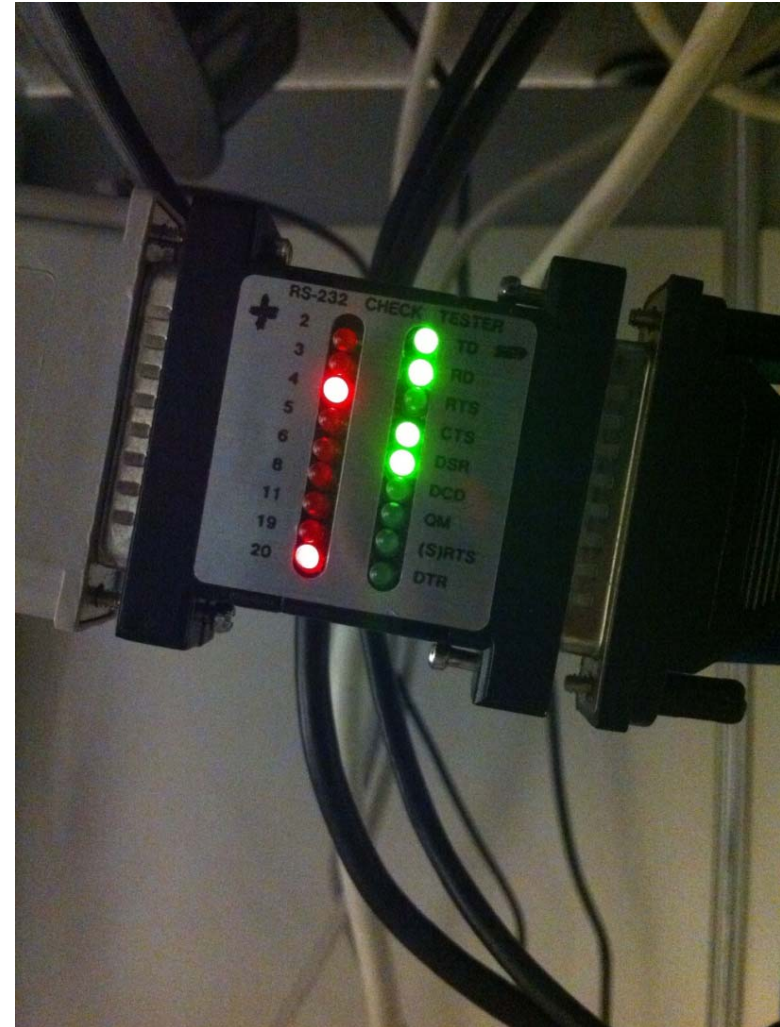
RS232C Check: Comprobar el cable (y III)

- Conectado a ambos lados
 - El monitor activa los LEDs de uno y otro lado de forma conjunta



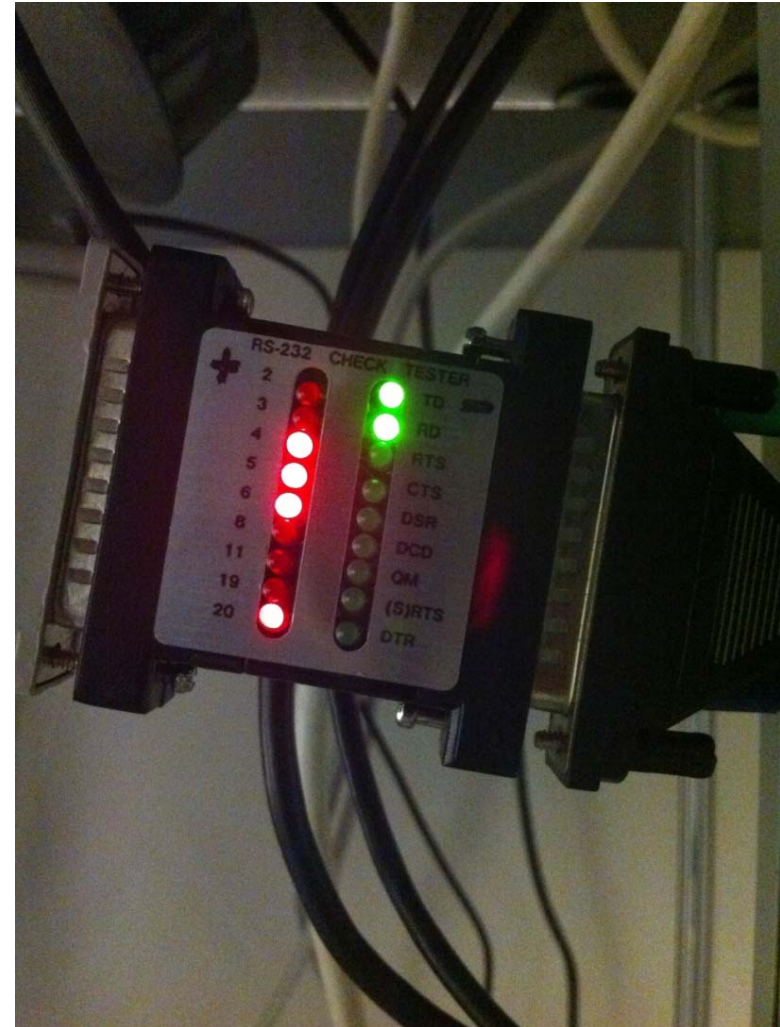
RS232C Check: Putty abierto (I)

- L-Izq (+) abre una sesión por el puerto serie (putty)
 - Se activa el 20 (*Data Terminal Ready, DTR*) para solicitar **establecimiento de conexión**
 - Se activa el 4 (*Request To Send, RTS*) para **iniciar la comunicación**



RS232C Check: Putty abierto (II)

- L-Dcho (-) abre otra sesión por el puerto serie (putty)
 - Se enciende el 6 (*Data Set Ready, DSR*) para confirmar el establecimiento de conexión
 - Se enciende el 5 (*Clear To Send, CTS*) para confirmar el inicio del envío de datos



RS232C Check: Transmitiendo

- L-Izq (+) y L-Dcho (-) escriben datos en sus terminales del puerto serie (putty)
 - Se enciende el 2 (*Transmitted Data, TD*) al escribir en L-Izq
 - Se enciende el 3 (*Received Data, RD*) al escribir en L-Dcho

