UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

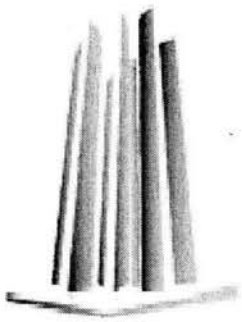
ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
CAMPUS: ARAGÓN

"DISEÑO DE UN SISTEMA DE ADQUISICIÓN DE DATOS A TRAVÉS DEL PUERTO SERIE PARA EL MONITOREO DE LA ESTACIÓN METEOROLÓGICA DEL PROTOTIPO ARAGONÉS 'TIM'".

T E S I S

PARA OBTENER EL TITULO DE
INGENIERO MECÁNICO ELECTRICISTA
P R E S E N T A N :

E D G A R A L O N S O T R U E B A
JESÚS ALONSO REBOLLAR PEDROZA



ASESOR: ING. ENRIQUE GARCÍA GUZMÁN

SAN JUAN DE ARAGÓN, ESTADO DE MÉXICO 2004

AGRADECIMIENTOS:

A la **UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO,** a la **ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGÓN** por habernos dado una formación integral y por haberme permitido conocer a mi compañero tesista.

A **DIOS** por habernos ofrecido esta oportunidad de vivir, por sus retos y desafíos, y sobre todo por darnos el valor y la fuerza necesaria para enfrentarlos.

Al **Ing. Enrique García Guzmán** por habernos asesorado en nuestro proyecto de Tesis, ya que sin su ayuda este sueño no hubiera sido posible.

A los **MIEMBROS DEL SÍNODO**, por brindarnos el mayor reto para un universitario, el EXAMEN PROFESIONAL.

EDGAR (CUN)

A mis padres:
Edith Trueba Perez
Carmen Alonso Castro (q.e.p.d.)

A mi familia:
Araceli, Alma, Jesús, Javier, Pily, Maribel, Julio, Julieta, Densisse, Daniel, Stephanie, Chuchin, Irma, Diego, Chava, Tía Gema, salvador, Omar, Israel, Ivan y Tete.

A mis amigos:
Jorge, Gabriel, Sonia, Simón, Rous, Feno, Obed, Beto, Lety, Alber, Charly, Xochitl, Pez, Chino, "Caguas", a toda la banda del C.C.H. Vallejo, de la Sec. 72. y de Prados.

Gracias a todos ellos por haberme ayudado en mi formación personal y profesional, por estar conmigo en las buenas y en las malas, por brindarme su apoyo y cariño cuando lo necesite y por compartir grandes momentos conmigo.

En especial, este trabajo esta dedicado a la memoria de mi padre.

JESÚS (CACHARPAS)

A mis padres:
Elodia Pedroza Salgado
José Fco. Rebollar Figueroa

A mi familia:
Lupis, Emilio, Pollo, Padris gracias por tu fuerza, Tía Queña, Tía Edith, Vick, Borrego, Tío Mundo, Alex, Tía Rosi, Tía Chayo, Familia Pedroza Salgado, Familia Rebollar Figueroa.

A mis seres amados que ya no están conmigo:
Doña Beta, Don Beto, Doña Victorina.

A mis amigos:
Moni, Chico, Chepe, Alaín, Ara, Carlos, Toño, Descriadillo, Magda, Rabiolas, AJEF mis brothers, Fer, y a toda la banda del C.C. Apache Vallejo.

Gracias a todos los guerreros que sufrieron y gozaron una parte de mi ida y que sin su ayuda no habría alcanzado mi mayor reto.

ÍNDICE

OBJETIVO GENERAL

Diseñar, desarrollar y proponer una implementación de un sistema de adquisición de datos a través del puerto serie para el monitoreo de la temperatura atmosférica en la estación meteorológica del prototipo aragonés "TIM"

OBJETIVOS ESPECÍFICOS

- Conocer la estructura y funcionamiento del puerto serie (RS-232).
- Aprender los fundamentos de la programación en Visual Basic 6.0.
- Diseñar un circuito de adquisición de datos a través del puerto serie, que sea compatible con computadoras personales comerciales.
- Conocer el funcionamiento de una estación meteorológica así como el funcionamiento de sensores meteorológicos.
- Realizar un programa para la recepción y transmisión de datos en Visual Basic 6.0
- Crear una interfaz grafica para la adquisición de datos, en Visual Basic 6.0.

JUSTIFICACIÓN

Una de las funciones del proyecto prototipo aragonés 'TIM' es realizar observaciones astronómicas de calidad. Por ello, es necesario contar con una estación meteorológica que realice mediciones atmosféricas.

El clima es un factor importante en la observación astronómica, un exceso de humedad podría ser dañino para el telescopio, se deben proteger los instrumentos que el telescopio mantiene en forma periférica, es necesario mantener una temperatura estable dentro de la cúpula, similar o igual a la exterior, por tal razón es imperioso la necesidad de contar con un sistema de adquisición de datos (software y hardware) para tener un monitoreo con una interfaz grafica, para que en lo sucesivo se puedan realizar predicciones meteorológicas y se puedan tomar medidas preventivas; no olvidemos que la humedad, la temperatura y un sistema de baja presión son factores que nos permiten determinar una posible lluvia.

Por otro lado, este sistema de adquisición de datos en sus componentes de hardware y software es muy económico, con respecto a los equipos robustos como son datalogger y los instrumentos de medición, nuestra propuesta es dotar al proyecto prototipo aragonés 'TIM' de un sistema económico y fácil de manipular, con la posibilidad de ampliar la captura de diferentes variables en sensores y equipos.

INTRODUCCIÓN

La necesidad de todo ingeniero es mejorar, diseñar e implementar todo tipo de circuitos para ofrecer a la sociedad herramientas que faciliten su desempeño profesional, laboral e intelectual. Esta tesis esta encaminada a solucionar un problema especifico que es el monitoreo de las condiciones atmosféricas y climáticas de nuestro entorno.

Nuestro trabajo de tesis es una extensión de un proyecto de investigación denominado "Prototipo Aragonés TIM", que le proporcionara la posibilidad de tener a la mano estadísticas meteorológicas del medio ambiente que coadyuvaran a la protección de los equipos de medición y observación.

En el capitulo 1 nuestra principal tarea fue empaparnos de las tecnologías básicas y modernas del puerto serial, que es el que mas se utiliza en el mercado para la adquisición de datos. Además fue indispensable tener a la mano los conceptos de tipos de transmisión, así como las diferentes normas que demarca el mercado estandarizado mundial.

En el capitulo 2 fue necesario planear una forma agradable de visualizar nuestros datos adquiridos, así como una programación ágil y precisa, por ello, la programación en Visual Basic fue nuestra mejor opción, ya que como estudiantes de la carrera de Ingeniería Mecánica – Eléctrica esta plataforma es muy digerible y de amplias posibilidades en sistemas operativos visuales,.

En el capitulo 3 esta plasmado el comportamiento profesional de una estación meteorológica, así como la utilización de sensores que ayudaran a los estudios meteorológicos de dicha estación, cabe mencionar que nuestro proyecto de tesis se ajusta a la economía de un estudiante pero con la posibilidad de una plataforma con recursos. Por otro lado, el diseño del circuito del proyecto de tesis, así como sus componentes, se explica detalladamente en este capitulo.

Por ultimo, en el capitulo 4 se crea una interfaz gráfica de programación en Visual Basic que interactúa con nuestro circuito proporcionándonos de manera visual la adquisición de los datos de nuestro sensor de temperatura.

CAPITULO 1. PUERTO SERIE E INTERFAZ

CAPITULO1. PUERTO SERIE E INTERFAZ

1.1 TIPOS DE COMUNICACIÓN

COMUNICACIÓN EN PARALELO. En la **comunicación en paralelo** los bits de información viajan simultáneamente, uno por cada hilo de conexión
COMUNICACIÓN SERIE. En la **comunicación serie** los bits se transmiten uno detrás de otro (de ahí el nombre), lo que hace que sean mucho más lentas que sus homólogas "paralelo" en las que se transmiten varios bits a la vez. La ventaja es que puede utilizarse un solo par de hilos (o incluso uno solo si el retorno se realiza por la tierra).

Atendiendo a como viajan los datos en un medio de comunicación, tenemos tres tipos de transmisión:

- **Simplex**: Un equipo transmite, el otro recibe.
- **Half duplex:** transmiten ambos equipos pero no simultáneamente; los equipos se alternan en la transmisión, uno transmite mientras el otro recibe.
- **Full duplex:** Ambos equipos transmiten simultáneamente. Para ello se requieren dos líneas independientes, una de transmisión y otra de recepción; la línea de transmisión de un equipo se conecta a la entrada de recepción del otro y viceversa. Los puertos serie del PC son capaces de utilizar este modo.

1.2TIPOS DE SINCRONIZACION DE LA COMUNICACIÓN

Transmisión asíncrona

En este modo de transmisión no existe sincronización; no es necesario enviar caracteres de relleno, pero hay que indicar cuando empieza un dato y cuando termina. Esto se hace incluyendo en la transmisión señales de inicio y fin de dato (bits de "start" y "stop"). En la comunicación asíncrona, la información (cada carácter) es enviada en el interior de un cuadro ("Frame") de tamaño variable, que comienza con la mencionada señal de inicio y termina con la de final; es el tipo de comunicación utilizada en los puertos serie del PC. En este tipo de comunicación, el estado de reposo (cuando no se transmite nada) se identifica con un "1" (marca). Cuando se recibe un bit de inicio, que es un "0" (espacio), el receptor toma nota que va a comenzar a recibir un dato.

La transmisión asíncrona define el principio y el final de cada carácter u octeto de 8 bits enviado por las líneas. La palabra asíncrono puede prestarse a confusiones, ya que implica una no sincronización. En realidad, se inserta un bit de principio y otro de final (o de "inicio" y "paro") entre cada palabra de 8 bits para sincronizar el transmisor y el receptor. Se incluye un bit de paridad para detectar errores.

La transmisión asíncrona no requiere señales de temporización individual para cada carácter; sólo son necesarias señales de temporización para largos párrafos o bloques de datos. Por tanto, entre los caracteres no hay bits de "inicio" y "paro".

La transmisión de datos binarios puede expresarse como una de las dos condiciones, marca para un "1" binario y espacio para un "0" binario. En la transmisión asíncrona el transmisor sube a una condición de marca al final de cada octeto y permanece en este nivel hasta que se anuncia el siguiente octeto mediante un espacio. Por tanto, la marca al final del octeto es el bit de paro y el espacio al principio del octeto es el bit de inicio. Estos dos bits de sincronización permiten que el receptor del final de la línea quede fijado o sincronizado con el transmisor. Sin embargo, un octeto de 8 bits necesita dos bits adicionales para señalar cuándo llega un octeto y cuándo está completado: estos bits no transportan datos y, por tanto, el sistema es relativamente ineficaz. Las señales de reloj o de temporización del transmisor y del receptor quedan sincronizadas o fijadas cada vez que llega un octeto.

Pueden existir largos períodos (siempre en el rápido mundo del nanosegundo de los ordenadores) en que no se transmiten octetos. Sin embargo, en cuanto aparece un octeto, se produce de nuevo la sincronización.

Transmisión síncrona

En este modo, los dispositivos que se comunican se sincronizan en el momento inicial de la transmisión y constantemente se intercambian información a una cadencia predefinida. Con objeto de mantener la sincronización, cuando no existen datos que enviar se transmite caracteres sin valor ("idle characters").Esta transmisión es más rápida que la asíncrona porque no es necesario transmitir señales de inicio o fin de dato; constantemente se reciben caracteres que pueden ser de datos o sin valor (de relleno).

La transmisión síncrona se emplea para datos de alta velocidad, con la cual, el reloj transmisor dispara el reloj del receptor y permite que corra durante una larga secuencia de bits o bloques de datos. Los octetos se transmiten en una corriente rápida y fija; en el caso de que se produzcan vacíos en el flujo de datos, el transmisor debe inyectar octetos vacíos para mantener el sincronismo. El sistema de transmisión síncrona se inicia mediante un formato de bits o código predeterminado que envía el transmisor.

1.3NORMALIZACION DE LA INTERFAZ

Desde el punto de vista técnico, se le llama interfaz al dispositivo que permite la conexión entre dos equipos para que exista comunicación entre ellos. En general, una interfaz puede ser un elemento software o hardware, ya que también puede recibir el nombre de interfaz aquel software que permite comunicar por ejemplo, dos aplicaciones distintas. No obstante, en nuestro caso llamamos interfaz al conector y cable que conecta al ordenador (Equipo Terminal de Datos) con el modem (Equipo Terminal de Circuito de Datos) u otro dispositivo ETCD.

Desde 1969, se han publicado distintas normas de interfaz, cada una con la finalidad de evolucionar para permitir mayor velocidad en la transmisión o mayor funcionalidad; sin embargo, la historia le ha dado el papel protagónico a la norma RS232C, la cual, aun siendo de las más antiguas, ha sido y es la más utilizada.

RS232C y RS232D de EIA

La norma RS-232C fue definida por la Asociación de Industrias de Electrónica, EIA (Electronic Industrie Association), en 1969. Esta norma es idéntica a la norma V24 definida por el UIT-T (Unión Internacional de Telecomunicaciones, antes CCITT, Comité Consultivo Internacional Telegráfico y Telefónico), siendo la principal diferencia que en el caso de V24, las características eléctricas de las señales se especifican separadamente en la recomendación V28. Cabe destacar que aunque la norma RS-232C se hizo pensando en una interfaz que conectase terminales y modems, hoy en día se utiliza para unir dos dispositivos cualesquiera que requieran ser conectados.

A finales de los sesenta, la EIA remplazó la recomendación RS-232C por las recomendaciones RS-449, RS-422 y RS-423. Estas recomendaciones fueron diseñadas para permitir velocidades de transmisión de datos mayores, así como para tener mayor funcionalidad. A pesar de estas ventajas y de la promoción que tuvieron estas normas por parte de determinados organismos gubernamentales norteamericanos, los fabricantes prefirieron seguir con el estándar RS-232C, entre otras razones, por el hecho de estar ampliamente difundido. Este hecho obligó a que la EIA publicase en enero de 1987 la revisión D de la recomendación RS-232 (RS232-D), así como un nuevo estándar conocido como RS-530. En la tabla 1.1 se muestra una comparación entre las diferentes normas.

COMPARACIÓN ENTRE LAS SEÑALES DE LAS DISTINTAS NORMAS					
V24		RS-232		RS-449	
102 102a 102b	Tierra de señales ETD común ETCD común	AB	Tierra de señal	SG SC RC	Tierra de señalización Emisión común Recepción común
125 108/2 107	Indicador señal de llamada Terminal de datos preparado Conjunto de datos preparado	CE CD CC	Indicador señal de llamada Terminal datos preparado Conjunto datos preparado	IS IC TR DM	Terminal en servicio Llamada entrante Terminal preparado Modo de datos
103 104	Transmisión de datos Recepción de datos	BA BB	Transmisión da datos Recepción de datos	SO RD	Emisión de datos Recepción de datos
113 114 115	Sincronismo transm. ETD Sincronismo transm. ETCD Sincronismo recepción	DA DB DO	Sincronismo transm. ETD Sincronismo transm. ETCD Sincronismo recepción	TI ST RT	Sincronización de terminal Sincronización de emisión Sincronización de recepción
105 106 109 110 126 111 112	Petición de emisión Preparado para transmitir Detector de señal de línea recibida Detector de calidad de señal Selecc. de frecuencia de transmisión Selección de velocidad. ETD Selección de velocidad. ETCD	CA CB CF CG CH CI	Petición de emisión Preparado para transmitir Detector señal de línea recibida Detector de calidad de señal Selección de velocidad. ETD Selección de velocidad. ETCD	RS CS RR SQ NS SF SR SI	Petición de emisión Preparado para transmitir Receptor preparado Calidad de la señal Señal nueva Frecuencia seleccionada Selector de velocidad binaria Indicador de velocidad binaria
118 119	Transmisión de datos por el canal de vuelta Recepción de datos por el canal de vuelta	SBA SBB	Transmisión de datos secundaria Recepción de datos secundaria	SSD SRD	Emisión de datos secundaria Recepción de datos secundaria
120 121 122	Transmisión de señal de línea por el canal de vuelta Canal d. vuelta preparado Detector de señal de línea recibida por el canal de vuelta	SCA SCB SCF	Petición de emisión secundaria Preparado para transmitir secundaria Detector de señal de línea recibida secundaria	SRS SCS SRR	Petición de emisión secundaria Recepción de datos secundario Receptor preparado secundario
141 140 142	Bucle local Bucle remoto Indicador de prueba			LL RL TM	Bucle local Bucle remoto Modo de prueba
116 117	Espera selectiva Indicador de espera			SS SB	Espera selectiva Indicador de espera

Tabla 1.1 Comparación entre las señales de las distintas normas

Las diferencias entre las recomendaciones RS-232C y RS-232D van dirigidas fundamentalmente a la definición de determinados contactos y a determinadas operaciones de prueba, teniendo ambas 25 contactos y siendo su uso prácticamente idéntico. (Ver Fig. 1.1). En adelante, para simplificar, cuando se hable de RS-232 se va hacer referencia tanto a RS-232C como a RS232-D.

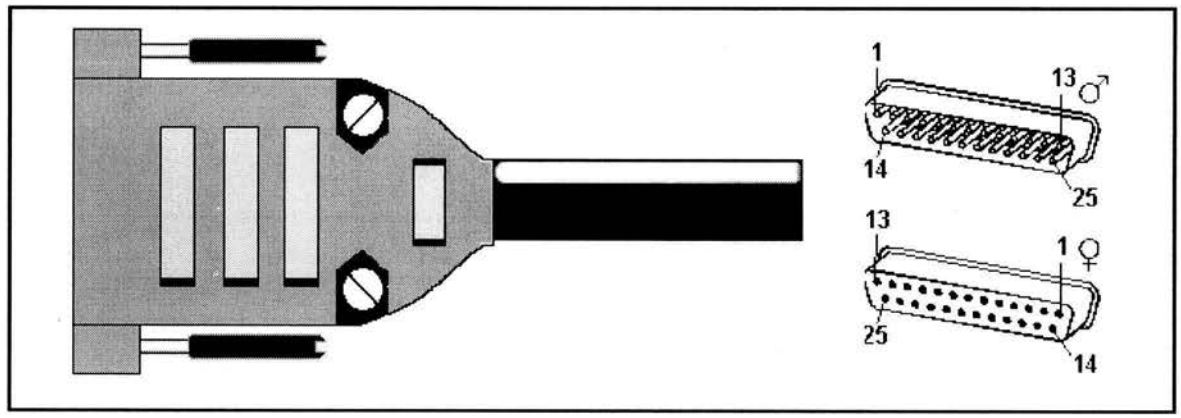


Figura 1.1 Interfaz RS-232

Estandar V24 y V28 de la UIT

La UIT (Unión Internacional de Telecomunicaciones, antes el CCITT) lleva a cabo la normalización de la interfaz separando claramente los tres niveles fundamentales: nivel mecánico, nivel eléctrico y nivel lógico. El nivel mecánico se refiere al dimensionado físico del conector, así como a definir dónde están situados el conectar macho y el conectar hembra. El nivel eléctrico define las características puramente eléctricas de los circuitos que forman el enlace modem - terminal. Estas características están definidas en las siguientes normas:

V28 Define las características eléctricas de los circuitos de enlace asimétricos (no equilibrado) para uso con equipos que funcionen a velocidades menores de 20 Kbps.

V10 Define las características eléctricas de los circuitos de enlace asimétricos para uso con equipos que funcionen a velocidades de 20 a 100 Kbps.

V11 Define las características eléctricas de los circuitos de enlace simétricos (equilibrados) para su uso con equipos que funcionen a velocidades de hasta 10 Mbps.

En cuanto a la recomendación V28, utilizada por la mayoría de los modems actuales, define las siguientes características eléctricas:

- La impedancia de entrada al receptor debe estar entre 3000 y 7000 ohmios.
- La capacidad máxima de la carga debe ser menor de 2500 picofaradios. La tensión del generador en circuito abierto debe ser igual o menor de 25 voltios.
- Corriente de cortocircuito de 0,5 amperios.
- El generador de un circuito de enlace debe soportar el circuito abierto y el cortocircuito entre él y cualquier otro circuito del enlace sin que ni el generador ni el equipo asociado sufran daños importantes.
- La capacidad efectiva de los conductores del cable debe ser del orden de 160 picofaradios por metro.

Por último, el nivel lógico o funcional está recogido en la norma V24 y define la utilidad de cada uno de los circuitos que componen el enlace. Cada uno de estos circuitos se designa con un número de tres cifras, existiendo dos categorías:

- Serie 100. Son los circuitos cuya numeración comienza por el número 1. Estos abarcan los circuitos de utilización general, existiendo un total de 39.
- Serie 200. Son los circuitos cuya numeración comienza por el número 2. Estos circuitos están utilizados exclusivamente en las llamadas automáticas vía red telefónica, existiendo un total de 13.

En la práctica, sólo se utiliza una parte de estos circuitos, siendo el nivel mecánico independiente para cada serie. Los circuitos utilizados en las conexiones módem - terminal son los definidos en la tabla 1.2.

CIRCUITOS HABITUALES DE LA RECOMENDACIÓN V24				
CIRCUITO V24	RS-232C EQUIV	CONTACTO	ORIGEN DE LA SEÑAL	DENOMINACION
102	AB	7	-	Tierra de señalización o retorno común
103	BA	2	ETD	Transmisión de datos
104	BB	3	ETCD	Recepción de datos
105	CA	4	ETD	Petición de emisión
106	CB	5	ETCD	Preparado para transmitir
107	CC	6	ETCD	Modem preparado
108/1	-	20	ETD	Conéctese el modem a la línea
108/2	CD	20	ETD	Terminal de datos preparado
109	CF	8	ETCD	Detector de portadora en línea
110	CG	21	ETCD	Detector de calidad de señales de línea
111	CH	23	ETD	Selector de velocidad binaria
113	DA	24	ETD	Sincronismo en transmisión por ETD
114	DS	15	ETCD	Sincronismo en transmisión por ETCD
115	DD	17	ETCD	Sincronismo en recepción
125	CE	22	ETCD	Detector de señal de llamada
126	CY	11	ETD	Selector de canal de transmisión

Tabla 1.2 Circuitos habituales de la recomendación V24

X21 de la UIT

Las comunicaciones del PC suelen llevarse a cabo en modo asíncrono y la interfaz utilizada en la mayoría de estos casos es la RS-232C o su equivalente V24 y V28 de la UIT-T (CCITT), no obstante, también existen comunicaciones en las que se utilizan tarjetas adaptadoras donde la interfaz no es la estándar RS-232C, sino la que le corresponda a la aplicación particular que se vaya a llevar a cabo. Una de estas interfaces es la X21, que es utilizado, por ejemplo, por los ordenadores que se conectan a la red de conmutación de paquetes X25 en modo dedicado.

La recomendación X21 de la UIT-T (CCITT), presentada en 1976, incluye un protocolo para enviar y recibir llamadas, y para enviar y recibir datos mediante transmisión síncrona dúplex. Esta interfaz incorpora señales de sincronismo de bit y de sincronismo de carácter (byte), aunque estas últimas señales son opcionales. En contraposición con las especificaciones RS, X21 utiliza únicamente 6 señales. Las especificaciones eléctricas están recogidas en las recomendaciones X26 (equivalente a la RS-422A) y X27 (equivalente a la RS-423A). La menor velocidad de línea de X21 es de 64 Kbps. La tabla 1.3 nos describe las señales X21.

En esta interfaz, el ordenador envía los datos al módem por la línea de Transmisión, y el módem envía sus datos al ordenador por la línea de Recepción. Estos dos circuitos se utilizan también para enviar la mayor parte de las señales de control. Los circuitos llamados de Control y de Indicador se utilizan para controlar las llamadas en curso. El ordenador cambia el estado del circuito de control cuando quiere hacer una llamada, devolviendo dicha señal al estado de reposo cuando desea terminar la llamada. Igualmente, el módem cambia el estado del circuito de Indicador cuando la llamada que ha generado es contestada en el otro extremo de la línea.

La mayor diferencia entre X21 y RS-232 Y RS-449, es que las señales de X21 van codificadas, realizándose la mayor parte del control de la comunicación por los mismos canales de transmisión y recepción X21 se define como el nivel físico del protocolo de conmutación de paquetes X25.

SEÑALES DE X21		
CIRCUITO	NOMBRE	ORIGEN
G	Tierra común o retorno	
Ga	común	ETD
T	Retorno común del ETD	ETD
R	Transmisión de datos	ETCD
C	Recepción de datos	ETD
I	Control	ETCD
S	Indicador	ETCD
B	Sincronización de bit Sincronización de byte	ETCD

Tabla 1.3 Señales de X21

1.4 CARACTERÍSTICAS DE LAS SEÑALES RS-232

El estándar RS-232 se aplica a las comunicaciones serie entre el ETD (ordenador) y el. ETCD (módem) para comunicaciones a velocidades iguales o menores a 20 Kbps, con una longitud de cable igual o menor a 15 metros. La longitud del cable puede ser mayor si se transmite a velocidades menores, o bien se puede transmitir a velocidades más elevadas si se utiliza una longitud de cable menor.

Aunque la interfaz RS-232 define un cable con 25 conductores, para conectar un PC a un módem se requieren normalmente un número menor. Las comunicaciones asíncronas requieren como máximo 9 o 12 conductores, y las comunicaciones síncronas requieren como máximo 12 o 16 conductores. Esta diferencia del número de conductores necesarios se debe a las distintas características de operación del módem.

La recomendación RS-232C establece que la señal de cualquier contacto esté en el estado llamado activo (ON) cuando su tensión eléctrica se encuentre entre los +3 y los + 15 voltios. De la misma forma, establece que la señal esté en el estado no activo (OFF) cuando su tensión eléctrica se encuentre entre los -3 y los -15 voltios. El rango de tensión entre los +3 y los -3 voltios se considera un estado de transición que no tiene efecto sobre las condiciones del circuito. Por su parte, la recomendación RS-232D establece que los estados activo y no activo de las señales se den cuando los niveles de tensión se encuentren entre los +3 y +25 voltios y los -3 y -25 voltios, respectivamente. (Ver Fig. 1.2)

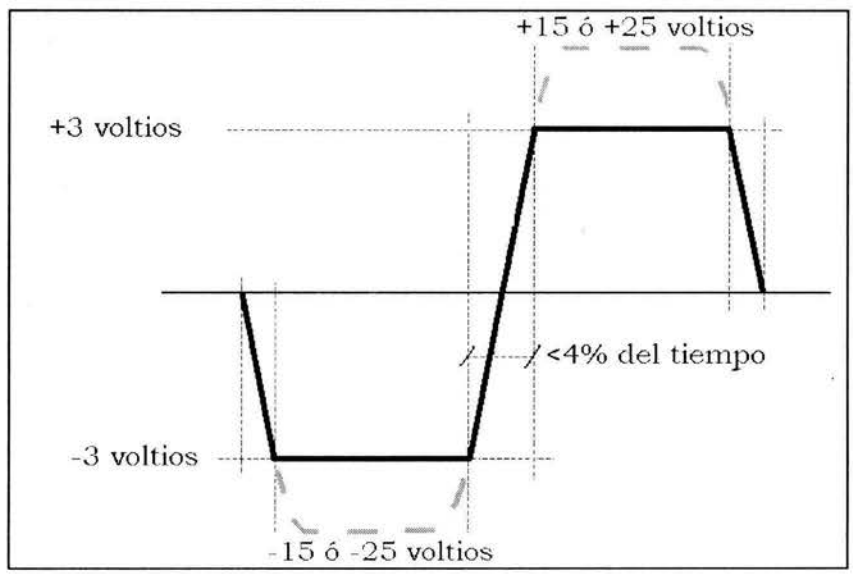


Figura 1.2. Características de las señales de la interfaz RS-232

Desde el punto de vista de los fabricantes, para hacer compatibles sus equipos con ambas normas utilizan tensiones de +5 y -5 voltios o bien de +12 y -12 voltios para definir los distintos estados de la señal. A este respecto, cabe comentar que en aquellos casos en los que interese utilizar cables de interfaz largos, conviene disponer de modems (o cualquier otro dispositivo) que utilicen niveles de tensión de +12 y -12 voltios.

RS-232 define que el estado activo se corresponde con el estado binario 0, y el estado no activo se corresponde con el estado binario 1. A estos dos estados también se les conoce por las denominaciones ESPACIO (0) y MARCA (1). En la tabla 1.4 se resumen las características de las señales de la interfaz RS-232.

Una vez vistas las distintas tensiones utilizadas por la interfaz, podemos explicarnos el porqué de la limitación de 15 metros de longitud del cable. Hay que tener en cuenta que la forma de onda utilizada por las señales digitales que se transmiten por la interfaz están definidas de forma que sólo un 4 por ciento del tiempo de cada periodo de bit es el que se debe utilizar para realizar una transición (paso de -3 a +3 voltios, o viceversa). Dado que la característica eléctrica de capacidad limita la velocidad de cambio de estados de tensión o, dicho de otra forma, limita el tiempo de transición de la señal (la capacidad se porta corno un freno a los cambios de tensión), para la velocidad máxima a la que está definida la interfaz RS-232 (20 Kbps) la capacidad del cable no debe superar los 2500 picofaradios (pF). Como los cables que se suelen utilizar tienen una capacidad de 130 a 170 picofaradios por metro, RS-232 limita los cables a 15 metros. No obstante, a menor velocidad, mayor longitud de pulso, lo que supone que el 4% de esta longitud es mayor, y por tanto el cable puede ser más largo.

SEÑALES DE LA INTERFAZ RS-232		
	NÉGATIVO	POSITIVO
Tensión RS-232C (voltios)	-3 a -15	+3 a +15
Tensión RS-232D (voltios)	-3 a -25	+3 a +25
Estado binario	1	0
Condición de señal	Marca	Espacio
Función	No activa (Off)	Activa (On)

Tabla 1.4. Señales de la interfaz RS-232

RS-232 identifica cada circuito de la interfaz mediante una designación de 2 o 3 letras. La primera de estas letras agrupa a cada circuito, cada contacto, en una de las ocho categorías posibles. (Ver tabla 1.5 conexiones RS-232).

Lo cierto es que, independientemente de las denominaciones dadas por la normas, en la práctica se identifica cada circuito bien por el número de la patilla del conector o bien por un mnemónico de 2 o 3 letras que define la función de cada uno. Esta última es la denominación más fácil de memorizar.

CONEXIONES RS-232				
NUMERO CONTACTO	IDENTIFICACION RS-232	MNEMONICO	NOMBRE COMPLETO	ORIGEN DE LA SEÑAL
DATOS				
2	BA	TD	Transmisión de datos	ETD
3	BB	RD	Recepción de datos	ETCD
CONTROL DE FLUJO				
6	CC	DSR	Módem preparado	ETCD
20	CD	DTR	Terminal de datos preparado	ETD
4	CA	RTS	Petición de envío	ETD
5	CB	CTS	Preparado para transmitir	ETCD
LINEAS DE MODEM				
8	CF	CD	Detección de portadora	ETCD
22	CE	RI	Indicador de llamada	ETCD
TIERRA COMUN				
7	AB	SG	Tierra de señal	
CONEXIONES MENOS USADAS				
1	AA	GND	Tierra de protección	ETCD
12	SCF		Detección de portadora secundario	ETD
13	SCB		Preparado para transmitir secundario	ETD
14	SBA		Transmisión de datos secundario	ETCD
15	DB		Sincronismo en transmisión por	ETCD
16	SBB		ETCD	ETCD
17	DD		Recepción de datos secundario	ETD
19	SCA		Sincronismo en recepción	ETCD
21	CG		Petición de envío secundario	ETD
23	CH		Detector de calidad de la señal de	ETD
23	CI		línea	ETCD
24	DA		Selector de velocidad binaria	ETD
			Selector de velocidad binaria	
			Sincronismo en transmisión por ETD	
PRUEBAS				
9	-		Reservado para pruebas (+Vcc)	
10	-		Reservado para pruebas (-Vcc)	
18	(LL)		Bucle local	ETD
25	(TM)		Modo prueba	ETCD

Tabla 1.5. Conexiones RS-232

1.5 CIRCUITOS PRINCIPALES DE RS-232

Las señales que se intercambian entre el terminal y el módem en el proceso de una comunicación son las siguientes:

GND **Contacto l. Tierra de protección (Protective Ground).** Este contacto está generalmente conectado al mismo chasis del equipo, e incluso puede estar conectado a una señal de tierra externa. Esta señal también se puede utilizar para apantallar un cable protegido, de forma que se minimicen las interferencias producidas en entornos con alto nivel de ruido. Hay que aclarar que la referencia común para todas las señales no es este contacto, sino el contacto 7.

SG **Contacto 7. Tierra de señal (Signal Ground).** Este contacto es la referencia de todo el resto de las señales de la interfaz, incluidas las señales de datos, señal de reloj y señales de control. La tensión de esta señal siempre debe ser 0 voltios. En teoría, los contactos 1 y 7 deben ser independientes pero en la práctica frecuentemente están unidos formando una señal de tierra común.

TD **Contacto2. Transmisión de datos (Transmited Data).** Este circuito es el utilizado para transmitir las señales de datos desde el equipo terminal (ETD) al modem (ETCD). Cuando no se transmite ningún dato, este contacto debe mantener la señal lógica 1. Para que el terminal pueda transmitir datos por el contacto 2, los circuitos RTS, CTS, DSR y DTR deben tener antes una tensión alta. Este contacto también se conoce con el mnemónico TXD.

RTS **Contacto 4. Petición de envío (Request to Send).** La señal de este circuito es enviada desde el terminal (ETD) al modem (ETCD) para preparar el módem para la transmisión. Una vez hecho esto, y antes de empezar a transmitir datos, el terminal debe recibir la señal CTS por el contacto 5. Ambas señales, RTS/CTS, también pueden ser utilizadas para controlar el flujo de datos entre el módem y el terminal. Para que estas señales puedan ser reconocidas como indicadores del flujo de datos, tanto el módem como el software de comunicaciones deben ser configurados para mantener un control de flujo RTS/CTS, también llamado control de flujo hardware. Cuando un módem opera de forma asíncrona, el software de comunicaciones suele mantener la señal RTS constantemente en alto, indicando que el módem puede enviar datos al terminal en cualquier momento.

CTS **Contacto 5. Preparado para transmitir (Clear to Send).** Este circuito se utiliza para indicarle al terminal que el módem está listo para transmitir. El módem activará esta señal después de que el terminal active su señal RTS. Este circuito también puede ser utilizado junto con RTS como control de flujo de datos entre el terminal y el módem. Al igual que con la señal RTS, para que CTS pueda ser reconocida como indicador del flujo de datos, tanto el módem como el software de comunicaciones deben ser configurados para mantener un control de flujo RTS/CTS.

CD **Contacto 8. Detección de portadora (Carrier Detect).** A este circuito también se le conoce por el nombre de Detector de la Señal de Línea Recibida, RLSD (Received Line Signal Detector), o como Detección de Portadora de Datos, DCD (Data Carrier Detect). Una señal en este circuito le indica al terminal que el módem está recibiendo una señal de portadora del módem remoto. La señal de portadora tiene que estar presente durante todo el tiempo que dure la comunicación. se transmitan datos o no. Por tanto, si el terminal no detecta la señal CD, dará por terminada la comunicación por pérdida de portadora. En este caso el software de comunicaciones dará un mensaje similar a Pérdida de portadora (Carrier lost) para indicar esta condición. En el caso de que el módem disponga de indicadores luminosos, la presencia de esta señal también ilumina el indicador CD del módem. Como veremos en capítulos posteriores, la portadora no es más que un tono a una frecuencia determinada, la cual sirve de referencia, pero en si mismo no transporta ninguna información de usuario.

RD **Contacto 3. Recepción de datos (Receive Data).** Los datos que va demodulando el módem los envía al terminal por este contacto. Si el módem no tiene ningún dato que enviar al terminal, debe mantener este circuito en estado no activo (OFF, estado binario 1). A este contacto también se le conoce por el mnemónico RXD.

DSR **Contacto 6. Modem preparado (Data Set Ready).** La señal de este circuito indica el estado del módem. Cuando este circuito está activo, (valor lógico 0), indica que el módem está conectado a la línea telefónica y está listo para transmitir datos. Este contacto también puede ser utilizado por el módem para indicar que ha terminado un proceso de auto revisión o que la marcación del número telefónico ha sido efectuada con éxito. La norma RS-232D ha cambiado el nombre de esta señal por el de ETCD preparado (DCE ready).

DTR **Contacto 20. Terminal de datos preparado (Data Terminal Ready).** Cuando esta señal está activa, le indica al módem que el terminal está encendido y listo para una comunicación. Si la señal no está activa, el módem cortará cualquier comunicación que esté en curso. Este circuito controla, por tanto, la conexión del módem a la línea telefónica. La norma RS-232D ha cambiado el nombre de esta señal por el de ETD preparado *(DTE ready)*.

RI **Contacto 22. Indicador de llamada (Ring Indicator).** Este circuito le indica al terminal que está siendo recibida una señal de llamada por el canal de comunicaciones. Este circuito es utilizado por aquellos modems que están en modo respuesta automática, para indicarle al terminal que se está recibiendo una llamada. En respuesta a esta señal de llamada, el terminal le pasa una tensión al contacto 20 (circuito DTR). Esta tensión le dice al módem que descuelgue y atienda la llamada.

CG **Contacto 21. Detector de calidad (Quality Detector).** Las señales de este circuito son transmitidas desde el módem al terminal siempre que el módem detecta una alta probabilidad de error en la recepción de los datos debido a una mala calidad de la línea. Este circuito permanecerá en estado activo cuando la calidad de la señal es aceptable cambiando al estado no activo si la calidad es inadecuada.

CH/CI **Contacto 23. Selector de velocidad (Data Signal Rate Selector).** Cuando el módem detecta una mala calidad de la línea y desactiva la señal CG, si este estado es mantenido durante un tiempo predeterminado, el terminal puede indicarle al módem que cambie su velocidad de operación por una más baja. Para hacer este cambio de velocidad se utiliza el contacto 23. El terminal pone el contacto 23 en estado activo para una velocidad de operación más elevada, y lo pone en estado no activo para una velocidad de operación más baja. Esta decisión de cambio de velocidad también puede ser tomada por el módem. Cuando es el terminal quien selecciona la velocidad de operación, la señal del contacto 23 va del terminal al módem, y el circuito es conocido como circuito CH. Si es el módem quien determina la velocidad de operación, la señal del contacto 23 va del módem al terminal, y el circuito es conocido como circuito CI.

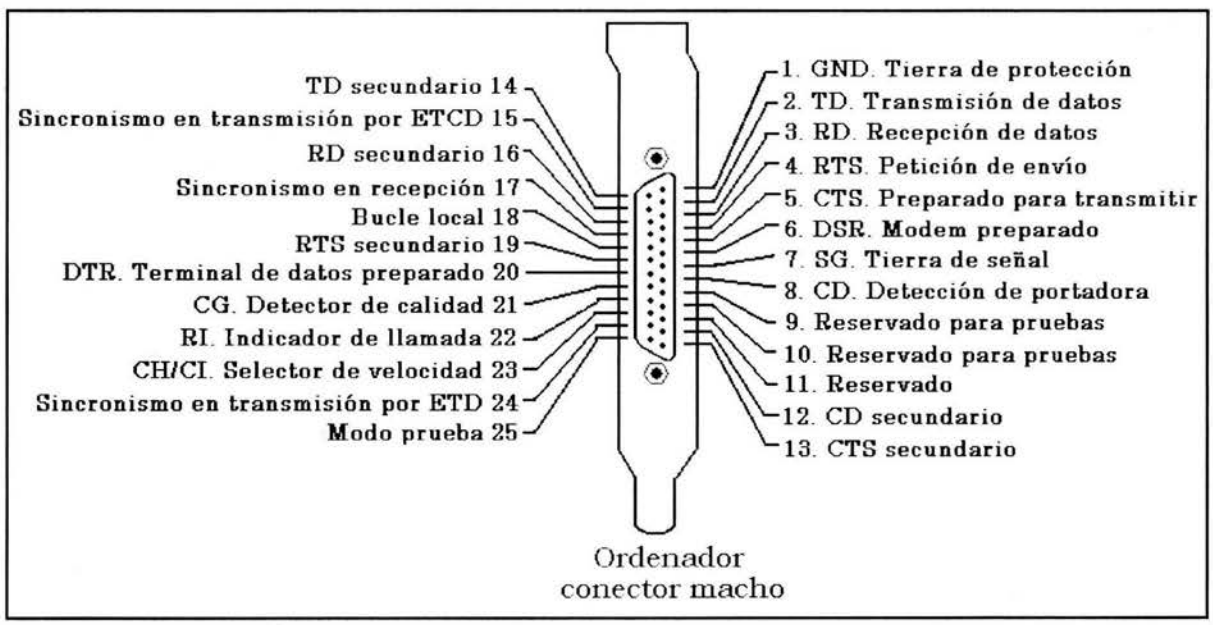


Figura 1.3 Circuitos de la interfaz RS-232

Proceso de comunicación

En el flujo de datos entre el ordenador (ETD) y el módem (ETCD) existen tres circuitos principales. El circuito 2, que es por donde circulan los datos del ordenador al módem (transmisión); el circuito 3, que es por donde circulan los datos del módem al ordenador (recepción), y el circuito 7, que es la señal de tierra a la que están referidas las tensiones de los circuitos anteriores.

Para que se produzca un intercambio de datos entre el ordenador (ETD) y el módem (ETCD), antes cada uno de ellos tiene que saber que el otro está conectado y listo para recibir los datos que pretende transmitir. Esto quiere decir que el ordenador (ETD) no iniciará ninguna acción si antes no comprueba que el circuito 6, DSR (Data Set Ready, módem listo), está en estado activo (valor binario 0). Este circuito le indica al ordenador (ETD) que el módem (ETCD) está conectado a la línea telefónica y está listo para transmitir datos. De la misma forma, el terminal le indica al módem que está preparado activando el circuito 20, señal DTR (Data Terminal Ready, terminal de datos activo). Hay que decir que algunos terminales (software de comunicaciones) mantiene siempre activa la señal DTR, pero otros sólo la activan cuando reciben la señal de llamada por el contacto 22.

Una vez que el ordenador (ETD) ha comprobado que el módem (ETCD) está activo, pone el contacto 4, RTS (Request to Send, petición de e envío), en estado activo para indicarle que a continuación le va a transmitir datos. A esta petición, el módem (ETCD) responde activando el circuito 5, CTS (Clear to Send, listo para enviar), indicándole al ordenador (ETD) que está preparado para modular cualquier bit que reciba por el circuito 2. Después de esto, el ordenador (ETD) transmitirá sus datos por el circuito 2.

Cuando se recibe una llamada, el módem responde pasando la señal de tensión de llamada al contacto 22, RI (Ring Indicator, indicador de llamada). Si el terminal es de los que no tienen siempre activo el contacto 20, lo activará después de recibir la señal RI del módem. A continuación, el módem descuelga y el módem distante transmite un tono de portadora. El tono de portadora debe ser mantenido durante toda la comunicación, ya que será la señal que indique, por un lado, la continuidad de la conexión establecida entre los dos circuitos, y por otro, el hecho de que en el otro extremo sigue habiendo un equipo activo. Cuando el módem detecta la portadora, activa la señal CD (circuito 8), y no la desactiva hasta que la señal portadora no desaparezca al final de la comunicación. Una vez realizado este preámbulo, se lleva a cabo el intercambio de datos.

En el proceso de la comunicación, los contactos 4 (RTS) y 5 (CTS) son también utilizados para controlar el flujo de datos entre el terminal (ETD) y el módem (ETCD). Tanto si el terminal (ETD) o el módem (ETCD) se ven saturados, cada uno puede hacer que el otro interrumpa temporalmente la transmisión desactivando la señal del circuito 4 o 5, respectivamente. Al activar de nuevo cualquiera de estos circuitos, se reanudará la transmisión.

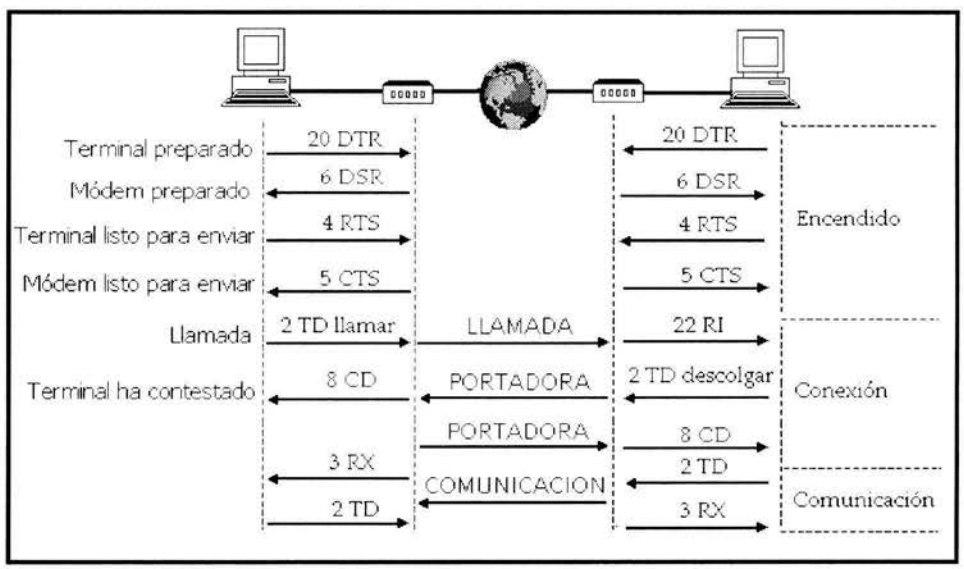


Figura 1.4 Proceso de comunicación

Todo el proceso descrito anteriormente está de acuerdo con la norma RS-232, la cual establece un conector de 25 contactos en cada extremo, estando conectado cada contacto de un extremo con su idéntico en el otro extremo. No obstante, algunos modems utilizan un cable más simple, donde, por ejemplo, no existe la línea 5, engañando al ordenador mediante un puente entre los contactos 5 y 8 del conector del PC. De la misma forma, también se ahorran los hilos 6 y 20, haciendo un puente entre ellos en ambos extremos. Con esta configuración no se controla el flujo, pero funciona.

1.6 OPERACIONES SINCRONAS

Para llevar a cabo comunicaciones en modo síncrono, se necesita un terminal o un PC con un adaptador de comunicaciones síncronas. Ejemplos de adaptadores de comunicaciones síncronas son el adaptador de comunicaciones bisíncronas de IBM y el adaptador de comunicaciones SDLC (Synchronous Data Link Control, control de enlace de datos síncronos), también de IBM. Existen también modems que disponen de la capacidad de auto sincronizarse (Auto-sync). Estos modems aceptan datos del terminal en modo asíncrono, convirtiendo y transmitiendo dichos datos a la línea en modo síncrono. En estos casos, el terminal no necesita ningún adaptador especial.

Cuando se utiliza un módem síncrono, éste coloca la señal de reloj en el contacto 15, de modo que le indica al terminal la cadencia exacta con la que tiene que transmitir las señales de datos. El módem puede generar las señales de reloj mediante un reloj interno o bien puede deducirlas de la propia señal de datos que recibe del módem distante. El procedimiento para hacer esto no está normalizado, por lo que puede haber diferencias entre unos fabricantes y otros. Por ejemplo, mientras unos fabricantes pueden utilizar las transiciones desde el nivel bajo al alto para deducir las señales de reloj, otros pueden utilizar las transiciones desde el nivel alto al bajo. Este hecho hace que algunos módems síncronos de fabricantes distintos sean incompatibles entre sí para trabajar en comunicaciones con circuitos punto a punto (líneas alquiladas).

El contacto 15 recibe el nombre informal de reloj de transmisión, siendo su nombre formal el de temporización de señal de transmisión ETCD (Transmission Signal Element Timing). ETCD hace referencia a que es el módem el que suministra la señal de reloj. Esta señal de reloj sirve para sincronizar los datos que transmite el terminal por el contacto 2.

Los datos que son recibidos por el módem desde el extremo distante son demodulados y enviados al terminal por el contacto 3 junto con la señal de reloj de recepción, que es enviada por el contacto 17. Esta señal de reloj se conoce formalmente como Temporización del elemento de señal recibida (Receiver Signal Element Timing).

En algunos casos, la señal de reloj utilizada por el módem puede ser facilitada por el terminal. En estas situaciones, el terminal facilita la señal de reloj al módem por el contacto 24, recibiendo esta señal el nombre de Temporización de señal de transmisión ETD.

En ciertos casos, un módem síncrono puede mantener una segunda comunicación por un canal secundario simultáneamente a la comunicación mantenida por el canal primario. En estos casos la velocidad de los datos del canal secundario es normalmente una fracción de la velocidad de los datos del canal primario. Para llevar a cabo la comunicación por el canal secundario, se utilizan cinco circuitos:

- Circuito 14: Transmisión de datos. Equivalente al contacto 2.
- Circuito 16: Recepción de datos. Equivalente al circuito 3.
- Circuito 19: Petición de envío. Equivalente al circuito 4.
- Circuito 13: Preparado para enviar. Equivalente al circuito 5.
- Circuito 12: Detección de portadora. Equivalente al circuito 8.

1.7 TIPOS DE PUERTOS SERIE EN UN PC

Hasta ahora se ha descrito la comunicación entre un ordenador y un módem teniendo en cuenta el estándar RS-232, el cual considera la existencia de un conector de 25 contactos; no obstante, como posiblemente ya habrá observado, los ordenadores de tipo PC disponen de dos clases de conectores de puertos serie:

- Conector tipo D de 25 contactos (conector 25D).
- Conector tipo D de 9 contactos (conector 9D).

Desde el punto de vista de las comunicaciones asíncronas, ambos conectores son idénticos y realizan las mismas funciones, ya que los 9 contactos del conector pequeño son precisamente los 9 contactos usados en cualquier comunicación asíncrona. La tabla 1.6 indica las equivalencias entre conectores de 9 y 25 contactos y la fig.1.5 nos muestra los diferentes tipos de conectores y cables para el puerto serial.

EQUIVALENCIAS DE CONECTORES DE 9 Y 25 CONTACTOS				
CONTACTO 25D	CONTACTO 9D	MNEMONICO	INTERFAZ V24	INTERFAZ RS-232C
7	5	SG	102	AB
2	3	TD	103	BA
3	2	RD	104	BB
4	7	RTS	105	CA
5	8	CTS	106	CB
6	6	DSR	107	CC
20	4	DTR	108/2	CD
8	1	CD	109	CF
22	9	RI	125	CE

Tabla 1.6 Equivalencias de conectores de 9 y 25 contactos

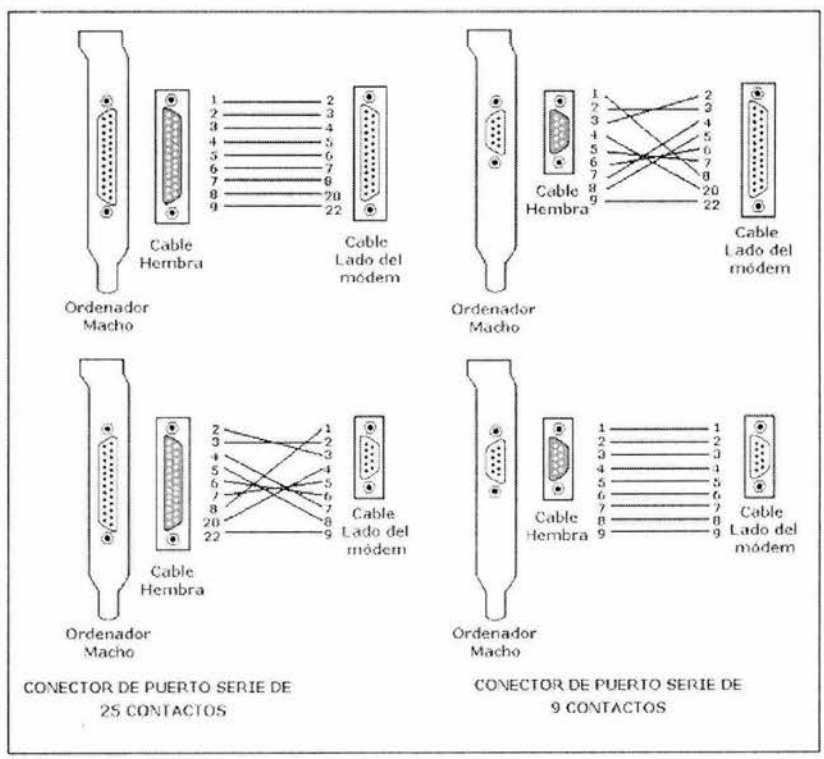


Figura 1.5 Tipos de conectores y de cables para puerto serie

1.8 ACCESO DEL PC AL PUERTO SERIE

1.8.1 El procesador

Un microprocesador o CPU (Central Processor Unit, unidad central de procesos) dispone de cuatro registros principales de 16 bits, conocidos como AX, BX, CX y DX. Cada uno de estos registros puede ser considerado como dos registros de 8 bits, por lo que tenemos un total de 8 registros: AH, AL, BH, BL, CH, CL, DH y DL. Desde un punto de vista práctico, cargar un número de 16 bits en AX es lo mismo que cargar su byte menos significativo en AL y el byte más significativo en AH. Se le llama byte menos significativo al que representa un valor menor (el que está en la parte de la derecha cuando escribimos el número), y byte más significativo al que representa un valor mayor (el byte de la izquierda). Por ejemplo, en el número 5A12h, el byte menos significativo es el 12h (00010010 bin), y el más significativo, el 5Ah (01011010 bin).

Dado que un registro puede contener como máximo un número de 16 bits, y dado que con 16 bits sólo podemos direccionar un máximo de 64K posiciones de memoria, para direccional toda la memoria de un ordenador se necesita utilizar dos registros. El primero de ellos, el más significativo, se conoce con el nombre de **segmento**, e identifica al bloque de 64K donde está localizada la memoria que se pretende direccionar. El segundo de ellos, el menos significativo, se conoce por **offset** (desplazamiento), e identifica la posición exacta de la memoria dentro del bloque de 64K identificado por el segmento. Los registros CS (Code Segment, segmento de código), SS (Stack Segment, segmento de pila), DS (Data Segment, segmento de dato) y ES (Extra Segment, segmento extra) guardan distintos segmentos de distintas direcciones. El offset, por su parte, puede ser guardado en cualquiera de los registros AX, BX, CX y DX, o en cualquiera de los registros especialmente pensados para ello, como son el registro IP (Instruction Pointer, puntero de instrucción) para el offset del segmento de código CS; SP (Stack Pointer, puntero de pila) y BP (Base Pointer, puntero base) para el offset del segmento de pila SS, y SI (Source Index, índice fuente) para el offset del segmento de datos DS.

Al modo de direccionamiento segmento: offset se le conoce con el nombre de modo real. Además de éste, existen otros modos de direccionamiento, como son el modo protegido, utilizado a partir de la aparición del microprocesador 80286 para direccionar más de 1 Mbyte, y el modo virtual real, utilizado a partir de la aparición del microprocesador 386SX para poder ejecutar varios programas simultáneamente.

Físicamente, entre el microprocesador y los distintos dispositivos conectados a el existen varias conexiones, conocidas por el nombre de bus. Se le da el nombre de bus a un conjunto de circuitos que tienen asignada una función común. Los ordenadores PC disponen de tres tipos de bus:

- Bus de datos, el cual transporta la información que el microprocesador intercambia con el exterior (bits que lee o escribe en las posiciones de memoria). Este bus consta de 8 circuitos en los microprocesadores 8088, 16 en los microprocesadores 8086, 80286 y 386SX, 32 circuitos en los 386DX y 486 y 64 circuitos en los Pentium.

- Bus de direcciones, que es por donde el microprocesador selecciona cada una de las posiciones de memoria con la que la que se quiere comunicar. El bus de direcciones consta de 20 circuitos en los microprocesadores8088 y 8086, 24 circuitos en los 80286 y 32 circuitos en los demás (386, 486 Y Pentium).

- Bus de control, el cual consiste en un número de circuitos a través de los cuales el microprocesador controla los distintos dispositivos con los que serelaciona.

Como ejemplo de lo anterior, cuando la CPU desea leer una posición de memoria, por un lado debe enviar la dirección a través del bus de direcciones, y por otro, debe activar la línea de lectura del bus de control. Como resultado, obtiene el contenido de dicha posición de memoria en el bus de datos. La fig. 1.6 nos muestra un diagrama a bloques del microprocesador y los diferentes tipos de bus.

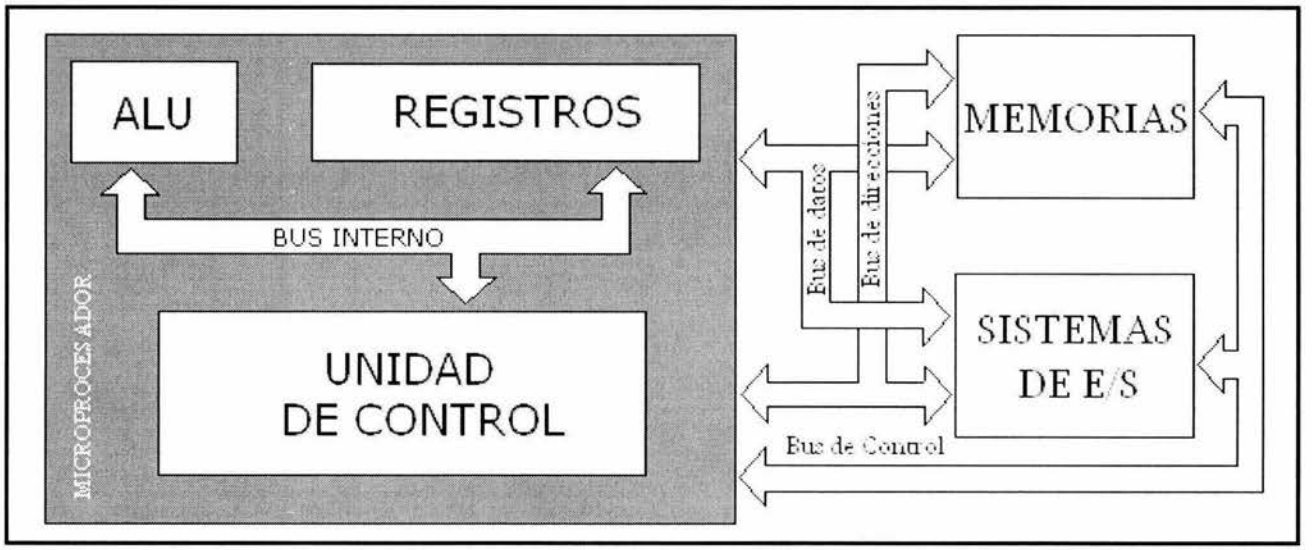


Figura 1.6 El microprocesador

Direcciones E/S

La CPU accede a la información contenida en la memoria RAM direccionando cada una de las posiciones que tiene asignadas esta memoria. De la misma forma, cada dispositivo que necesita comunicarse con la CPU tiene asignado un rango de direcciones de entrada/salida de información, o direcciones E/S (Entrada/Salida o I/O Input/Output). Cada rango de direcciones E/S sólo puede ser utilizado por un único dispositivo. Por ejemplo, cuando el teclado tiene datos para el sistema, pone estos datos en una dirección, y la CPU los recoge de ahí. Esta misma operación la realizan el ratón, los puertos serie o cualquier otro dispositivo.

Los PC dedican 1024 direcciones a E/S. Exactamente desde la posición 000 a la 3FF. De estas direcciones, las 256 primeras (de 000 a 0FF) sólo están disponibles para componentes del propio sistema (reloj, NMI, DMA, etc.), quedando el resto de las direcciones para las tarjetas de expansión, en total 768 localizaciones, situadas entre las posiciones 100 y 3FF. Las asignaciones más habituales que se les suelen dar a estas direcciones E/S en los PC están mostradas en la tabla 1.7, aunque estas direcciones dependen no sólo del tipo de ordenador, sino también del sistema operativo que se tenga instalado.

Para que el microprocesador acceda a cualquiera de estas direcciones E/S, debe recibir del programa que se esté ejecutando una instrucción IN u OUT en código máquina. Por ejemplo, para enviar el contenido del registro AL al puerto 3F8h, el comando adecuado en ensamblador es:

OUT 3F8h, AL

Los distintos lenguajes de programación ofrecen distintas formas de ejecutar estos comandos IN y OUT.
Cuando la CPU recibe uno de estos comandos, activa la línea del bus de control de lectura o de escritura de E/S. Cuando el dispositivo correspondiente detecta que la línea de lectura o de escritura del bus de control está activa y que la dirección del bus de dirección se corresponde con la suya, pone/recoge los datos correspondientes en el bus de datos.

A diferencia de una posición de memoria, un dispositivo E/S puede hacer muchas más cosas que simplemente leer o escribir datos en el bus de datos cuando recibe la orden correspondiente de la CPU. Por ejemplo, cuando la controladora del puerto serie (UART) recibe una petición de lectura de su registro del buffer del receptor, no solamente envía el nuevo byte recibido al bus de datos, sino que también borra dicho registro para que una nueva petición de lectura no pueda interpretar erróneamente que se han recibido dos datos idénticos seguidos.

DIRECCIONES E/S USADAS HABITUALMENTE EN LOS PC	
RANGO DE DIRECCIONES	USO ASIGNADO
00-0F	Controlador de DMA #1 (chip 8237)
20-21	Controlador programable de interrupciones #1 (chip 8259A)
40-43	Reloj programable (chip 8253)
60-63	Controlador de periféricos (chip 8255A). Sólo en los XT
60-64	Controlador de teclado (chip 8742). Sólo en los AT
70-71	Dirección de acceso de iniciación de RAM. Sólo en los AT.
80-8F	Registro de página de DMA
A0-A1	Controlador programable de Interrupciones #2 (chip 8259A). Sólo en los AT
A0-AF	Registro de máscara NMI. Sólo en los XT
C0-DF	Controlador de DMA #2 (chip 8237). Sólo en los AT.
F0-FF	Coprocesador matemático. Sólo en los AT.
1F0-1F8	Controlador de unidades de disco duro. Sólo en los AT
200-20F	Controlador de joystick
210-217	Unidad de expansión
238-23B	Ratón
23C-23F	Ratón (alternativo)
258	Tarjeta de memoria expandida (LIM)
278-27F	LPT2
2B0-2DF	EGA
2E8-2EF	Puerto serie COM4
2F8-2FF	Puerto serie COM2
300-30F	Tarjeta de red de área local Ethernet
320-32F	Controlador de unidades de disco duro. Sólo en los XT
330-337	Controlador de unidades de discos Bernoulli
378-37F	Puerto paralelo LPT1
380-38F	Tarjeta SDLC
3A0-3AF	Tarjeta BSC
3B0-3BF	Adaptador monocromo
3BC-3BF	Puerto paralelo LPT3
3D0-3DF	CGA
3E8-3EF	Puerto serie COM3
3F0-3F7	Controlador de unidades de discos flexibles
3F8-3FF	Puerto serie COM1

Tabla I.7 Direcciones E/S usadas habitualmente en los PC

Interrupciones software.

Un microprocesador puede verse afectado por dos tipos de interrupciones: interrupciones software e interrupciones hardware, las llamadas interrupciones software son rutinas del sistema operativo a las que el programador puede llamar para facilitar su tarea de programación. Estas rutinas realizan determinados procesos de control de los dispositivos, conectados al microprocesador. De no existir dichas interrupciones, cada programador tendría que escribir sus propias rutinas de control, con las consiguientes complicaciones que eso

traería. El sistema operativo contiene dos tipos de interrupciones software: las llamadas interrupciones de la BIOS (Basic Input/Output System, sistema básico de entrada/salida), y las interrupciones del DOS.
Las interrupciones de la BIOS son rutinas relacionadas con la lectura y escritura en disco, presentaciones en pantalla, lectura del teclado, control de los puertos de comunicaciones, actualización de la fecha y hora, manejo de errores, etc. Existen 12 interrupciones de la BIOS, divididas en 5 grupos:

- **Servicios de dispositivos periféricos.**
- INT 10H – Servicios de la pantalla de video.
- INT 13H – Servicio de disco.
- INT 14H – Servicios de comunicaciones (RS-232).
- INT 15H – Servicio de casette.
- INT 16H – Servicios de teclado.
- INT 17H – Servicios de la impresora.

- **Servicios de estado del equipo.**
- INT 11H - Servicios de lista de elementos del equipo.
- INT 12H - Servicio de cálculo del tamaño de la memoria.

- **Servicios de fecha y hora.**
- INT 1AH - Servicios de reloj.

- **Servicios de la pantalla de impresión.**
- INT 5H - Impresión de pantalla.

- **Servicios especiales.**
- INT 18H - Activación del Basic de la ROM.
- INT 19H - Activación de la rutina de arranque del ordenador.

Por su parte, el DOS ofrece dos tipos de servicios de acceso a las interrupciones; por un lado, directamente por medio de una Instrucción INT, y por otro, accediendo a las funciones que se invocan a través de la interrupción 21 H, especificando anteriormente en el registro AH el número de función que se invoca.

El DOS tiene los siguientes servicios de interrupción:

- INT 20H: Termina la ejecución de un programa.
- INT 21H: Llamada a función. Se debe colocar en el registro AH el número de la función, siendo las más importantes las siguientes:
  0 Terminación de un programa y devolución del control al DOS.
  1 Entrada de carácter con eco.
  2 Salida a la pantalla.
  3 Entrada por el puerto serie.
  4 Salida por el puerto serie.
  5 Salida a la impresora.
  6 E/S directa a pantalla.
  7 Entrada directa de carácter sin eco.
  8 Entrada de carácter sin eco.
  9 Visualización de una cadena de caracteres.
  10 Entrada desde el teclado. .
  11 Comprobación del estado de entrada.
  12 Borra registro de entrada y procede a una introducción de datos.
  13 Inicializar la unidad de disco.
  15 Abrir un fichero.
  16 Cerrar un fichero.
  17 Búsqueda del primer fichero.

18 Búsqueda del próximo fichero.
19 Borrar un fichero.
20 Leer registro de fichero secuencial.
21 Escribir registro de fichero secuencial.
22 Crear fichero.
23 Renombrar fichero.
33 Leer fichero de acceso aleatorio.
34 Escribir fichero de acceso aleatorio.
35 Cálculo del tamaño del fichero.
42 Obtener la fecha.
43 Fijar la fecha.
44 Obtener la hora.
45 Fijar la hora.

- INT 22H: Dirección de terminación. Guarda la dirección donde se transfiere el control cuando termina la ejecución de un programa.
- INT 23H: Dirección de Break. Dirección de la rutina que se ejecuta cuando se pulsa Ctrl- Break.
- INT 24H: Dirección del Manejador de errores críticos. Esta rutina se ejecuta siempre que se produce un error crítico.
- INT 25H: Lectura directa de sectores del disco.
- INT 26H: Escritura directa de sectores del disco.
- INT 27H: Termina un programa y devuelve el control al DOS sin borrar el programa de la memoria. Esta interrupción es la que utilizan los programas residentes y los virus informáticos.

Interrupciones hardware

Las interrupciones hardware son de dos tipos: las interrupciones hardware internas y la interrupciones hardware externas. La interrupciones hardware internas también reciben el nombre de interrupciones lógicas, y son invocadas por el propio microprocesador cuando se produce alguna operación incorrecta, como, por ejemplo, un intento de dividir por cero.

Las interrupciones hardware externas son provocadas por los distintos dispositivos periféricos que están conectados al ordenador. Cada dispositivo tiene acceso a una línea (circuito físico) de petición de interrupción diferente. A estas líneas se les conoce por el nombre de IRQ, Interrupt Request Line. Los ordenadores 8086 y 8088 tienen 8 IRQ (IRQ0 a IRQ7), mientras que los 286 y superiores tienen 16 IRQ (IRQ0 a IRQ15). La tabla 1.8 indica las IRQ mas usadas en los PC.

La razón de la existencia de las interrupciones externas es que la CPU no está continuamente mirando a las direcciones de E/S de cada dispositivo para ver si hay nuevos datos, sino que cada vez que uno de estos dispositivos tiene nuevos datos que intercambiar con la CPU hace sonar un timbre para llamar su atención. Este timbre son las interrupciones IRQ (Interrupt Request). Cada vez que un dispositivo activa una de estas interrupciones, la CPU deja todo lo que estaba haciendo y ejecuta la rutina correspondiente a la interrupción activada.

Las direcciones de las rutinas que se deben ejecutar para cada una de las interrupciones están almacenadas en lo que se llama tabla de vectores de interrupción.

Esta tabla es una serie de direcciones que están almacenadas en la memoria RAM a partir de la posición 8 (08h). Si desea ver la tabla de vectores de interrupción, puede usar la utilidad DEBUG del DOS, empleando luego el comando D 0:0, con lo que conseguirá leer los primeros 128 bytes de la memoria RAM. Cada vector está representado por 4 bytes, los dos primeros indican el offset, y los dos siguientes, el segmento.

IRQ USADAS HABITUALMENTE EN LOS PC		
ORDENADOR	INTERRUPCION	DISPOSITIVO
PC-XT y PC-AT	0	Reloj
	1	Teclado
	2	Disponible en los XT. Usado en los AT como pasarela para las IRQ 8-15.
	3	COM2
	4	COM1
	5	Disco duro en los XT. Disponible en los AT.
	6	Disco flexible.
	7	LPT1
Sólo en los PC-AT	8	Reloj
	9	Red de área local
	10	Disponible
	11	Disponible
	12	Disponible
	13	Coprocesador
	14	Disco duro
	15	Disponible

Tabla 1.8 IRQ usadas habitualmente en los PC

1.8.2 El PIC 8259A

Las líneas físicas IRQ procedentes de cada periférico no van directamente al chip de la CPU, sino que existe un chip intermedio que se encarga de controlar las interrupciones. Este chip, llamado PIC (Programmable Interrupt Controller, controlador programable de interrupciones), es el 8259A. Cada uno de estos chips puede controlar hasta 8 interrupciones, por lo que los primeros PC disponían de un solo chip 8259A, mientras que los PC de tipo AT o posteriores disponen de dos de estos chips.

El PIC da prioridad a las interrupciones y previene del caos que le supondría a la CPU el hecho de que varias interrupciones ocurriesen al mismo tiempo. El PIC le da la prioridad más alta a la IRQ 0, y la prioridad más baja a la IRQ 7. Si coinciden varios dispositivos demandando la atención de la CPU, el PIC le pasa cada una de estas interrupciones a la CPU en orden de prioridad. La fig. 1.7 indica un diagrama a bloques del PIC y sus conexiones con otros dispositivos.

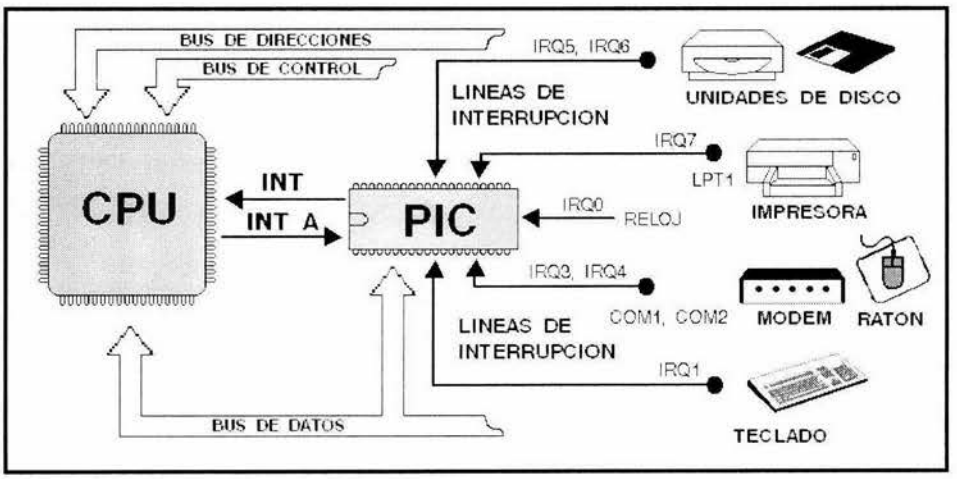


Figura 1.7 PIC. Controlador programable de interrupciones

El PIC puede ser programado, por lo que puede modificarse su forma de comportamiento. Por defecto, el PIC es inicializado por el POST (Power On Self Test, auto revisión de encendido), que es una serie de rutinas que ejecuta el ordenador cada vez que se enciende.

El PIC dispone de un registro donde son habilitadas cada una de las interrupciones de los distintos dispositivos. Para habilitar una interrupción, hay que poner el bit correspondiente del registro a cero (bit 3 para IRQ3, bit 4 para IRQ4, etc.). Esto se lleva a cabo leyendo el registro con un comando IN a la dirección E/S 21h, poniendo el bit correspondiente a 0 y escribiendo de nuevo el valor en la dirección E/S 21h con un comando OUT.

Cuando el PIC recibe una interrupción, pone a nivel alto la línea INT de la CPU. La CPU acepta dicha interrupción devolviéndole otra señal al PIC (INT A), y este le envía el número de la interrupción a través del bus de datos. El número que le envía el PIC a la CPU por el bus de datos es el resultado de sumarle 8 al número de la interrupción (por ejemplo, en el caso de IRQ3 el número enviado sería el 11, 0Bh). En este momento, la CPU guarda en una pila la dirección actual del programa que estuviese realizando y llama a la dirección de la memoria indicada por el puntero de la tabla de vectores de interrupción correspondiente a la interrupción en cuestión. Dicho puntero contiene la dirección de la rutina que se debe ejecutar. Esta rutina recibe el nombre general de rutina del servicio de interrupciones, ISR (Interrupt Service Routine). En realidad, atender a la interrupción IRQ3 tiene el mismo efecto que ejecutar directamente el comando INT 0Bh.

Hay veces en las que la CPU puede estar realizando una tarea que no interesa que sufra ninguna interrupción; en esos casos, el programador de la aplicación debe indicarle a la CPU que ignore las interrupciones. Para ello se utiliza el comando CLI (Clear Interrupt Flag). Para activar de nuevo la atención de las interrupciones se usa el comando STI (Set Interrupt Flag).

Al inhabilitar las interrupciones mediante CLI, lo que se hace es cambiar el estado de las banderas de interrupciones de la CPU (IF, Interrupt Flag). A pesar de ello, el PIC seguirá enviándole a la CPU la señal sobre la línea INT de que ha ocurrido una interrupción, pero la CPU no aceptará dicha interrupción.

Como se ha dicho anteriormente, el PC mantiene una tabla de vectores de interrupción, la cual contiene una dirección de 4 bytes por cada interrupción. Antes de inicializar cualquier elemento en esta tabla, lo primero es comprobar que dicha interrupción no está ya ocupada por otro manejador de interrupción. Para averiguar si un elemento ya tiene asignada una dirección ISR se pone el número de la interrupción en AL, el valor 35H en AH y se llama a la interrupción 21h. Como respuesta, ES:BX apuntarán al valor ISR existente.

Existen dos razones para averiguar si ya hay instalada una dirección de ISR antes de instalar la nuestra. Primero, que cuando se termine con nuestro programa (nuestra interrupción) se querrá restablecer el puntero al valor previo. Segundo, que algunos PC soportan compartir IRQ, por lo que en este caso cada dispositivo puede tener su propia rutina de interrupciones, y la rutina direccionada por el puntero debería llamar al otro ISR cada vez que ocurre una interrupción.

Una vez que se ha averiguado la existencia de otra posible ISR en la interrupción, podemos proceder a instalar nuestra propia ISR. Para hacerlo, se usa la interrupción 21h del DOS, función 25h. Basta con poner la dirección de nuestra rutina en DS:DX, el valor 25h en AH, el número de la interrupción a reemplazar en AL y llamar a INT 21h.

Cuando ocurre cualquier interrupción y se llama a su ISR correspondiente, éste comienza guardando los contenidos de los registros en una pila, para que cuando termine el trabajo de la interrupción poder devolver el procesador al estado que tenia antes de que fuese atendida la interrupción. La rutina que atiende la interrupción debe averiguar la causa de la misma, y debe reaccionar de acuerdo con ello. El ISR debe enviarle también una señal de aceptación de interrupción al PIC. Esto se hace enviando el valor 20h a la dirección E/S 20h. Al valor 20h se le conoce como EOI, End Of Interrupt, o fin de interrupción. Una vez ejecutada la rutina de la interrupción, se debe restaurar el contenido de los registros mediante el comando IRET (Interrupt Return).

1.8.3 Formato de la información

Para intercambiar información entre dos equipos, primero es fundamental que tanto el equipo transmisor como el receptor utilicen un mismo formato de la información.

Como ya se ha comentado, las comunicaciones asíncronas basan su transferencia de información en caracteres, cada uno de los cuales tiene una estructura especial, formando lo que se llama unidad de datos serie, SDU (Serial Data Unit). Cada SDU consta de un bit de comienzo (start bit), un número determinado de bits de datos, un bit de paridad y un bit de parada (stop bit).

El tamaño del carácter SDU es definido por el usuario vía software, pudiendo fijar el número de bits de datos por carácter entre 5 y 8 bits, siendo opcional la inclusión del bit de paridad, y pudiendo fijarse el bit de parada entre 1 y 2 bits, Eso significa que el carácter SDU puede tener una longitud entre 7 y 12 bits.

Bit de paridad

El método de paridad es un sistema simple de protección contra los errores de transmisión. Este método puede detectar el 100% de los errores que afectan a un sólo bit, sin embargo la probabilidad de detectar los errores que afectan a más de un bit es tan solo del 50%. Eso quiere decir que este sistema resulta útil y práctico cuando el medio sobre el que se van a transmitir los datos es suficientemente seguro; en cualquier otro caso el sistema más apropiado es el método CRC (Control de errores de transmisión). La ventaja del método de paridad es que cualquier puerto serie puede generar y comprobar la paridad mediante su propio hardware.

La paridad consiste en que cada carácter incluye un bit, llamado bit de paridad, el cual toma el valor 0 o 1, de forma que el número total de 1 del carácter sea par (paridad par) o impar (paridad impar). En el caso de los puertos serie, existen las siguientes posibilidades:

- SIN PARIDAD: No se utiliza bit de paridad.
- PARIDAD PAR: Se fija el valor del bit de paridad de forma que el número total de bits unos, contando el bit de paridad, sea par.
- PARIDAD IMPAR: Se fija el valor del bit de paridad de forma que el número total de bits unos, contando el bit de paridad, sea impar.
- MARCA: El bit de paridad se fija siempre en el valor 1.
- ESPACIO: El bit de paridad se fija siempre en el valor 0.

Los bits de MARCA y ESPACIO son utilizados para detectar los errores de transmisión cuando éstos afectan al bit de paridad. Es cierto que el hecho de que esté cambiado el bit de marca o de espacio no significa forzosamente que exista error en los bits de información, sin embargo este procedimiento de detección de error es ampliamente utilizado.

Errores en el puerto serie

Si el transmisor y receptor no utilizan un mismo formato de SDU, puede ocurrir que el receptor realice una mala interpretación de los datos que recibe. En este sentido, se pueden dar los siguientes errores:

- ERROR DE TRAMA (framing error): Este error se produce cuando el receptor detecta un bit de parada (stop) no válido. Esto se da cuando el formato de SDU transmitido no es el mismo que el que esperaba recibir el receptor.
- ERROR DE CORTE (break error): Si la línea de recepción está en el estado lógico bajo durante un largo tiempo, entonces el receptor asume que la conexión se ha roto, ya que está normalizado que cuando no se transmiten datos el transmisor debe poner esta línea en el estado lógico alto.
- ERROR DE SATURACIÓN (overrun error): Este error ocurre cuando los datos llegan al receptor más de prisa de lo que éste es capaz de asimilar. En este caso, algunos bytes serán escritos en el buffer antes de que haya dado tiempo a procesar el dato que contenía anteriormente.

- ERROR DE PARIDAD (parity error): Este error indica que el cálculo de] bit de paridad no coincide con el bit de paridad del carácter. La razón de este error puede ser que los datos hayan sido dañados durante la transmisión o que los modos de paridad del transmisor y del receptor sean distintos.

1.8.4 Funciones de comunicaciones de la BIOS

A través de la BIOS se pueden transmitir, recibir y controlar de una forma simple hasta cuatro puertos serie. Cada puerto serie dispone de una dirección de entrada y salida, la cual está almacenada en el área de datos de la BIOS. Este área de datos también guarda unos valores de temporización usados para prevenir esperas sin fin causadas por la falta de respuesta del dispositivo conectado al puerto serie. El área de datos de la BIOS para el puerto serie se muestra en la tabla 1.9.

ÁREA DE DATOS DE LA BIOS PARA EL PUERTO SERIE	
DIRECCION	CONTENIDO
40:00	Dirección base COM1
40:02	Dirección base COM2
40:04	Dirección base COM3
40:06	Dirección base COM4
40:7C	Temporización COM1 (segundos)
40:7D	Temporización COM2
40:7E	Temporización COM3
40:7F	Temporización COM4
40:11	Número de puertos serie (3 bits menos significativos)

Tabla 1.9 Área de datos de la BIOS para el puerto serie

A diferencia del control directo de la UART, las funciones de la BIOS no utilizan las características de interrupción, por lo que el sistema de programación de la comunicación a través de la BIOS se basa en realizar sondeos reiterados del estado de la UART. Con las funciones de la BIOS sólo se pueden alcanzar velocidades de 9600 bps o inferiores.

La BIOS ofrece la interrupción l4h para acceder al puerto serie no sólo para transmitir y recibir información, sino también para configurarlo. Las funciones que ofrece esta interrupción son las siguientes:

- Función 00h:Inicializa el puerto serie y configura el formato de datos y la velocidad de transferencia.
- Función 0lh: Escribe caracteres en el puerto serie.
- Función 02h: Lee caracteres en el puerto serie.
- Función 03h: Lee el estado del puerto serie.

Cada una de estas funciones devuelve un byte de estado de línea en el registro AH (ver Tabla 1.10). Alguna de ellas también devuelve un byte de estado del módem en el registro AL (ver Tabla 1.11).

BYTE DE ESTADO DE LÍNEA (AH)				
BIT		SIGNIFICADO	VALOR 1	VALOR 0
0	EMP	Datos recibidos	Listo	Sin datos
1	OVR	Error saturación	Error	Sin error
2	PAR	Error paridad	Error	Sin error
3	FRM	Error de trama	Error	Sin error
4	BRK	Corte	Ocurrió	No ocurrió
5	THR	Registro transmisor	Libre	Ocupado
6	TSR	Registro desplazamiento transmisor	Libre	Ocupado
7	TIM	Temporización	Sobrepasada	Correcta

Tabla 1.10 Byte de estado de línea

BYTE DE ESTADO DEL MODEM (AL)				
BIT		SIGNIFICADO	VALOR 1	VALOR 0
0	DCT	Delta CTS	Detectado	No detectado
1	DDT	Delta DTR	Detectado	No detectado
2	DRI	Delta RI	Detectado	No detectado
3	DDC	Delta DCD	Detectado	No detectado
4	CTS	CTS. Listo para enviar	Listo	No preparado
5	DSR	DSA. Terminal preparado	Listo	No preparado
6	RI	RI. Indicador de llamada	Detectado	No detectado
7	DCD	DCD. Detección de portadora	Detectado	No detectado

Tabla 1.11 Byte de estado del módem

Parámetros de la comunicación. Función 0

Cuando se va a utilizar el puerto serie, la primera operación que hay que llevar a cabo es su configuración. La función 00h se encarga de este trabajo. Para llamar la función 00h, primero se tienen que colocar determinados valores en los registros AH, DX y AL. Estos valores son los siguientes:

AH valor 0 (función 0)
DX número del puerto a utilizar (de 0 a 3, donde 0 es el COM1).
AL valor de configuración (ver Tabla 1.12).

Una vez fijados dichos valores se llama a la interrupción 14h.

Con las funciones de la BIOS tenemos ciertas limitaciones de configuración de los puertos serie, pudiendo configurar exclusivamente los siguientes parámetros:

Bits de datos	7 u 8 bits por carácter (no 5 ni 6)
Bit de parada	1 o 2 (no 1,5)
Paridad	par o impar (no marca o espacio)
Velocidad	desde 110 hasta 9600 (no superior)

Como puede ver en la tabla 1.12, el número de bits de datos se fija con los bits 0 y 1 del byte de configuración (bits DBT). Esos bits pueden tomar los siguientes valores:

10	7 bits de datos por carácter.
11	8 bits de datos por carácter.

El número de bits de parada se fija con el bit 2 (STP), pudiendo tomar los valores 0 para indicar un solo bit de parada, o 1 para indicar 2 bits de parada. El tipo de paridad es fijado con los bits 3 y 4 (PAR), pudiendo fijarse la paridad par asignándole a ambos bits el valor 1, o la paridad impar poniendo el bit 3 a 1y el 4 a 0. Por último, la velocidad es fijada con los bits 5, 6 Y 7 (bits BDR). Los valores posibles son los siguientes:

000	110 bps	100	1200 bps
001	150 bps	101	2400 bps
010	300 bps	110	4800 bps
011	600 bps	111	9600 bps

Como puede ver, los bits BDR no pueden definir velocidades mayores de 9600 bps. Si se está interesado en definir velocidades superiores a 9600 bps, con la función 04h se puede llegar hasta los 19200 bps y mediante una programación directa de la UART se puede llegar hasta los 115,500 bps.

BYTE DE CONFIGURACION DEL PUERTO SERIE				
BIT		SIGNIFICADO	VALOR 1	
0 1	DBT DBT	Número de bits de datos	10=7 bits 11=8 bits	
2	STP	Bits de parada	0= 1 bit	1= 2 bits
3 4	PAR PAR	Paridad	00= Ninguna 10= Impar	01= Ninguna 11= Par
5 6 7	BDR BDR BDR	Velocidad bps.	000=110 001=150 010=300 011=600	100=1200 101=2400 110=4800 111 =9600

Tabla 1.12 Byte de configuración del puerto serie

Una vez llamada la función 00h de la interrupción 14h, ésta devuelve los siguientes valores:

- El byte del registro de estado de línea en AH
- El byte del registro de estado del módem en AL

Como ejemplo de lo anterior, la siguiente rutina en ensamblador configura el puerto COM2 para trabajar a 2400 bps, sin bits de paridad, 8 bits de datos y 1 bit de parada:

```
MOV ah, 00h ; Carga el número de la función en ah
MOV al, A3h ; Byte de configuración 10100011
MOV dx, 01h ; COM2
INT 14h
```

Transmitir un carácter. Función 1

La función 1 es la encargada de realizar las operaciones adecuadas para transmitir un carácter. Para llamar a esta función, primero se tienen que colocar determinados valores en los registros ah, dx y al. Estos valores son los siguientes:

AH valor 1 (función 1)
DX número del puerto a utilizar.
AL carácter a transmitir.

Una vez fijados dichos valores, se llama a la interrupción 14h.

Hay que tener en cuenta que el carácter no será transmitido a menos que los valores de las líneas CTS, RTS, DSR y DTR sean los adecuados. En este sentido las líneas DTR y RTS deberán estar a nivel alto. La BIOS esperará un tiempo prefijado para que estas líneas cumplan dicha condición. Si eso no ocurre, devolverá un error de temporización, lo que quiere decir que antes de llamar a esta función lo normal es llamar a la función 3 (estado del puerto serie) para asegurarse que las condiciones son favorables.

Una vez llamada esa función, en AH tendremos el byte del registro de estado de línea. Si el bit 7 de este registro está a 0, es que la transmisión se efectuó con éxito, pero si este bit tiene el valor 1, indica que ocurrió un error. El registro al seguirá teniendo el byte a transmitir.

Recibir un carácter. Función 2

La función 2 es utilizada para leer los caracteres recibidos por el puerto serie. Para llamar a esa función, los registros que hay que fijar de antemano son los siguientes:

AH valor 2 (función 2)
DX número del puerto a utilizar (de 0 a 3).

Una vez fijados dichos valores, se llama a la interrupción 14h. Cuando ocurre eso, la BIOS espera hasta que se reciba un carácter por el puerto serie o hasta que transcurra un tiempo de temporización preestablecido. Si durante ese tiempo se recibe un carácter, la función lo devuelve en el registro AL. Si, por el contrario, se llega al final de la temporización sin que se reciba ningún carácter, la función devolverá en el registro AH el byte de estado de línea con el bit 7 puesto a 1. Hay que aclarar que al igual que en el caso de la transmisión, la BIOS establece una conexión al puerto serie siguiendo estrictamente el estándar RS-232C por lo que hay que tener en cuenta el estado de las señales CTS, RTS, DSR y DTR. En ese caso, las señales DTR y DSR deben estar a nivel alto, si eso no ocurre antes de que se cumpla un tiempo prefijado la BIOS dará un error de temporización.

Desde un punto de vista práctico, para no perder tiempo agotando las temporizaciones de recepción de caracteres en vez de hacer llamadas repetitivas a la función 2, esas llamadas se deben realizar a la función 3 (estado del puerto serie), llamándose a la función 2 sólo cuando se sabe seguro que ha sido recibido un carácter. Ese procedimiento le da más control al programa, ya que se puede elegir un periodo de temporización propio, que puede ser variable, y además se puede aprovechar el tiempo de espera para hacer algo útil.

Como se ha comentado anteriormente, esta función devuelve los siguientes valores:

- El byte del registro de estado de línea en AH
- El byte recibido en AL

A continuación podemos ver un ejemplo de lectura de un carácter del puerto serie utilizando la función 02h:

```
MOV ah, 02h   ; Carga el número de la función en ah
MOV dx, 03h  ; COM4
INT 14h       ; Llama a la interrupción. La función espera y devuelve el carácter en el registro al
```

Estado del puerto serie. Función 3

Si se desea acceder al byte de estado de línea o al byte de estado del módem, los valores del primero son mostrados en el registro AL después de llamar a cualquiera de las funciones anteriores, pero el segundo sólo será mostrado cuando se llama a la función 0. Si se desea acceder a ambos bytes de estado directamente, se puede usar la función 03h de la interrupción 14h.

Para llamar a esta función, se deben pasar los siguientes valores:

AH valor 3 (función 3)
DX número del puerto a utilizar (de 0 a 3).

Una vez fijados dichos valores se llama a la interrupción 14h. Esta llamada devuelve en AH el valor del registro de estado de línea (ver tabla 1.9) y en AL el valor del registro de estado del módem (ver tabla 1.10).

Un ejemplo de lo anterior sería lo siguiente:

```
MOV ah, 03h ; Carga el número de la función en ah
MOV dx, 01h ; COM2
INT 14h; Llama a la interrupción
        ; El estado de línea es mostrado en ah y el estado del módem es mostrado en al.
```

Funciones extendidas

Las funciones que hemos visto anteriormente son funciones básicas que permiten el intercambio de información a través de un puerto serie. Esas funciones son válidas para cualquier ordenador de tipo PC; no obstante, los sistemas MCA y algunos sistemas AT tienen acceso a una serie de funciones adicionales conocidas como funciones extendidas. Esas funciones son las siguientes:

- Función 04h: Configuración extendida del puerto serie.
- Función 500h: lectura de control extendida.
- Función 501h: Escritura de control extendida.

La función 04h añade nuevas características de configuración no disponibles con la función 00h. Para acceder a esta función, se deben fijar primero los siguientes valores:

AH valor 4 (función 4)

AL 0 = Control de corte desactivado (modo normal).
1 = Control de corte activado. Se fuerza a que la salida permanezca a 0. Eso hace que la UART receptora corte la comunicación.

BL 0 = 1 bit de parada
1 = 2 bits de parada (1,5 si se fijan 5 bits de datos)

BH
0 = No hay bit de paridad
1 = Paridad impar
2 = Paridad par
3 = Espacio (siempre 0)
4 = Marca (siempre 1)

CL
0 = 110 bps
1 = 150 bps
2 = 300 bps
3 = 600 bps
4 = 1200 bps
5 = 2400 bps
6 = 4800bps
7 = 9600 bps
8 = 19200 bps
9 = 115200 bps

CH
0 = 5 bits de datos por carácter
1 = 6 bits de datos por carácter
2 = 7 bits de datos por carácter
3 = 8 bits de datos por carácter

DX número del puerto a utilizar (de 0 a 3).

Una vez fijados dichos valores, se llama a la interrupción 14h. Esa llamada devuelve en AH el valor del registro de estado de línea (ver tabla 1.10) y en Al el valor del registro de estado del módem (ver Tabla 1.11).

La función 500h (interrupción 14h) lee el byte del registro de control del módem de un puerto serie especificado. Para llamar a esa función sólo hay que fijar los valores de dos parámetros:

AX valor 500h (función 500h)
DX número del puerto a utilizar (de 0 a 3).

Esa función devuelve los siguientes datos:
AH registro de estado de línea
AL registro de estado del módem
BL registro de control del módem. Los valores de este registro son los siguientes:

bits 7-5 sin usar		bit 2	OUT1
bit 4	activa modo bucle	bit 1	RTS
bit 3	OUT2	bit 0	DTR

Por último, la función 501h (interrupción 14h) permite modificar el registro de control del módem para un puerto serie especificado. Para llamar a esta función, se deben pasar los siguientes valores:

AX valor 50lh (función 501)
BL registro de control del módem.
DX número del puerto a utilizar (de 0 a 3).

Esta función devuelve los siguientes datos:
AH registro de estado de línea.
AL registro de estado del módem.

1.9 La UART

Introducción

La UART (Universal Asynchronous Receiver and Transmitter, transmisor y receptor asíncrono universal) es un chip programable que se encarga de controlar la transferencia y recepción de datos asíncronos por el puerto serie. Las primeras versiones de los ordenadores personales del tipo PC traían el chip 8250 como UART. Este chip contenía un pequeño fallo, por lo que posteriormente salieron las versiones 8250A y 8250B. En cualquier caso, esos chips admiten una velocidad máxima de 9600 bps. Para mejorar esa característica, apareció el chip UART 16450 (también llamado 82450), pensado para los ordenadores del tipo AT y con el cual se pueden alcanzar velocidades de hasta 115200 bps.

TIPOS DE UART	
IDENTIFICACION	DESCRIPCION
8250	UART básica, con algunas limitaciones de actuación.
8250A	UART 8250 con correcciones de fallos (pero incompatible con BIOS).
8250B	UART 8250 mejorada compatible con BIOS.
16450	Versión mejorada de la 8250.
16550	16450 con FIFO receptor y transmisor.
16552	Dos UART 16550 en una.
16C454	Cuatro UART 16450 en una.
16C554	Cuatro UART 16550 en una.
16C1450	16450 mejorada.
16C1550	16550 mejorada.
82510	16450 con FIFO transmisor y receptor.

Tabla 1.13 Tipos de UART

1.9.1 Hardware de la UART 8250/16450

Como se ha dicho anteriormente, la UART es el chip que controla la transferencia y recepción de información a través del puerto serie. Eso implica que la UART tiene que hacer de intermediaria entre el puerto serie y el microprocesador o, dicho de otra forma, entre el puerto serie y el bus de datos y de direcciones del ordenador. Para poder llevar a cabo esa función, la UART dispone de patillas conectadas tanto a los bus de datos y de direcciones como a la interfaz RS-232C, así como otras patillas que manejan determinadas señales de control. (Ver fig. 1.8).

La UART es un chip electrónico de tecnología TTL (Transistor-Transistor Logic), lo que quiere decir que maneja señales de 0 y +5 voltios para representar las informaciones binarias 0 y 1. Por otro lado, la interfaz RS-232C maneja señales positivas (entre +3 y +15 voltios) para representar la información binaria 0, y señales negativas (entre -3 y -15 voltios) para representar la información binaria 1. Para poder resolver el problema de la adaptación entre la UART y las líneas de la interfaz serie existe lo que se llama un amplificador (EIA line driver).

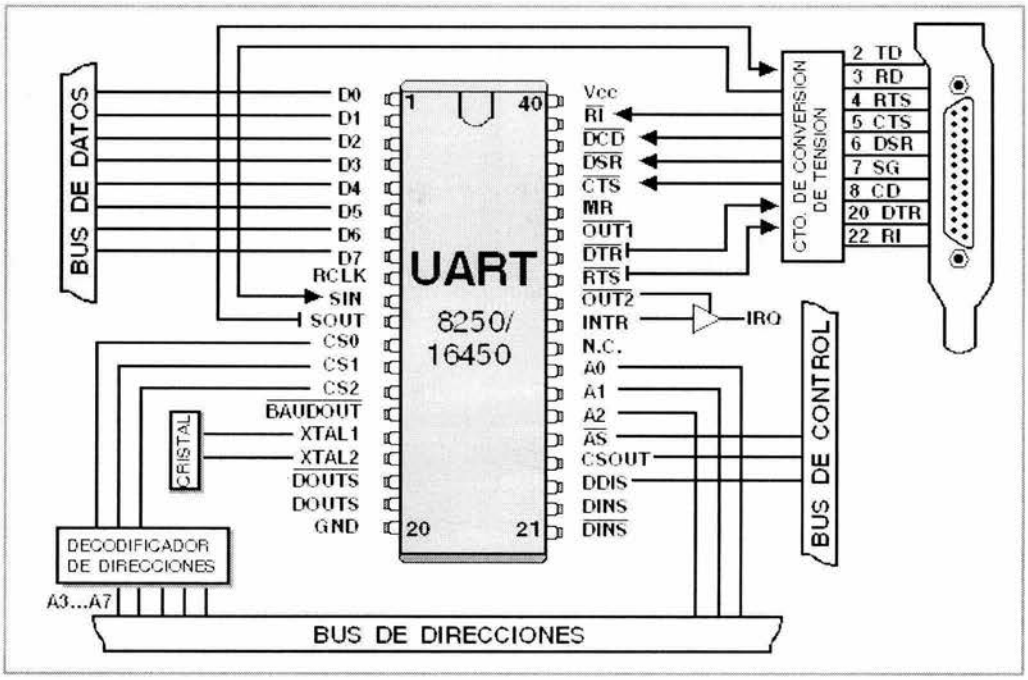


Figura 1.8 El chip UART 8250//6450

Los chips de la UART 8250/16450 vienen encapsulados según un estándar DIP (Dual In-line Package) de 40 patillas, siendo el significado de cada una de sus conexiones el siguiente:

D0- D7 Contactos 1-8. Estos 8 contactos forman el bus de datos bidireccional entre la CPU y el 8250/16450. A través de este bus se transmite la información de estado, de control y de datos.

RCLK Contacto 9. Reloj receptor (Receiver Clock). A esta conexión le llega una señal de reloj equivalente a 16 veces la velocidad de recepción de datos.

SIN Contacto 10. Entrada serie (Serial In). Entrada de datos serie.

SOUT Contacto 11. Salida serie (Serial Out). Salida de datos serie.

CS0, CS1, CS2 Contactos 12 al 14. Selección de chip (Chip Select). Cuando CS0 y CS1 están a nivel alto y /CS2 está a nivel bajo, la UART es la encargada de transferir los datos a y desde la CPU.

/BAUDOUT Contacto15. Reloj transmisor. Por esta patilla sale una señal de reloj con una frecuencia 16 veces superior a la velocidad de transmisión. Si se conecta esta patilla con RCLK, entonces la velocidad de transmisión y la de recepción serán iguales.

XTAL1, XTAL2 Contactos 16 y 17. Estas patillas están conectadas al cristal u oscilador externo que fija la frecuencia de referencia de la UART.

/DOUTS, DOUTS Contactos 18 y 19. Cuando la CPU va a escribir información de datos o de control en los registros internos de la UART, /DOUTS se pone a nivel bajo (o DOUTS a nivel alto).

GND Contacto 20. Tierra (Ground). Potencial de tierra (0 voltios).

/DINS, DINS Contactos 21 Y 22. Cuando la CPU va a acceder a la información de los registros internos de la UART, /DINS se pone a nivel bajo (o DINS a nivel alto).

DDIS Contacto 23. Esta patilla se pone a nivel bajo cuando la CPU va a leer datos de la UART.

CSOUT Contacto 24. Esta patilla ofrece un nivel alto cuando la CPU selecciona la UART usando CS0, CS 1 y /CS2.

/AS Contacto 25. Un pulso en esta patilla valida la señal de dirección A0-A2 y la señal de dirección de chip CS0-/CS2.

A2-A0 Contacto 26-28. Estas tres señales de selección de registro o de dirección determinan el registro al que accede la CPU para leer o escribir.

INTR Contacto 30. La UART pone esta patilla a nivel alto si detecta una condición de interrupción y el bit relativo al registro de habilitación de interrupción está activado.

/OUT2 Contacto 31. Esta patilla puede ser programada por el usuario. En los PC, esto suele ser utilizado como una habilitación de interrupción vía una puerta lógica.

/RTS Contacto 32. Petición de envío (Request to Send). Señal RTS.

/DTR Contacto 33. Terminal preparado (Data Terminal Ready). Señal DTR.

/OUTl Contacto 34. Esta patilla puede ser programada por el usuario.

MR Contacto 35. Inicialización principal (Master Reset). Cuando esta patilla recibe un nivel alto, la UART inicializa todos sus registros, excepto el buffer del receptor, el registro del transmitir y el registro del divisor.

/CTS Contacto 36. Listo para enviar (Clear to Send). Señal CTS.

/DSR Contacto 37. Modem listo (Data Set Ready). Señal DSR.

/DCD Contacto 38. Detección de portadora (Data Carrie Detect). Señal CD.

/RI Contacto 39. Indicador de llamada (Ring Indicator). Señal RI.

Vcc Contacto 40. Alimentación (+5 V).

Como habrá podido observar en el listado anterior, las señales de control de la interfaz RS-232C son transmitidas o procesadas por la UART 8250/16450 en forma invertida, mientras que las señales de datos SIN y SOUT lo hacen en forma normal.

1.9.2 Estructura y funcionamiento

La UART está dividida internamente en dos partes: la sección transmisora y la sección receptora. Cada una de estas partes contiene sus propios registros de datos, registros de control y convertidor serie/paralelo, o viceversa, que le permite trabajar de forma independiente. (Ver fig. 1.9).

Una de las características más importantes de cualquier puerto de comunicaciones es su velocidad de trabajo. La UART está construida de forma que la velocidad de trabajo o, mejor dicho, la frecuencia de referencia de la sección transmisora pueda ser distinta de la frecuencia de referencia de la sección receptora, pudiendo funcionar cada sección a velocidades diferentes.

En el caso de la sección transmisora, la UART maneja dos tipos de frecuencias. La **frecuencia de referencia principal** es una frecuencia generada internamente gracias al oscilador o cristal conectado a las patillas XTAL1 Y XTAL2. En el caso de los ordenadores de tipo PC, esta frecuencia suele ser 1.8432 MHz. La frecuencia de referencia principal también puede ser suministrada externamente.

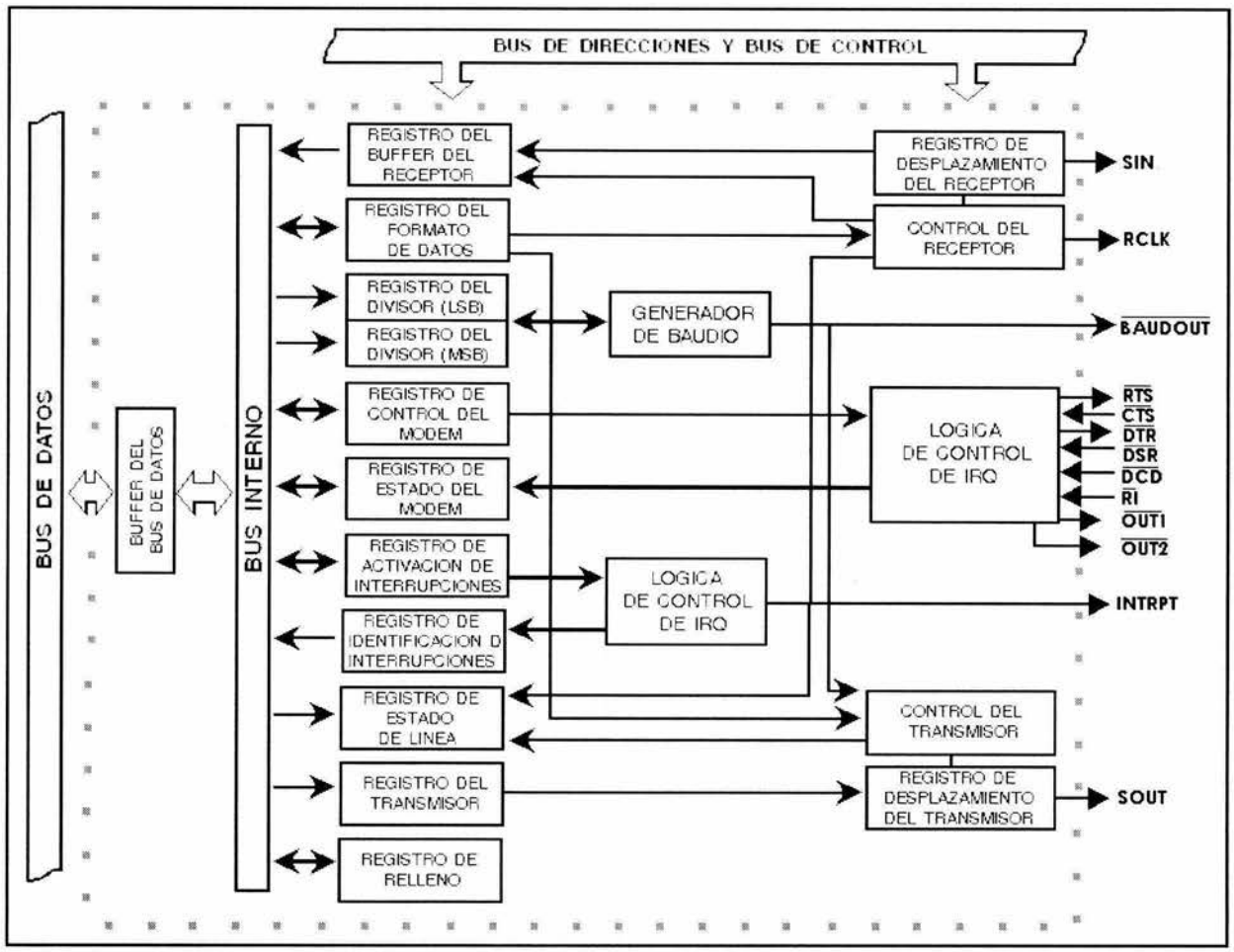


Fig. 1.9 Diagrama de bloques de la UART 8250/16450

La llamada **frecuencia de referencia** (no la frecuencia de referencia principal) es la que define la velocidad de transmisión, y se obtiene por la división de la frecuencia de referencia principal entre el número de 16 bits almacenado en el registro del divisor. Esta frecuencia es la facilitada por la UART en la patilla /BAUDOUT.

Por su parte, la frecuencia utilizada por la sección receptora es generada externamente (generalmente por el módem) y suministrada a la UART por la patilla RCLK. Tanto esta frecuencia como la frecuencia de referencia de la UART se corresponde con un valor 16 veces superior a la velocidad de recepción o transmisión de información en bps. Este hecho es debido a que la UART 8250/16450 siempre muestrea cada bit 16 veces, tomando el valor medio de estas muestras como valor del bit. Este sistema permite una mayor inmunidad frente a las posibles distorsiones de la señal, sobre todo a velocidades elevadas.

Recepción de información

Cuando se recibe información por la patilla SIN, el control del receptor detecta y separa el bit de arranque (start), bit de paridad (si está presente) y el bit de parada (stop) de cada carácter. Por su parte, los bits de datos son convertidos de modo serie a paralelo por el registro de desplazamiento del receptor, guardándose el byte resultante en el registro del buffer del receptor a la espera de ser leído por el software de comunicaciones. El número de bits de datos puede ser de 5, 6, 7 u 8 por cada carácter recibido. En el caso de que cada carácter tenga menos de 8 bits de datos, los bits sobrantes son los más significativos del byte del registro del buffer, y pueden tomar cualquier valor aleatorio. Eso quiere decir que en el caso de realizarse un programa que lea dichos bytes, se tendrá que tener la precaución de eliminar los bits más significativos no válidos.

Transmisión de información

En el lado transmisor, el software de comunicaciones coloca cada carácter que desea transmitir en el registro del transmisor. El control del transmisor copia automáticamente dicho carácter en el registro de desplazamiento del transmisor y le inserta el bit de arranque, bit de paridad y bits de parada correspondiente. La información sale a continuación por la patilla SOUT a la velocidad de transferencia programada. Si no hay datos que transmitir, la salida SOUT se mantiene en el estado 1 (marca).

Control de la comunicación

El control de la comunicación en cuanto a sus relaciones con el exterior se lleva a cabo desde las unidades conocidas corno lógica de control del módem y lógica de control IRQ. La lógica de control del módem suministra y recibe las señales de control de la interfaz RS-232C (/RTS, /CTS, /DTR, /DSR, /DCD Y RI). Por su parte, la lógica de control IRQ detecta los cambios de las líneas de control de la interfaz, del registro del buffer del receptor y del registro del transmisor, así corno los posibles errores de transferencia.

Identificación de los registros

Si contamos el número de registros, teniendo en cuenta que el registro del divisor son realmente dos (LSB y MSB, registros del byte menos significativo y registro del byte más significativo), veremos que hay un total de once. Todos estos registros están conectados entre sí a través del bus interno, y pueden ser leídos y/o escritos con las instrucciones IN y OUT en la dirección correspondiente. En este sentido, los bits de dirección que identifican a cada registro son los A0, Al Y A2, con los cuales sólo se pueden direccionar 8 posiciones distintas ($2^3 = 8$). Si tenemos en cuenta que el registro del buffer del receptor y registro del transmisor comparten la dirección 000h (en condiciones normales, una instrucción IN a la dirección 000h leerá el byte existente en el buffer receptor, y una instrucción OUT a esta misma dirección escribirá un byte en el registro del transmisor), nos quedan 10 registros para 8 direcciones.

Para resolver esta situación, se utiliza el bit más significativo del registro de formato de datos (bit DLAB, ver tabla 1.14). Este bit permite utilizar las direcciones 000h y 00lh para cinco registros distintos. Cuando DLAB está a 1, la dirección 000h se refiere al registro LSB del divisor, y la dirección 00lh se refiere al registro MSB del divisor. Por el contrario, cuando DLAB está a 0, la dirección 000h se refiere al registro del buffer del receptor o registro del transmisor, y la dirección 001 h se refiere al registro de activación de interrupción.

DIRECCIONES DE LOS REGISTROS DE LA UART 8250			
REGISTRO	BIT DLAB	A2 A1A0	OFFSET
Registro del buffer del receptor Registro del transmisor	0	0 0 0	00h
Registro de activación de interrupción	0	0 0 1	01h
Registro de identificación de interrupción	-	0 1 0	02h
Registro de formato de datos	-	0 1 1	03h
Registro de control del módem	-	1 0 0	04h
Registro de estado de señalización	-	1 0 1	05h
Registro de estado del módem	-	1 1 0	06h
Registro de relleno	-	1 1 1	07h
Registro del divisor (LSB)	1	0 0 0	00h
Registro del divisor (MSB)	1	0 0 1	01h

Tabla 1.14 Direcciones de los registros de la UART 8250

1.9.3 Programación de los registros de la UART

Como hemos visto anteriormente, la UART 8250 dispone de 10 registros, entre los de control y los de estado. La UART 16450 dispone de un registro adicional, llamado registro de relleno (scratch register), el cual puede ser utilizado como un byte de memoria adicional para datos temporales, pero que no afecta al funcionamiento de la UART. Todos estos registros disponen de direcciones sucesivas, diferenciándose unas de otras por los tres últimos bits (entre 000 y 111). La dirección del primero de estos registros para cada puerto de comunicaciones, llamada dirección base, está almacenada en el área de datos de la BIOS, y generalmente suelen coincidir con las de la tabla 1.15. Hay que tener en cuenta que algunas BIOS antigua sólo admiten dos puertos serie.

DIRECCIONES BASE Y CANAL IRQ DE LOS PUERTOS SERIE		
PUERTO SERIE	DIRECCION BASE	CANAL IRQ
COM1	3F8h	IRQ4
COM2	2F8h	IRQ3
COM3	3E8h	IRQ4
COM4	2E8h	IRQ3

Tabla 1.15 Direcciones base y canal IRQ de los puertos serie

Registro del buffer del receptor

Al llegar una cadena de datos serie a la entrada SIN de la UART, cada carácter serie es convertido internamente en un byte, el cual es guardado de forma temporal en el registro del buffer del receptor, para desde allí ser leído por el software de comunicaciones. El último bit de datos de cada carácter serie recibido por la entrada SIN se corresponde con el bit menos significativo del registro del buffer del receptor (bit R0, ver tabla 1.16). Para leer ese dato, se utiliza una instrucción IN en la dirección de dicho registro. Hay que tener en cuenta que una instrucción IN siempre lee los 8 bits del registro, pero si la longitud del carácter es de menos de 8 bits, habrá uno o más bits en el registro (de los más significativos) que no se corresponden con bits de datos del carácter recibido. En esos casos, los bits deben ponerse a cero.

Como ejemplo de lo anterior, las siguientes líneas leen el registro del buffer del receptor del puerto COM 1:

```
MOV dx, 3f8h ;      Carga la dirección del buffer del receptor en dx
IN al, dx;          Lee el contenido del buffer y la pone en al.
```

BITS DE LOS REGISTROS DE LA UART 8250/16450								
REGISTRO	7	6	5	4	3	2	1	0
Registro del buffer del receptor	R7	R6	R5	R4	R3	R2	R1	R0
Registro del Transmisor	T7	T6	T5	T4	T3	T2	T1	T0
Registro de activación de interrupción	0	0	0	0	SNP	ERBK	TBE	RxRD
Registro de identificación de interrupción	0	0	0	0	0	ID1	ID0	/PND
Registro del formato de datos	DLAB	BRK	PAR2	PAR1	PAR0	STOP	DBA1	DAB0
Registro del control del modem	0	0	0	LOOP	/OUT2	/OUT1	/RTS	/DTR
Registro de estado de línea	0	TXE	TBE	BREK	FRME	PARE	OVFE	RxRD
Registro de estado del modem	/DCD	/RI	/DSR	/CTS	DDCD	DRI	DDSR	DCTS
Registro de relleno	SPR7	SPR6	SPR5	SPR4	SPR3	SPR2	SPR1	SPR0
Registro del divisor (LSB)								
Registro del divisor (MSB)								

Tabla 1. 16 Bits de los registros de la UART 8250/16450

Registro del transmisor

Para transmitir información, cada carácter de los que forman el mensaje debe ser colocado en el registro del transmisor. Esos caracteres van pasando automáticamente al registro de desplazamiento del transmisor, para desde allí ser transmitidos por la patilla SOUT como una cadena de bits serie una vez que el control del transmisor de la UART les haya insertado los bits de arranque, paridad y parada correspondientes.

Como podemos ver en el siguiente ejemplo para escribir un carácter en el registro del transmisor se utiliza la instrucción OUT dirigida a la dirección del registro. Las líneas siguientes escriben el carácter 'a' en el registro del transmisor del puerto COM 1:

```
MOV dx, 3f8h ;      Carga la dirección del registro transmisor en dx
MOV al, 'a' ;       Carga el byte 'a' en al
OUT dx, al ;        Escribe el carácter 'a' en el registro transmisor
```

Registro de activación de interrupción

Una de las formas de controlar una comunicación es a través del servicio de interrupciones que proporciona el sistema. La UART puede producir una interrupción (típicamente IRQ 3 o IRQ 4) cada vez que ocurre una cualquiera de las siguientes acciones:

- se recibe un nuevo byte
- se está listo para transmitir un nuevo byte
- cambia el estado de alguna línea de control del módem
- ocurre cualquier error

La activación de una interrupción debido a cualquiera de estas acciones puede ser habilitada o deshabilitada selectivamente a través del registro de activación de interrupción. Una vez que ocurre una interrupción, es el propio software de comunicaciones el que debe leer el registro de identificación de interrupciones del puerto serie para ver la causa específica de la interrupción.

Los cuatro bits más significativos del registro de activación de interrupción siempre tienen el valor 0, careciendo de significado, De los cuatro bits restantes, si el bit SINP (ver Tabla 1.17) está en 1, entonces la UART activará su línea INTRPT cada vez que cambia cualquiera de los estados de las líneas de control del módem (señales /CTS, /DSR, /CD o /RI). De esta forma, una señal de llamada (/RI) o una recepción de información puede ser atendida vía interrupción hardware. Por otro lado, si el bit ERBK está en 1, la UART generará una interrupción cada vez que el control del receptor detecta un error de paridad, de trama, de saturación o un corte de la conexión. Si el bit TBE está en 1, la UART generará una interrupción tan pronto como es transferido el byte de dato del registro del transmisor al registro de desplazamiento del transmisor. Eso quiere decir que esta interrupción se genera cada vez que el registro del transmisor está dispuesto para recibir el siguiente byte a transmitir. Por último, si el bit RxRD está en 1, la UART generará una interrupción tan pronto como el registro del buffer del receptor contenga un byte de datos completo; esto es, tan pronto como el buffer del receptor disponga de un nuevo carácter para leer. El registro del buffer del receptor se debe leer antes de que llegue un nuevo carácter.

Las siguientes líneas activan los bits TBE y RxRD del registro de activación de interrupción, con lo que la UART generará una interrupción hardware cada vez que se transmita o reciba un carácter por el puerto COM 1:

```
MOV dx, 3f9h ;      Carga la dirección del registro en dx
IN al, dx ;         Coloca en al el valor del registro
OR al, 03h ;        Coloca a uno los bits TBE y RxRD
OUT dx, al ; Escribe el registro
```

En resumen, los bits del registro de activación de interrupción son los siguientes:

Bits 7 al 4: Sin usar

Bit 3, SINP: Entradas RS-232 (RS-232 input)
1 = Interrupción si se produce un cambio en el estado de señales /CTS, /DSR, /CD o /RI
0 = No se genera interrupción

Bit 2, ERBK: Error o corte de comunicación (error and break)
1 =Interrupción si se da un error de paridad, de trama, de saturación o un corte de la conexión
0 = No se genera interrupción

Bit 1, TBE: Registro del transmisor vacío (transmitter buffer empty)
1 = Interrupción si el registro del transmisor se vacía
0 = No se genera interrupción

Bit 0, RxRD: Dato recibido (received data ready)
1 = interrupción cuando el buffer del receptor tiene un nuevo dato
0 = No se genera interrupción

BITS DEL REGISTRO DE ACTIVACION DE INTERRUPCION							
7	6	5	4	3	2	1	0
0	0	0	0	SINP	ERBK	TBE	RxRD

Tabla 1.17 Bits del registro de activación de interrupción

Registro de identificación de interrupción

Con el registro de identificación de interrupción se puede determinar si hay o no pendiente una interrupción. Cuando hay una interrupción activa, el bit /PND (ver Tabla 1.18) está a 0. En este caso, los bits ID0 e ID1 indican tanto la fuente de la petición de interrupción como la prioridad de dichas interrupciones. Las interrupciones identificadas como 00 se corresponden con la prioridad más baja, mientras que las identificadas como 11 representan la prioridad más alta. Como se puede suponer, cualquier interrupción de prioridad superior es atendida antes que cualquier interrupción de prioridad inferior.

BITS DEL REGISTRO DE IDENTIFICACION DE INTERRUPCION							
7	6	5	4	3	2	1	0
0	0	0	0	0	ID1	ID0	/PND

Tabla 1.18 Bits del registro de identificación de interrupciones

La tabla 1.19 nos muestra los hechos que producen las interrupciones, así como las acciones necesarias para atenderlas. Como puede ver en la tabla, en la mayoría de los casos la interrupción es atendida leyendo el registro correspondiente. En el caso de que la interrupción sea producida porque el registro del transmisor se encuentre vacío, esta interrupción puede ser atendida tanto escribiendo un nuevo byte de dato en él como leyendo el registro de identificación de interrupción. Piense que si no fuese así se tendría una transmisión de datos sin fin.

IDENTIFICACION DE INTERRUPCIONES			
INTERRUPCION PRODUCIDA POR	ID1	D0	INTERRUPCION PENDIENTE DE
Cambio en el estado de las señales /CTS, /DSR, /CD o /RI	0	0	Leer el registro de estado del módem
Registro del transmisor vacío	0	1	Escribir un byte de datos en el registro del transmisor o leer el registro de identificación de interrupción.
Byte de datos en el buffer del receptor	1	0	Leer el byte de datos del buffer del receptor
Error de recepción o corte de la Comunicación	1	1	Leer el registro de estado de línea (serialización)

Tabla 1.19 Identificación de interrupciones

A veces, una misma interrupción es utilizada por más de un puerto serie o por un puerto serie y otro dispositivo. En estos casos, para determinar si realmente ha sido nuestro puerto serie el causante o no de la interrupción, el software de comunicaciones debe leer el registro de identificación de interrupción del puerto correspondiente; si no hay ninguna interrupción activa (bit /PND a 1), entonces es que el responsable de la interrupción ha sido el otro dispositivo. Como es lógico, cuanto más puertos o dispositivos compartan y usen la misma interrupción, el tiempo de respuesta de cada uno de ellos será bastante menor, hasta el punto de que puede ser necesario reducir la velocidad normal de trabajo para evitar que se pierdan datos por saturación.

Por otro lado las UART 8250/16450 no utilizan el bit 3 de este registro, sin embargo, las UART que utilizan FIFO (16550) usan este bit para indicar una interrupción de temporización de carácter. Cuando este bit se pone a 1 (ID0 e ID1 iguales a 10), es que la interrupción se ha producido porque existe al menos un byte en la FIFO del receptor y éste no ha sido modificado durante los últimos cuatro tiempos de caracteres. Para eliminar esta indicación de interrupción, hay que leer el registro del buffer del receptor.

Como ejemplo, la siguiente rutina comprueba si ha ocurrido o no una interrupción en el puerto COM1, y en caso afirmativo, según el tipo de interrupción, salta a una u otra dirección de las guardadas en la tabla tab_int:

```
MOV dx, 3fah  ; Carga la dirección del Reg. Ident. Int. en dx
IN al, dx             ; Lee el registro y guarda su valor en al
TEST al, 0lh          ; Comprueba el bit PND
JNZ noint             ; Salta a noint  si no hay interrupción
MOV bx, tab_int [al]  ; De acuerdo a D0 y D1 carga dirección en bx
JMP bx          ; Salta a la dirección correspondiente
noint:
...
```

En resumen, los bits del registro de identificación de interrupción son los siguientes:

Bits 7 al 3: Sin usar
Bit 2 Y 1, ID1, ID0: Bit de identificación de interrupción (indentify bit)
00 = Cambio en el estado de las señales /CTS, /DSR, /CD o /RI (prioridad 0)
01 = Registro del transmisor vacío (prioridad 1)
10 = Byte de datos en el buffer del receptor (prioridad 2)
11= Error de recepción o corte de la comunicación (prioridad 3)
Bit 0, /PND: Existencia de interrupción pendiente (pending bit)
1 = Ninguna interrupción pendiente
0 = Existe una interrupción pendiente

Registro del formato de datos

El registro del formato de datos define, fundamentalmente, la estructura de cada uno de los caracteres a transmitir y de cada uno de los caracteres recibidos; esto es, define el formato del SDU (Serial Data Unit). Además de eso, este registro dispone de los bits DLAB Y BRK (ver Tabla 1.20) utilizados para gestionar las direcciones de los registros de la UART y para provocar un corte de la comunicación, respectivamente. A este registro también se le conoce como registro de control de línea.

BITS DEL REGISTRO DE FORMATO DE DATOS							
7	6	5	4	3	2	1	0
DLAB	BRK	PAR2	PAR1	PAR0	STOP	DBA1	DAB0

Tabla 1.20. Bits del registro del formato de datos

El bit DLAB determina si las direcciones con los desplazamientos 000h y 00lh identifican a los dos registros del divisor o al buffer receptor/registro transmisor y al registro de activación de interrupciones. Si el bit DLAB está a 0, el desplazamiento 000h leerá el registro del buffer del receptor o escribirá en el registro del transmisor, mientras que el desplazamiento 00lh direccionará el registro de activación de interrupciones. En el caso de que el bit DLAB se encuentre al, estos desplazamientos direccionarán a los registros del divisor.

El bit BRK provoca un corte de la comunicación. Para ello, cuando se coloca este bit a 1, la UART mantiene la salida SOUT en estado 0, produciendo así el corte de la comunicación, ya que cuando el ordenador receptor recibe la señal 0 durante un tiempo determinado (generalmente el tiempo equivalente a dos caracteres completos), entiende que el ordenador transmisor está inactivo y corta la comunicación. Tenga en cuenta que cuando no hay información que transmitir, en una transmisión asíncrona el ordenador transmisor debe mantener constantemente la señal l.

Los tres bits de paridad PAR0, PAR1 y PAR2 sirven para determinar el tipo de paridad utilizado. El bit PAR0 es el llamado bit de activación de paridad, y determina si se va a utilizar o no un bit de paridad en el SDU (PAR0 = 0, no hay bit de paridad; PAR0 = 1, hay bit de paridad). El bit PARl es el llamado bit de selección de paridad, y determina el tipo de paridad a utilizar (PAR1 = 0, paridad impar, PAR1 = 1, paridad par). Por último, el bit PAR2 es el llamado bit de paridad forzada, y cuando está a 1 hace que el bit de paridad siempre tenga el valor inverso de PAR1. Así, si PAR2 tiene el valor 1 y PAR1 tiene el valor 0, el bit de paridad siempre tendrá el valor 1 (marca).

Con el bit STOP se define el número de bits de parada (stop) que va a tener el carácter (SDU). Si STOP tiene el valor 1, el carácter contendrá dos bits de parada, mientras que si STOP tiene el valor 0 el carácter tendrá un solo bit de parada.

Por último, con los bits DAB1 Y DAB0 se define el número de bits de datos por carácter. Los valores posibles son de 5, 6, 7 u 8 bits de datos por carácter dependiendo de que DAB1 y DAB0 tomen los valores 00, 01, 10 o 11 respectivamente.

En resumen, los bits del registro del formato de datos son los siguientes:

Bit 7, DLAB: Bit de acceso al registro del divisor (divisor latch access bit)
1 = Acceso al registro del divisor
0 = Acceso al registro del transmisor/buffer del receptor y al registro de activación de interrupción

Bit 6, BRK: Corte de conexión (break)
1 = Genera condición de corte
0 = No crea condición de corte (normal)

Bits 5, 4 y 3, PAR2, PAR1, PAR0: Tipo de paridad (parity)
000 = No hay bit de paridad
001 = Paridad impar
011 = Paridad par
101 =Marca
111 =Espacio

Bit 2, STOP: Número de bits de parada (stop bits)
1 = 2 bits de parada
0=1 bit de parada

Bits 1 y 0, DAB1, DAB0: Número de bits de datos (data bits)
00=5 bits de datos
01=6 bits de datos
10=7 bits de datos
11 =8 bits de datos

Registro del divisor

El registro del divisor es un registro de 16 bits que contiene el número que va a utilizar el generador de baudios para determinar la velocidad de transmisión. Como se ha visto en apartados anteriores, la UART maneja dos tipos de frecuencias: la frecuencia de referencia principal es una frecuencia patrón suministrada externamente o generada internamente por la UART; por otro lado, la llamada frecuencia de referencia es la que define la velocidad de transmisión, y se obtiene por la división de la frecuencia de referencia principal entre el número de 16 bits almacenado en el registro del divisor. Esta es la frecuencia manejada por la UART para la transmisión y recepción de información, aunque esta frecuencia es 16 veces la velocidad real de comunicación en bps.

Dado que el menor valor que puede contener el registro del divisor es 1, y dado también que la frecuencia de referencia principal de un PC suele ser de 1,8432 MHz, la máxima velocidad de comunicación a la que se puede trabajar con la UART 8250/16450 es de 1.843.200/16=115200 bps. (Ver tabla 1.21).

VELOCIDAD DE COMUNICACIÓN Y REGISTRO DEL DIVISOR		
VELOCIDAD EN BPS	VALOR DEL REGISTRO DEL DIVISOR	USO TIPICO
50	2304 (900h)	Teletipos antiguos.
75	1536 (600h)	Teletipos antiguos.
10	1047 (417h)	Teletipos muy antiguos.
300	384 (180h)	Viejos teletipos y terminales obsoletos.
1200	96 (60h)	Viejos terminales, modems.
2400	48 (30h)	Terminales, modems e impresoras.
4800	24 (18h)	Terminales, modems e impresoras.
9600	12 (0Ch)	Terminales, modems, impresoras y fax.
14400	8 (08h)	Modems de alta velocidad.
19200	6 (06h)	Modems de alta velocidad. Transferencia de datos.
28800	4 (04h)	Modems de alta velocidad. Transferencia de datos.
38400	3 (03h)	Modems de alta velocidad. Transferencia de datos.
57600	2 (02h)	Transferencia de datos.
115200	1 (01h)	Transferencia de datos (velocidad máxima).

Tabla 1.21 Velocidad de comunicación y registro del divisor

Como ejemplo de lo anterior, las siguientes líneas fijan una velocidad de comunicación de 300 bps. Para ello, debemos poner el valor 384 en el registro del divisor (1843200/16*300=384). El valor 384 decimal se corresponde con el valor 0180 hexadecimal, por lo que la operación será la siguiente:

```
MOV dx, 3fbh ;      Carga dirección del reg. de formato de datos en dx
IN al, dx  ;        Lee el registro de formato de datos en al
OR al, 80h ;        pone el bit DLAB a 1
OUT dx, al ;        Escribe el registro de formato de datos
MOV dx, 3f8h  ;     Carga la dirección del reg. del divisor LSB en dx
MOV al, 80h ;       Carga el valor 128 en al
OUT dx, al ;        Escribe el valor 128 en el registro LSB
INC dx ;            Direcciona el registro del divisor MSB
MOV al, 0lh ;       Carga el valor 1 en al (valor absoluto 256)
OUT dx, al ;        Escribe el valor 1 en el registro MSB
MOV dx, 3fbh  ;     Carga dirección del reg. de formato de datos en dx
IN al, dx ;         Lee el registro de formato de datos en al
AND al, 7fh ;       pone el bit DLAB a 0
OUT dx, al  ;       Escribe el registro de formato de datos
```

Registro de control del módem

El registro de control del módem se utiliza para supervisar el estado de las líneas RTS y DTR, así como para generar un bucle de prueba interno de la UART 8250/16450.

Los tres bits más significativos de este registro no son utilizados, manteniendo siempre el valor 0. El bit LOOP permite que se lleve a cabo un bucle de prueba interno de la UART. Al poner este bit a 1, la UART 8250/16450 lleva a cabo las siguientes conexiones internas en la lógica de control del módem: /RTS-/CTS, /DTR-//DSR, /OUTl-/RI y /OUT2-/DCD. Por otro lado, la salida del transmisor SOUT es puesta a 1 (marca), y el registro de desplazamiento del transmisor es conectado directamente al registro de desplazamiento del receptor. Con este bucle activo, se pueden probar las funciones de control del módem, la generación de interrupciones en la UART y la conversión paralelo - serie de los registros de desplazamiento del transmisor y del receptor. Cuando se escribe un dato en el registro del transmisor, éste se pasa al registro de desplazamiento del transmisor, se genera la interrupción correspondiente (TBE) y se pasan los bits con el formato definido en el registro del formato de datos al registro de desplazamiento del receptor. Estos bits son convertidos en un byte y transferidos al registro del buffer del receptor, generándose por último una interrupción RxRD. Una comparación del byte que se escribió en el registro de transmisión con el que ahora se encuentra en e! buffer del receptor nos indica si la lógica del transmisor y del receptor de la UART funcionan correctamente o no.

Los bits OUT2 y OUTl son dos bits de propósito general. El bit OUT2 se utiliza para habilitar o no la señal INTRPT, mientras que el bit OUTl no tiene utilidad. Por su parte, los dos bit /RTS y /DTR controlan los niveles de salida de las señales /RTS y /DTR.

En resumen, los bits del registro de control del módem son los siguientes:

Bits 7, 6 Y 5: Sin usar

Bit 4, LOOP: Bucle de prueba (Loopback check)
1 = Habilitado
0 = Inhabilitado

Bit 3, /OUT2: Salida 2 de propósito general (Output 2). Habilita la señal INTRPT.
1 = Habilitado
0 = Inhabilitado

Bit 2, /OUT1: Salida 1 de propósito general (Output 1)
1 = Habilitado
0 = Inhabilitado

Bit 1, /RTS: Petición de envío (Request to Send)
1 = Habilitado
0 = inhabilitado

Bit 0, /DTR: Terminal de datos preparado (Data Terminal Ready)
1 = Habilitado
0 = inhabilitado

BITS DEL REGISTRO DE CONTROL DEL MODEM							
7	6	5	4	3	2	1	0
0	0	0	LOOP	/OUT2	/OUT1	/RTS	/DTR

Tabla 1.22. Bits del registro de control del módem

Registro de estado de línea

El registro de estado de línea, también llamado registro de estado de serialización, contiene información sobre el estado de las secciones receptora y transmisora de la UART. El bit más significativo no está definido, manteniendo siempre el valor 0. El bit TXE indica que el registro del transmisor y el registro de desplazamiento del transmisor están vacíos. Si TXE es 0, entonces los datos están todavía presentes en el registro del transmisor o en el registro de desplazamiento del transmisor. El bit TBE indica si el registro del transmisor está o no vacío. Si este bit está puesto a uno, el registro del transmisor está vacío. Si el bit BREK es 1, entonces es que el receptor ha detectado un corte en la conexión. Eso ocurre si la entrada SIN detecta el nivel lógico del espacio por un tiempo superior al de una SDU (un carácter).

Los tres bits siguientes, indican los posibles errores de recepción. El bit FRME indica un error de trama, el bit PARE indica un error de paridad y el bit OVRE indica un error de saturación del registro del buffer del receptor. Por último, si el bit RxRD es 1, entonces es que en el buffer del receptor hay disponible un byte de datos.

El registro de estado de línea se puede utilizar junto con el registro de identificación de interrupción para determinar la causa de un error o para determinar el estado de la UART.

En resumen, los bits del registro de estado de la línea son los siguientes:

Bit 7: Sin usar

Bit 6, TXE: Transmisor vacío (transmitter empty)
1 = El registro del transmisor y el registro de desplazamiento del transmisor están vacíos
0 = El registro del transmisor o el de desplazamiento del transmisor contienen todavía información

Bit 5, TBE: Registro del transmisor vacío (transmitter buffer empty)
1 = Registro del transmisor vacío
0 = Hay un byte en el registro del transmisor

Bit 4, BREK: Corte (break)
1 = Detectado 0= Sin corte

Bit 3, FRME: Error de trama (framing error)
1 = Error 0 = Sin error

Bit 2, PARE: Error de paridad (parity error)
1 = Error 0 = Sin error

Bit 1, OVRE: Error de saturación (overrun error)
1 = Error 0 = Sin error

Bit 0, RxRD: Recibido un nuevo dato (Received data ready)
1 = Existe un nuevo dato en el buffer del receptor
0 = No se han recibido nuevos datos

BITS DEL REGISTRO DE ESTADO DE LINEA							
7	6	5	4	3	2	1	0
0	TXE	TBE	BREK	FRME	PARE	OVFE	RxRD

Tabla 1.23. Bits del registro de estado de línea

Como ejemplo, las siguientes líneas comprueban si se ha recibido un nuevo dato, cargando dicho dato en al en caso afirmativo:

```
MOV dx, 3fdh        ; Carga dirección del reg. de estado de línea en dx
IN al, dx           ; Lee el reg. de estado de línea en al
TEST al, 0lh        ; Comprueba RxRD
JZ seguir           ;Salta a seguir si RxRD es 0
MOV dx, 3fdh        ;Carga la dirección del buffer del receptor en dx
IN al, dx           ;Lee el dato del buffer del receptor en al
seguir: ...
```

Registro de estado del módem

El registro de estado del módem se utiliza para determinar el estado de las señales de entrada de RS-232C. Los cuatro bits más significativos muestran información sobre el estado actual, y los cuatro bits menos significativos muestran información sobre los cambios ocurridos desde la última lectura del registro. Cada vez que se realiza una lectura de este registro mediante una instrucción IN, los cuatro bits menos significativos son puestos a cero.

Los bits /DCD, /RI, /DSR y /CTS muestran directamente el estado de estas señales de entrada. Cuando esos bits están a 1, indican que sus señales correspondientes están activas. Dicho de otra forma, desde el punto de vista de la interfaz RS-232C, cuando cualquiera de esos bits está al, indica que la línea correspondiente del interfaz tiene una tensión positiva (entre +3 y + 15 voltios). Eso quiere decir que si se está recibiendo una llamada el bit /RI se pondrá a uno; si se detecta una señal portadora, el bit /DCD estará al; si el módem está listo, IDSR estará al, Y si la UAR T está lista, ICTS estará a 1.

Los bits DDCD, DRI, DDSR y DCTS indican si las señales DCD, RI, DSR y CTS han cambiado desde la última operación de lectura de este registro. Si cualquiera de esos bits está a 1, es que su señal correspondiente ha cambiado.

En resumen, los bits del registro de estado del módem son los siguientes:

Bit 7, /DCD: Detección de portadora (Data Carrier Detect)
1 = Señal DCD activa
0 = Señal DCD inactiva

Bit 6, IRI: Indicador de llamada (Ring Indicator)
1 = Señal RI activa
0= Señal RI inactiva

Bit 5, /DSR: Módem preparado (Data Set Ready)
1 = Señal DSR activa
0 = Señal DSR inactiva

Bit 4, /CTS: Listo para enviar (Clear to Send)
1 = Señal CTS activa
0 = Señal CTS inactiva

Bit 3, DDCD: Cambios en detección de portadora (Delta Data Carrier Detect)
1 = DCD cambió desde la última lectura
0 = DCD no cambió

Bit 2. DRI: Cambios en indicador de llamada (Delta Ring Indicator)
1 = RI cambió desde la última lectura
0 = RI no cambió

Bit 1, DDSR: Cambios en módem preparado (Delta Data Set Ready)
1 = DSR cambió desde la última lectura
0 = DSR no cambió

Bit 0, DCTS: Cambios en listo para enviar (delta clear to send)
1 = CTS cambió desde la última lectura
0 = CTS no cambió

BITS DEL REGISTRO DE ESTADO DEL MODEM							
7	6	5	4	3	2	1	0
/DCD	/RI	/DSR	/CTS	DDCD	DRI	DDSR	DCTS

Tabla 1.24. Bits del registro de estado del módem

1.9.4 UART 16550/82510

La última aportación en la evolución de la unidades controladoras de los puertos serie de comunicaciones son los chips UART 16550 y 82510. La serie 16550 opera, como las UART 8250/16450, pero añade un modo FIFO opcional (First In First Out, primero en entrar, primero en salir). A pesar de ello, dado que la mayoría de los sistemas no usan el chip 16550, es común que aun disponiendo de él no se utilicen sus características. En aquellos casos en los que las UART 8250/16450 residan en zócalos, estos chips pueden ser reemplazados fácilmente por la nueva UART 16550, ya que sus patillas son idénticas.

Por su parte, el chip de Intel 82510 es una evolución del 8250, pero, al igual que el chip 16550, añade el modo FIFO. No obstante, a diferencia del 16550, el 82510 resulta más complejo de programar, ofreciendo a cambio muchos modos y registros adicionales.

Cuando se activa el modo FIFO, la UART tiene acceso a un registro de 16 bytes para la recepción de datos, y a otro registro de 16 bytes para la transmisión. Esta capacidad adicional le permite a la CPU del ordenador disponer de un tiempo adicional para procesar interrupciones o atender a otras funciones de mayor prioridad. Las rutinas de lectura y escritura de un programa de comunicaciones que maneje la UART en modo FIFO son distintas que las que lo manejan en modo no-FIFO. Al igual que en el modo no-FIFO, la programación del modo FIFO puede estar basada en la utilización de interrupciones o en la utilización de técnicas de sondeo. En este último caso, las rutinas que manejan el receptor y el transmisor comprobarán periódicamente el registro de estado de línea, En este registro, el bit 0 es puesto a 1 cuando el FIFO del receptor contiene uno o más bytes, y el bit 5 es mantenido a 1 mientras el FIFO del transmisor contenga uno o más bytes.

En el caso de que se utilicen las interrupciones de la FIFO, la UART producirá una interrupción cuando ocurra una de las siguientes circunstancias:

- El FIFO del transmisor acaba de vaciarse.
- El FIFO del receptor tiene uno o más bytes.
- No se reciben bytes por un tiempo superior al de temporización.
- La CPU no ha leído ningún byte de la UART en un tiempo superior al de temporización.

El periodo de temporización es un tiempo equivalente al de recepción de cuatro caracteres SDU. Eso significa que la temporización es directamente proporcional a la velocidad de transmisión/recepción. Por ejemplo, la temporización para una comunicación de 9600 bps con 10 bits totales por carácter es de 4.2 mseg (4*10/9600).

1.9.5 MÉTODOS DE PROGRAMACIÓN DE LA UART

Para realizar un programa de comunicaciones que controle el intercambio de información a través de la UART, existen dos métodos: el método de sondeo, basado en realizar comprobaciones repetitivas del estado de la UART, y el método de interrupción, basado en aprovechar las características de interrupción de la UART. El método de interrupción es más complicado de programar, pero también consigue mejores resultados. Para aquellas personas que se introduzcan en estos temas, es más conveniente comenzar por el método de sondeo.

Método de sondeo

Para programar la UART siguiendo el método de sondeo, lo primero que debe hacerse es poner el registro de activación de interrupciones a cero. Una vez hecho eso, debe realizarse un bucle que controle tanto la transmisión como la recepción de datos.

En el caso de la transmisión, el bucle debe realizar tantas transmisiones de bytes como caracteres tenga el mensaje a transmitir. Con este bucle, cada vez que se queda vacío el registro del transmisor se comprueba que las señales de control del módem sean las adecuadas y se envía un nuevo carácter. En el caso de la recepción, el proceso es similar, pero en este caso se comprueba en el registro de estado de línea si se ha recibido un nuevo carácter para, en caso afirmativo, leerlo en el registro del buffer del receptor.

Hay que tener en cuenta que en las comunicaciones dúplex, cuando se transmite y recibe al mismo tiempo, el programa de comunicaciones debe comprobar tanto la transmisión como la recepción.

Método de interrupción

Para controlar las comunicaciones de la UART siguiendo el método de interrupción, se tienen que realizar dos programas: uno, llamado principal, que se encarga de controlar todas las interrelaciones con el usuario, y otro, llamado manejador de interrupciones, que se encarga de atender las interrupciones de la UART. Ambos programas funcionan de forma independiente, y se comunican a través de la memoria del ordenador.

Una vez escrito el código del manejador de interrupciones, se debe tocar la tabla de vectores de interrupción para que el puntero correspondiente apunte a ese código. Por otro lado, en el registro de activación de interrupción se debe habilitar la generación de interrupciones.

Cuando ocurra una de las interrupciones, la CPU buscará en la tabla de los vectores de interrupción el puntero del manejador de interrupción, el cual será llamado. Cuando el manejador recibe una interrupción, en un principio no sabe cuál fue la causa exacta de la misma. Para averiguar la causa de la interrupción, el manejador de interrupciones debe examinar el registro de identificación de interrupción. La causa de la interrupción puede ser:

- Se ha recibido un nuevo dato. Indica que hay un nuevo byte en el registro del buffer del receptor. En este caso, se debe leer dicho registro y localizar el byte en una zona de la memoria donde pueda ser leído por el programa principal.
- El registro del transmisor está vacío. Indica que el registro del transmisor se ha quedado vacío, y por tanto el manejador de interrupciones podría colocar en él el próximo byte a transmitir.
- Cualquier otra causa. Si la interrupción fue producida por otra causa distinta el manejador debe analizar los bits delta del registro de estado del módem y el registro de estado de línea para ver lo que ha ocurrido y tomar la acción adecuada.

Modo de operación FIFO

Las UART 16550 disponen de un registro que no está presente en las UART 8250/16450, este es el registro de control de FIFO. Este registro usa la misma dirección que el registro de identificación de interrupción, y es utilizado para controlar las operaciones con los registros FIFO. El hecho de que este registro comparta la dirección con el registro de identificación de interrupción no es problema, porque mientras el registro de control de FIFO es de sólo escritura, el de identificación de interrupciones es de sólo lectura.

Los bits del registro de control de FIFO tienen el siguiente significado:

Bit 0	habilita las FIFO del receptor y del transmisor.
Bit 1	borra todos los bytes del FIFO del receptor.
Bit 2	borra todos los bytes del FIFO del transmisor.
Bit 3	longitud de palabra (acceso DMA al puerto serie).
Bit 4-5	reservado.
Bit 6-7	fija el nivel de llenado del FIFO del receptor.
	00 1 carácter
	01 4 caracteres
	10 8 caracteres
	11 14 caracteres

Con el modo FIFO se puede utilizar tanto el método de sondeo como el método de interrupción vistos anteriormente. El método de sondeo es idéntico al del 8250. Por su parte, el método de interrupción en la UART 16550 en lo único que se diferencia del aplicado en la UART 8250 es que no se produce una interrupción cuando se recibe un carácter, sino cuando se recibe el número de caracteres indicados por el nivel de llenado del registro de control de FIFO (bit 6 Y 7), o bien cuando se reciben menos caracteres pero se ha sobrepasado un tiempo de temporización sin recibir más información equivalente a la recepción de cuatro caracteres SDU.

1.9.6 PROCESO DE COMUNICACIÓN EN MODO SONDEO

Para transmitir y recibir información a través del puerto serie utilizando el método de sondeo, lo primero que se tiene que hacer es configurar el puerto que se va a utilizar, fijando el formato del carácter SDU y la velocidad en bits por segundo. Para hacer eso, debemos hacer lo siguiente:

Se determina el puerto serie a utilizar, así como la velocidad y el formato de carácter adecuado. Según el puerto serie elegido, se determina su dirección base. Esa dirección se encuentra en el área de datos de la BIOS (ver Tabla 1.16). Si el puerto serie en cuestión fuese el puerto 1, su dirección base se encontraría en 40:0h. Si ese valor está a 0, es que dicho puerto no está activo, y por tanto no se podrán enviar datos por él.
Según la velocidad y formato de carácter SDU, se le pasa dicha configuración al registro del formato de datos.

Proceso de transmisión

Una vez fijados los parámetros de la comunicación, para transmitir información debemos llevar a cabo los siguientes pasos:

- ⇨ Lo primero es indicarle al módem que el ordenador está preparado y que tiene la intención de transmitir datos. Para ello, se ponen a nivel alto las líneas de control del módem DTR y RTS. DTR informa al módem que el ordenador está activo y listo para una comunicación. Por su parte, RTS le indica al módem que el ordenador quiere transmitir datos. Para activar esas dos líneas, se debe escribir el valor 3 en el registro de control del módem.

- ⇨ A continuación se comprueba el estado del módem. Para ello, comprobamos en el registro de estado del módem la situación de las líneas CTS y DSR; esto es, el valor de los bits 4 y 5. DSR indica que el módem está encendido y listo para una comunicación: por su lado, CTS indica que el módem está listo para enviar los datos que reciba del ordenador.

- ⇨ Lo último que se debe comprobar antes de transmitir un byte es si el registro del transmisor de la UART está listo para recibir dicho byte; esto es, si el registro del transmisor está vacío. Para ello, el bit 5 del registro de estado de línea debe de estar a 1.

- ⇨ Una vez efectuadas todas las comprobaciones, se transmite el byte. Si el byte se intentó transmitir sin comprobar alguno de los estados anteriores y alguno no fuese el correcto, la UART esperará un tiempo prefijado de temporización a que dicho estado pase a ser correcto, devolviendo un error de temporización en el caso de que eso no ocurra.

- ⇨ Si todo va bien, la UART transfiere dicho byte del registro del transmisor al registro de desplazamiento del transmisor, y desde ahí se envía el carácter SDU ya formado por la línea de salida.

Proceso de recepción

Por su parte, los pasos a seguir para leer un byte recibido en el puerto serie se basa en los pasos que vamos a ver a continuación (se supone que el puerto serie ya ha sido configurado con la velocidad y formato de carácter adecuado):

- ⇨ Se activa la línea DTR de control del módem para indicarle al módem que el ordenador está activo y listo para una comunicación. Para ello, se escribe el valor 1 en el registro de control del módem.

- ⇨ A continuación comprobamos en el registro de estado del módem la situación de la línea DSR. DSR indica que el módem está encendido y listo para una comunicación. Esa información está contenida en el bit 5 de dicho registro.

- ⇨ Una vez que todo está preparado, se espera a que se reciba un dato en el buffer del receptor. El registro de estado de línea, bit 0, indica si se dispone de un nuevo dato en el buffer del receptor. Este es puesto a 1 cuando el buffer del receptor dispone de un nuevo dato.

⇨ Si no ocurre ningún error de temporización, el nuevo byte recibido puede ser leído en el buffer del receptor de la UART.

1.9.7 ERRORES DE LOS CHIPS UART

La primera versión del chip 8250 tenia un error (un bug) que hacia que se produjese una interrupción después de acceder a la interfaz, y sin que aparentemente hubiese razón para ello. Como ese error era conocido, las BIOS de los PC XT lo ignoraban: sin embargo, al salir una nueva versión del chip 8250, conocido como 8250A, se corrigió el error, pero ahora, las BIOS del PC XT, que ya contaban con el error, eran las que producían el fallo. Eso hizo que apareciese una segunda versión del 8250, el 8250B, la cual volvía a introducir el error, aunque esta vez de forma intencionada.

Para que el error del 8250 no sea un gran impedimento para nuestros programas, tenemos dos formas de corregirlo: la primera consiste en utilizar las funciones de la BIOS, la cual ya dispone de los mecanismos de corrección adecuados; la segunda solución consiste en entender claramente las limitaciones de este chip.

Los problemas que introduce el chip 8250 son los siguientes:

- Interrupciones no válidas del registro del transmisor. Cuando en el registro de activación de interrupción se tiene activo el que se produzcan interrupciones al quedar vacío el registro del transmisor, esas interrupciones se producen tanto si este registro está vacío como si no. Para corregir este fallo, el software de comunicaciones debería verificar si el buffer del transmisor está realmente vacío.

- Pérdida del bit de activación de interrupción. La UART puede perder el bit de activación de interrupción del registro del transmisor. Esta pérdida se produce al activar este bit por primera vez. Para resolver el problema, basta con escribir dos veces el registro de activación de interrupción.

```
MOV dx, 3f9h ;        direcc. reg. acto inter.
MOV al, 2;            activa interr. transm. vacío
OUT dx,al ;           escribe el registro
JMP short $+2  ;      retardo
OUT dx, al;           escribe el registro de nuevo
```

- Velocidad de operación limitada. En general, todas las UART podrían ser programadas para velocidades de hasta 115200 bps; sin embargo, las UART 8250, 8250B y 8250C están limitadas por el fabricante a un máximo de 56000 bps. La UART 8250A no tiene esta limitación.

- Problemas con la habilitación de la paridad y con los caracteres de 8 bits de datos.

Por su parte, la UART 16550A trabaja perfectamente cuando lo hace en modo 16450, pero debido a un defecto de diseño, la FIFO de esta UART no opera correctamente. Todas las variantes posteriores incluidas las AF, C y CF tienen este problema corregido.

Lo que sí tienen todas las versiones de la UART 16550 es un fallo particular de la FIFO al transmitir un byte. El problema ocurre si el registro de desplazamiento está transmitiendo un byte en el momento en el que la FIFO está vacía. Si en ese momento se carga un nuevo byte simple en la FIFO del transmisor, la UART debería esperar a que se transmitiera el byte precedente y a continuación debería cargar dicho byte en el registro de desplazamiento para su transmisión. Sin embargo en este caso el nuevo byte no se transfiere al registro de desplazamiento, sencillamente permanece en la FIFO. Cuando se carga el siguiente byte en la FIFO, la UART recupera la operación normal. La única forma de corregir este problema es prevenir la situación. Para ello, basta con comprobar que la FIFO está realmente vacía una vez que se ha transmitido el último byte de un bloque.

CAPITULO 2. PROGRAMACION EN VISUAL BASIC 6.0

CAPITULO 2. PROGRAMACIÓN EN VISUAL BASIC 6.0

DEFINICIÓN

Visual Basic 6.0 es uno de los lenguajes de programación que más entusiasmo despiertan entre los programadores de PCs, tanto expertos como novatos. En el caso de los programadores expertos por la facilidad con la que desarrollan aplicaciones complejas en poquísimo tiempo (comparado con lo que cuesta programar en Visual C++, por ejemplo). En el caso de los programadores novatos por el hecho de ver de lo que son capaces a los pocos minutos de empezar su aprendizaje. El precio que hay que pagar por utilizar Visual Basic 6.0 es una menor velocidad o eficiencia en las aplicaciones.

Visual Basic 6.0 es un lenguaje de programación visual, también llamado lenguaje de 4ª generación. Esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla.

Visual Basic 6.0 es un programa basado en objetos, aunque no orientado a objetos como **C++** o **Java**. La diferencia está en que Visual Basic 6.0 utiliza "objetos" con propiedades y métodos, pero carece de los mecanismos de herencia y polimorfismo de los verdaderos lenguajes orientados a objetos como **Java** y **C++**.

2.1. FORMULARIOS (FORMS) Y CONTROLES

Cada uno de los elementos gráficos que pueden formar parte de una aplicación típica de ***Windows 95/98/NT*** es un tipo de ***control***: los botones, las cajas de diálogo y de texto, las cajas de selección desplegables, los botones de opción y de selección, las barras de desplazamiento horizontales y verticales, los gráficos, los menús, y muchos otros tipos de elementos son controles para ***Visual Basic 6.0***. Cada control debe tener un ***nombre*** a través del cual se puede hacer referencia a él en el programa. ***Visual Basic 6.0*** proporciona nombres *por defecto* que el usuario puede modificar.

En la terminología de ***Visual Basic 6.0*** se llama ***formulario*** (*form*) a una ventana. Un formulario puede ser considerado como una especie de contenedor para los controles. Una aplicación puede tener varios formularios, pero un único formulario puede ser suficiente para las aplicaciones más sencillas. Los formularios deben también tener un nombre, que puede crearse siguiendo las mismas reglas que para los controles.

2.2 FUNCIONES Y PROCEDIMIENTOS

Las aplicaciones informáticas mas utilizadas, incluso a nivel de informática personal, suelen contener decenas y aún miles de líneas de código fuente. A medida que los programas se van desarrollando y aumentan de tamaño, se convertirían rápidamente en sistemas poco manejables si no fuera por la modularización, que es el proceso consistente en dividir un programa muy grande en una serie de módulos mucho más pequeños y manejables. A estos módulos se les suele denominar de distintas formas (subprogramas, subrutinas, procedimientos, funciones, etc.) según los distintos lenguajes. Sea cual sea la nomenclatura, la idea es siempre la misma: dividir un programa grande en un conjunto de subprogramas o **funciones** más pequeñas que son llamadas por el programa principal; éstas a su vez llaman a otras funciones más específicas y así sucesivamente.

La división de un programa en unidades más pequeñas o funciones presenta entre otras las siguientes ventajas:

1. **Modularización**. Cada función tiene una misión muy concreta, de modo que nunca tiene un número de líneas excesivo y siempre se mantiene dentro de un tamaño manejable. Además, una misma función (por ejemplo, un producto de matrices, una resolución de un sistema de ecuaciones lineales, ...) puede ser llamada muchas veces en un mismo programa, e incluso puede ser reutilizada por otros programas. Cada función puede ser desarrollada y comprobada por separado.

2. **Ahorro de memoria y tiempo de desarrollo**. En la medida en que una misma función es utilizada muchas veces, el número total de líneas de código del programa disminuye, y también lo hace la probabilidad de introducir errores en el programa.
3. **Independencia de datos y ocultamiento de información**. Una de las fuentes más comunes de errores en los programas de computador son los *efectos colaterales* o perturbaciones que se pueden producir entre distintas partes del programa. Es muy frecuente que al hacer una modificación para añadir una funcionalidad o corregir un error, se introduzcan nuevos errores en partes del programa que antes funcionaban correctamente. Una función es capaz de mantener una gran independencia con el resto del programa, manteniendo sus propios datos y definiendo muy claramente la *interfaz* o comunicación con la función que la ha llamado y con las funciones a las que llama, y no teniendo ninguna posibilidad de acceso a la información que no le compete.

Funciones (Function) y procedimientos (Sub)

En Visual Basic 6.0 se distingue entre funciones y procedimientos Sub. En ocasiones se utiliza la palabra genérica procedimiento para ambos. La fundamental diferencia entre un procedimiento Sub y una función es que ésta última puede ser utilizada en una expresión porque tiene un valor de retorno. El valor de retorno ocupa el lugar de la llamada a la función donde esta aparece. Por ejemplo, si en una expresión aparece sin(x) se calcula el seno de la variable x y el resultado es el valor de retorno que sustituye a sin(x) en la expresión en la que aparecía. Por tanto, las funciones devuelven valores, a diferencia de los procedimientos que no devuelven ningún valor, y por tanto no pueden ser utilizadas en expresiones. Un procedimiento Sub es un segmento de código independiente del resto, que una vez llamado por el programa, ejecuta un número determinado de instrucciones, sin necesidad de devolver ningún valor al mismo (puede dar resultados modificando los argumentos), mientras que una función siempre tendrá un valor de retorno.

Los nombres de los procedimientos tienen reglas de visibilidad parecidas a las de las variables. Para llamar desde un formulario a un procedimiento Public definido en otro formulario es necesario preceder su nombre por el del formulario en que está definido. Sin embargo, si se desea llamar a un procedimiento definido en un módulo estándar (*.bas) no es necesario precederlo del nombre del módulo más que si hay coincidencia de nombre con otro procedimiento de otro módulo estándar.

Funciones (function)

La sintaxis correspondiente a una función es la siguiente:
[**Static**] [**Private**] **Function** nombre ([parámetros]) [**As** tipo]
[sentencias]
[nombre = expresion]
[**Exit Function**]
[sentencias]
[nombre = expresion]
End Function

donde *nombre* es el nombre de la función. Será de un tipo u otro dependiendo del dato que devuelva. Para especificar el tipo se utiliza la cláusula ***As Tipo*** (***Integer***, ***Long***, ***Single***, ***Double***, ***Currency***, ***String*** o ***Variant***). *parámetros* son los argumentos que son pasados cuando se llama a la función. *Visual Basic* asigna el valor de cada argumento en la llamada al parámetro que ocupa su misma posición. Si no se indica un tipo determinado los argumentos son ***Variant*** por defecto. Como se verá en un apartado posterior, los argumentos pueden ser pasados por referencia o *por valor*.

El nombre de la función, que es el valor de retorno, actúa como una variable dentro del cuerpo de la función. El valor de la variable expresion es almacenado en el propio nombre de la función. Si no se efectúa esta asignación, el resultado devuelto será 0 si la función es numérica, nulo ("") si la función es de caracteres, o **Empty** si la función es **Variant**.

Exit Function permite salir de una función antes de que ésta finalice y devolver así el control del programa a la sentencia inmediatamente a continuación de la que efectuó la llamada a la función.

La sentencia ***End Function*** marca el final del código de la función y, al igual que la ***Exit Function***, devuelve el control del programa a la sentencia siguiente a la que efectuó la llamada, pero lógicamente una vez finalizada la función.

La ***llamada a una función*** se hace de diversas formas. Por ejemplo, una de las más usuales es la siguiente:

```
variable = nombre([argumentos])
```

donde *argumentos* son un lista de constantes, variables o expresiones separadas por comas que son pasadas a la función. En principio, el número de argumentos debe ser igual al número de parámetros de la función. Los *tipos* de los argumentos deben coincidir con los tipos de sus correspondientes parámetros, de lo contrario puede haber fallos importantes en la ejecución del programa. Esta regla no rige si los argumentos se pasan *por valor* (concepto que se verá más adelante).

En cada llamada a una función hay que incluir los paréntesis, aunque ésta no tenga argumentos.

El siguiente ejemplo corresponde a una función que devuelve como resultado la raíz cuadrada de un número *N*:

```
Function Raiz (N As Double) As Double
    If N < 0 Then
    Exit Function
    Else
    Raiz = Sqr(N)
End Function
```

La llamada a esta función se hace de la forma siguiente:

```
Cuadrada = Raiz(Num)
```

A diferencia de C y C++ en ***Visual Basic 6.0*** no es necesario devolver explícitamente el valor de retorno, pues el nombre de la función ya contiene el valor que se desea devolver. Tampoco es necesario declarar las funciones antes de llamarlas.

Procedimientos Sub

La sintaxis que define un *procedimiento Sub* es la siguiente:

```
[Static] [Private] Sub nombre [( parámetros)]
[ sentencias]
[Exit Sub]
[ sentencias]
End Sub
```

La explicación es análoga a la dada para funciones.

La llamada a un *procedimiento Sub* puede ser de alguna de las dos formas siguientes:

```
Call nombre[(argumentos)]
```

o bien, sin pasar los argumentos entre paréntesis, sino poniéndolos a continuación del nombre simplemente separados por comas:

```
nombre [argumentos]
```

A diferencia de una *función*, un *procedimiento Sub* no puede ser utilizado en una expresión pues no devuelve ningún valor. Por supuesto una función puede ser llamada al modo de un *procedimiento Sub*, pero en esta caso no se hace nada con el valor devuelto por la función.

El siguiente ejemplo corresponde a un ***procedimiento Sub*** que devuelve una ariable *F* que es la raíz cuadrada de un número *N*.

```
Sub Raiz (N As Double, F As Double)
If N < 0 Then
Exit Sub 'Se mandaría un mensaje de error
Else
F = Sqr(N)
End If
End Sub
```

La llamada a este ***procedimiento Sub*** puede ser de cualquiera de las dos formas siguientes:

```
Raiz N, F
Call Raiz(N, F)
```

En el ejemplo anterior, el resultado obtenido al extraer la raíz cuadrada al número *N* se devuelve en la variable *F* pasada como argumento, debido a que como se ha mencionado anteriormente, un ***procedimiento Sub*** no puede ser utilizado en una expresión.

2.3 FUNCIONES PARA MANEJO DE CADENAS DE CARACTERES

Existen varias funciones útiles para el manejo de *cadenas de caracteres* (***Strings***). Estas funciones se utilizan para la evaluación, manipulación o conversión de cadenas de caracteres. Algunas de ellas se muestran en la siguiente tabla:

Utilidad	Función en Visual Basic 6.0	Comentarios
Número de caracteres de una cadena	Len(string | varname)	
Conversión a minúsculas o a mayúsculas	LCase(x), UCase(x)	
Conversión de cadenas a números y de números a cadenas	Str(n), CStr(n), Val(string)	
Extracción de un n° de caracteres en un rango, de la parte derecha o izquierda de una cadena	Mid(string, ini[, n]), Right(string, length), Left(string, length)	el parámetro ***n*** de Mid es opcional e indica el número de caracteres a extraer a partir de "***ini***"
Extracción de sub-cadenas	Split(string, [[delim], n])	devuelve un array con las ***n*** (-1 para todas) subcadenas separadas por ***delim*** (por defecto, el espacio)
Unión de sub-cadenas	Join(string, [delim])	
Comparación de cadenas de caracteres	strComp(str1, str2)	devuelve -1, 0, 1 según ***str1*** sea menor, igual o mayor que ***str2***
Hallar si una cadena es parte de otra (está contenida como sub-cadena)	InStr([n], str1, str2)	devuelve la posición de ***str2*** en ***str1*** buscando a partir del carácter ***n***
Hallar una cadena en otra a partir del final (reverse order)	InstrRev(str1, str2, [n])	devuelve la posición de ***str2*** en ***str1*** buscando a partir del carácter ***n***
Buscar y reemplazar una subcadena por otra en una cadena	Replace(string, substring, replacewith)	reemplaza ***substring*** por ***replacewith***

Tabla 2.1 Funciones de manejo de cadenas de caracteres en ***Visual Basic 6.0****.*

Es necesario tener presente que cuando se quieren comparar dos cadenas de caracteres, dicha comparación se realiza por defecto en función del código ASCII asociado a cada letra. Esto significa que por ejemplo *caña* es posterior a *casa* debido a que la letra ***ñ*** tiene un código ASCII asociado superior a la letra ***s*** (***ñ*** es el 164; ***s*** es el 115). Esto mismo ocurre con las vocales acentuadas. Si se desea conseguir una comparación alfabética lógica es necesario incluir al comienzo del fichero de código la sentencia ***Option Compare Text*** (frente a ***Option Compare Binary*** establecida por defecto). La función ***strComp()*** admite un tercer argumento que permite especificar el tipo de comparación (constantes ***vbBinaryCompare*** o ***vbTextCompare***).

Ejemplos:

```
MyDouble = 437.324                ' MyDouble es un Double.
MyString = CStr(MyDouble)         ' MyString contiene "437.324".
MyValue = Val("2457")             ' Devuelve 2457.
MyValue = Val(" 2 45 7")          ' Devuelve 2457.
MyValue = Val("24 and 57")        ' Devuelve 24.

AnyString = "Hello World"         ' Se define el string.
MyStr = Right(AnyString, 6)       ' Devuelve " World".
MyStr = Left(AnyString, 7)        ' Devuelve "Hello W".
MyStr = Right(AnyString, 20)      ' Devuelve "Hello World".

i = StrComp("casa", "caña")       ' Devuelve -1 por defecto y 1 con Option
                                  Compare Text
MyString = "Mid Function Demo"    ' Se crea un nuevo string.
LastWord = Mid(MyString, 14, 4)   ' Devuelve "Demo".
MidWords = Mid(MyString, 5)       ' Devuelve "Function Demo".
```

2.4. FUNCIONES MATEMÁTICAS

Al igual que las funciones vistas para el manejo de cadenas de caracteres, existe una serie de funciones matemáticas las cuales permiten realizar cálculos dentro de un programa de ***Visual Basic***.

Función matemática	Función en Visual Basic	Función matemática	Función en Visual Basic
Valor absoluto	Abs(x)	Nº aleatorio	Rnd
Arco tangente	Atn(x)	Seno y coseno	Sin(x), Cos(x)
Exponencial	Exp(x)	Tangente	Tan(x)
Parte entera	Int(x), Fix(x)	Raiz cuadrada	Sqr(x)
Logaritmo	Log(x)	Signo (1, 0, -1)	Sgn(x)
Redondeo	Round(x, ndec)		

*Tabla 2.2 Funciones matemáticas en **Visual Basic 6.0**.*

Ejemplos:

```
MyNumber = Abs(50.3)              ' Devuelve 50.3.
MyNumber = Abs(-50.3)             ' Devuelve 50.3.
MyAngle = 1.3                     ' El ángulo debe estar en radianes.
MySecant = 1 / Cos(MyAngle)       ' Calcula la secante.
MySqr = Sqr(4)                    ' Devuelve 2.
MySqr = Sqr(23)                   ' Devuelve 4.79583152331272.
MyVar1 = 12: MyVar2 = -2.4: MyVar3 = 0 'Declaración de las variables
MySign = Sgn(MyVar1)              ' Devuelve 1.
MySign = Sgn(MyVar2)              ' Devuelve -1.
MySign = Sgn(MyVar3)              ' Devuelve 0.
```

Las funciones trigonométricas de ***Visual Basic*** utilizan ***radianes*** para medir los ángulos.

Con el fin de completar estas funciones, se ofrece a continuación una relación de funciones que son derivadas de las anteriores. El alumno podría programar dichas funciones en un fichero ****.bas*** y así poderlas utilizar posteriormente en cualquier programa. Dichas funciones se muestran en la tabla 2.3.

Función matemática	Expresión equivalente
Secante	Sec(X) = 1 / Cos(X)
Cosecante	Cosec(X) = 1 / Sin(X)
Cotangente	Cotan(X) = 1 / Tan(X)
Arcoseno	Arcsin(X) = Atn(X / Sqr(-X * X + 1))
Arcocoseno	Arccos(X) = Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
Arcosecante	Arcsec(X) = Atn(X / Sqr(X * X - 1)) + Sgn((X) - 1) * (2 * Atn(1))
Arcocosecante	Arccosec(X) = Atn(X / Sqr(X * X - 1)) + (Sgn(X) - 1) * (2 * Atn(1))
Arcocotangente	Arccotan(X) = Atn(X) + 2 * Atn(1)
Seno Hiperbólico	HSin(X) = (Exp(X) - Exp(-X)) / 2
Coseno Hiperbólico	Hcos(X) = (Exp(X) + Exp(-X)) / 2
Tangente Hiperbólica	Htan(X) = (Exp(X) - Exp(-X)) / (Exp(X) + Exp(-X))
Secante Hiperbólica	HSec(X) = 2 / (Exp(X) + Exp(-X))
Cosecante Hiperbólica	Hcosec(X) = 2 / (Exp(X) - Exp(-X))
Cotangente Hiperbólica	Hcotan(X) = (Exp(X) + Exp(-X)) / (Exp(X) - Exp(-X))
Arcoseno Hiperbólico	Harcsin(X) = Log(X + Sqr(X * X + 1))
Arcocoseno Hiperbólico	Harccos(X) = Log(X + Sqr(X * X - 1))
Arcotangente Hiperbólica	Harctan(X) = Log((1 + X) / (1 - X)) / 2
Arcosecante Hiperbólica	Harcsec(X) = Log((Sqr(-X * X + 1) + 1) / X)
Arcocosecante Hiperbólica	Harccosec(X) = Log((Sgn(X) * Sqr(X * X + 1) + 1) / X)
Arcocotangente Hiperbólica	Harccotan(X) = Log((X + 1) / (X - 1)) / 2
Logaritmo en base N	LogN(X) = Log(X) / Log(N)

*Tabla 2.3 Funciones auxiliares matemáticas (no las tiene **Visual Basic 6.0**).*

2.5 CONTROLES MAS USUALES

En la figura 2.1 se muestran algunos de los controles más habituales en ***Visual Basic 6.0***. Estos controles se explican a continuación con más detalle.

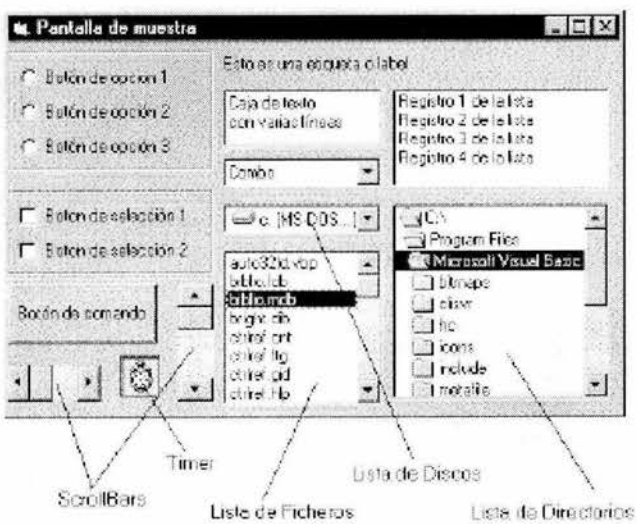
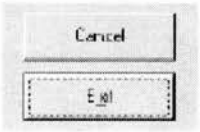


*Figura 2.1 Algunos de los controles más habituales de **Visual Basic**.*

2.5.1 Botón de Comando (Command Button)



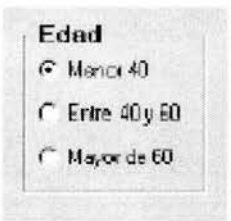
La propiedades más importantes del botón de comando son su ***Caption***, que es lo que aparece escrito en él, las referentes a su posición (***Left*** y ***Top***) y apariencia externa (***Height***, ***Width*** y tipo de letra) y la propiedad ***Enabled***, que determina si en un momento dado puede ser pulsado o no. No hay que confundir la propiedad ***Caption*** con la propiedad ***Name***. La primera define a un texto que aparecerá escrito en el control, mientras que las segunda define el nombre interno con el que se puede hacer referencia al citado objeto.

Si en la propiedad ***Caption*** se pone el carácter (&) antes de una de sus letras, dicha letra aparece subrayyada en el botón (como la "x" en el botón ***Exit*** de la figura anexa). Esto quiere decir que, como es habitual en ***Windows***, dicho botón puede activarse con el teclado por medio de la combinación ***Alt+letra subrayada***. Esta característica es común a muchos de los controles que tienen propiedad ***Caption***.

El evento que siempre suelen tener programado los botones de comandos es el evento ***Click***.

2.5.2 Botones de Opcion (Option Button)

Además de las mencionadas para el caso anterior estos botones tienen la propiedad ***Value***, que en un determinado momento sólo puede ser ***True*** en uno de los botones del grupo ya que se trata de opciones que se excluyen mutuamente.



Para agrupar botones se coloca primero un ***marco*** o ***frame*** en el formulario y, estando seleccionado, se colocan después cuantos botones de opción se desee. En un mismo formulario se pueden colocar cuantos grupos de botones de opción se quiera, cada uno de ellos agrupado dentro de su propio marco. Es muy importante colocar primero el ***frame*** y después los botones de opción. Con esto se consigue que los botones de opción estén agrupados, de modo que sólo uno de ellos pueda estar activado. Si no se coloca ningún ***frame*** todos los botones de opción de un mismo formulario forman un único grupo. Si los botones ya existen y se quieren introducir un un ***frame*** se seleccionan, se hace ***Cut*** y luego ***Paste*** dentro del ***frame*** seleccionado.

Sólo un grupo de botones de opción puede recibir el ***focus***, no cada botón por separado. Cuando el grupo tiene el ***focus***, con las flechas del teclado (- y ¯) se puede activar una u otra opción sin necesidad de usar el ratón. También se puede utilizar ***Alt+carácter*** introduciendo antes de dicho carácter un (&) en el ***Caption*** del botón de opción.

2.5.3 Botones de Selección (Check Box)

La única diferencia entre estos botones y los anteriores es que en los botones de selección puede haber más de uno con la propiedad ***Value*** a ***True***. Estos botones no forman grupo aunque estén dentro de un ***frame***, y reciben el ***focus*** individualmente. Se puede también utilizar el carácter (&) en el ***Caption*** para activarlos con el teclado.

El usuario debe decidir qué tipo de botones se ajustan mejor a sus necesidades: en el caso de la edad, está claro que no se puede ser de dos edades diferentes; sí es posible sin embargo conocer varios lenguajes de programación.

2.5.4 Barras de Desplazamiento (Scroll Bars)

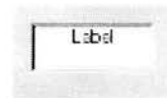
En este tipo de control las propiedades más importantes son ***Max*** y ***Min***, que determinan el rango en el que está incluido su valor, ***LargeChange*** y ***SmallChange*** que determinan lo que se modifica

su valor al clicar en la barra o en el botón con la flecha respectivamente y ***Value*** que determina el valor actual de la barra de desplazamiento. Las barras de desplazamiento no tienen propiedad ***Caption***.

El evento que se programa habitualmente es ***Change,*** que se activa cuando la barra de desplazamiento modifica su valor. Todo lo comentado en este apartado es común para las barras de desplazamientos verticales y horizontales.

Además de las ***Scroll Bars*** horizontal y vertical, ***Visual Basic 6.0*** dispone también del control ***Slider***, utilizado en los paneles de control de ***Windows***, que tiene una función similar.

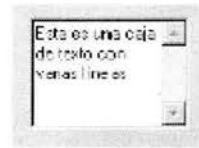
2.5.5 Etiquetas (Labels)



En las etiquetas o labels la propiedad más importante es ***Caption***, que contiene el texto que aparece sobre este control. Esta propiedad puede ser modificada desde programa, pero no interactivamente clicando sobre ella (a diferencia de las ***cajas de texto***, que se verán a continuación). Puede controlarse su tamaño, posición, color de fondo y una especie de borde 3-D. Habitualmente las ***labels*** no suelen recibir eventos ni contener código.

Las ***Labels*** tienen las propiedades ***AutoSize*** y ***WordWrap***. La primera, cuando está a ***True***, ajusta el tamaño del control al del texto en él contenido. La segunda hace que el texto se distribuya en varias líneas cuando no cabe en una sola.

2.5.6 Cajas de Texto (Text Box)



La propiedad más importante de las cajas de texto es ***Text***, que almacena el texto contenido en ellas. También se suelen controlar las que hacen referencia a su tamaño, posición y apariencia. En algún momento se puede desear impedir el acceso a la caja de texto, por lo que se establecerá su propiedad ***Enabled*** como ***False***. La propiedad ***Locked*** como ***True*** hace que la caja de texto sea de sólo lectura. La propiedad ***MultiLine***, que sólo se aplica a las cajas de texto, determina si en una de ellas se pueden incluir más de una línea o si se ignoran los saltos de línea. La justificación o centrado del texto se controla con la propiedad ***Alignment***. La propiedad ***ScrollBars*** permite controlar el que aparezca ninguna, una o las dos barras de desplazamiento de la caja.

En una caja de texto no se pueden introducir ***Intros*** con el teclado en modo de diseño. En modo de ejecución se deben introducir como caracteres ASCII (el 13 seguido del 10, *esto Carriage Return* y *Line Feed*). Afortunadamente ***Visual Basic 6.0*** dispone de la constante ***vbCrLf***, que realiza esta misión de modo automático.

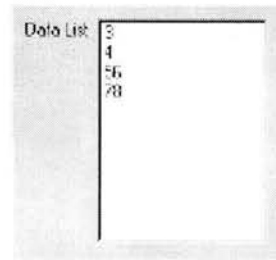
Otras propiedades importantes hacen referencia a la selección de texto dentro de la caja, que sólo están disponibles en tiempo de ejecución. La propiedad ***SelStart*** sirve para posicionar el cursor al comienzo del texto que se desea seleccionar (el primer carácter es el cero); ***SelLength*** indica el número de caracteres o longitud de la selección; ***SelText*** es una cadena de caracteres que representa el texto seleccionado. Para hacer ***Paste*** con otro texto sustituyendo al seleccionado basta asignarle a esta propiedad ese otro texto (si no hay ningún texto seleccionado, el texto de ***SelText*** se inserta en la posición del cursor); para entresacar el texto seleccionado basta utilizar esta propiedad en alguna expresión.

Los eventos que se programan son ***Change***, cuando se quiere realizar alguna acción al modificar el contenido de la caja, ***Click*** y ***DblClick*** y en algunos casos especiales ***KeyPress*** para controlar los caracteres que se introducen. Por ejemplo, se puede chequear la introducción del código ASCII 13 (***Intro***) para detectar que ya se finalizado con la introducción de datos. También se utiliza la propiedad ***MaxLength*** para determinar el número máximo de caracteres que pueden introducirse en la caja de texto.

En aquellos casos en los que se utilice una caja de texto como ***entrada de datos*** (es el control que se utiliza la mayoría de las veces con esta finalidad), puede ser interesante utilizar el método ***SetFocus*** para enviar el foco a la caja cuando se considere oportuno.

Otras propiedades de las cajas de texto hacen referencia a los tipos de letra y al estilo. Así la propiedad ***FontName*** es una cadena que contiene el nombre del *Font* (***Courier New***, ***Times New Roman***, etc.), ***FontSize*** es un tipo ***Short*** que contiene el tamaño de la letra, y ***FontBold***, ***FontItalic***, ***FontUnderline*** y ***FontStrikethrough*** son propiedades tipo ***Boolean*** que indican si el texto va a tener esa característica o no.

2.5.7 Listas (List Box)



Una ***lista*** es un control en el que se pueden mostrar varios registros o líneas, teniendo uno o varios de ellos seleccionado(s). Si en la lista hay más registros de los que se pueden mostrar al mismo tiempo, se añade automáticamente una ***scrollBar***.

Para añadir o eliminar registros de la lista en modo de ejecución se utilizan los métodos ***AddItem*** y ***RemoveItem***. Las listas se suelen inicializar desde el evento ***Form_Load***.

La propiedad ***List*** es un array que permite definir el contenido de la lista en modo de diseño a través de la ventana de propiedades. ***List*** permite también acceder a los elementos de la lista en tiempo de ejecución, para utilizar y/o cambiar su valor. Para ello se pone en índice del elemento entre paréntesis (empezando a contar por cero) a continuación de ***List***, como se muestra a continuación por ejemplo, para cambiar el tercer elemento:

lstName.**List(2)** = "Tercero"
Para añadir un registro en tiempo de ejecución se utiliza ***AddItem***:
lstName.**AddItem** Registro_Añadido, posicion

donde ***posicion*** es un argumento opcional que permite especificar la posición en que se debe añadir. Si se omite el registro se añade al final de la lista. Lo anterior es válido si la propiedad ***Sorted*** está a ***False***; si está a ***True*** el nuevo registro se añade en la posición ordenada que le corresponde. Para eliminar un registro,

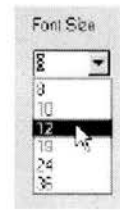
lstName.RemoveItem Posición_del_registro_en_la_lista

En el caso de que se quiera vaciar completamente el contenido de una lista se puede utilizar el método ***Clear***.

Dos propiedades interesantes de las listas son ***ListCount*** y ***ListIndex***. La primera contiene el número total de registros incluidos en la lista. La segunda permite acceder a una posición concreta de la lista para añadir un registro nuevo en esa posición, borrar uno ya existente, seleccionarlo, etc. Hay que recordar una vez más que los elementos de la lista ***se empiezan a numerar por cero***. El valor de propiedad ***ListIndex*** en cada momento coincide con el registro seleccionado y en el caso de no haber ninguno seleccionado esta propiedad vale -1.

Es interesante saber que al seleccionar uno de los registros de la lista se activa el evento ***Clic*** de dicha lista. Las listas permiten seleccionar más de un elemento poniendo la propiedad ***MultiSelect*** a valor ***1-Simple*** o ***2-Extended***. En el primer caso los elementos se seleccionan o se elimina la selección simplemente clicando sobre ellos. En el segundo caso la forma de hacer selecciones múltiples es la típica de ***Windows***, utilizando las teclas ***Ctrl*** y ***Shift***. Con selección múltile la propiedad ***SelCount*** indica el número de elementos seleccionados, mientras que la propiedad ***Selected()*** es un array de valores boolean que indica si cada uno de los elementos de la lista está seleccionado o no.

2.5.8 Cajas Combinadas (Combo Box)



Un ***ComboBox*** tiene muchas cosas en común con una ***lista***. Por ejemplo los métodos ***AddItem***, ***RemoveItem*** o ***Clear*** y las propiedades ***List***, ***ListIndex*** o ***ListCount***.

La diferencia principal es que en un ***ComboBox*** tiene una propiedad llamada ***Style***, que puede adoptar tres valores (***1,2*** ó ***3***) que corresponden con tres distintas formas de presentar una lista:

1. ***Style***=0 ó ***Style=vbComboDropDown*** (***Dropdown Combo***). Éste es el valor más habitual y corresponde con el caso en el que sólo se muestra el registro seleccionado, que es editable por el usuario, permaneciendo el resto oculto hasta que el usuario despliega la lista completa clicando sobre el botón-flecha.
2. ***Style=1*** ó ***Style=vbComboSimple*** (***Simple Combo***). En este caso el registro seleccionado también es editable, y se muestra una lista no desplegable dotada si es necesario de una ***scrollbar***.
3. ***Style=2*** ó ***Style=vbComboDropDownList*** (***DropDown List***). En este último caso el registro seleccionado no es editable y la lista es desplegable.

En una ***caja combinada***, al igual que en una ***caja de texto*** sencilla, está permitido escribir con el teclado en tiempo de ejecución, si la propiedad ***Enabled*** vale ***True***. En una ***lista*** esto no es posible.

La propiedad ***Text*** corresponde con lo que aparece en la parte de caja de texto del control ***ComboBox***, bien sea porque el usuario lo ha introducido, bien porque lo haya seleccionado.

2.5.9 Controles Relacionados con Ficheros

Trabajando en un entorno ***Windows 95/98/NT*** es habitual tener que abrir y cerrar ficheros parar leer datos, guardar un documento, etc. Hay tres controles básicos que resultan de suma utilidad en esta tarea. Son la lista de unidades lógicas o discos (***Drive ListBox***), la lista de directorios (***Dir ListBox***) y la lista de ficheros (***File ListBox***).

2.5.10 Control Timer

Si se desea que una acción suceda con cierta periodicidad se puede utilizar un control ***Timer***. Este control produce de modo automático un evento cada cierto número de milisegundos y es de fundamental importancia para crear ***animaciones*** o aplicaciones con movimiento de objetos. La propiedad más importante de un objeto de este tipo es ***Interval***, que determina, precisamente, el intervalo en milisegundos entre eventos consecutivos. La acción que se desea activar debe programarse en el evento ***Timer*** de ese mismo control.

Si en algún momento se desea detener momentáneamente la acción periódica es suficiente con hacer ***False*** la propiedad ***Enabled*** del control ***Timer*** y para arrancarla de nuevo volver a hacer ***True*** esa propiedad. Haciendo 0 la propiedad ***Interval*** también se consigue inhabilitar el ***Timer***.

2.6 CAJAS DE DIALOGO ESTÁNDAR (CONTROLES COMMON DIALOG)

El control de ***cuadro de diálogo estándar*** de ***Windows 95/NT*** (***Common Dialog***) ofrece una forma sencilla y eficiente de realizar algunas de las tareas más comunes de un programa, tales como la selección de un fichero para lectura/escritura, la impresión de un fichero o la selección de un tipo de letra o un color.

Lo primero que hay que hacer es ubicar el control en el formulario. El control se representará como un icono de tamaño invariable. No es posible especificar la ubicación que tendrá la caja de diálogo cuando se abra en la pantalla, ya que se trata de una propiedad no accesible por el usuario.

Un único ***cuadro de diálogo estándar*** puede bastar para realizar todas las funciones que se deseen, es decir, no es necesario insertar un cuadro de diálogo para imprimir un texto y otro para guardarlo, sino que ambos pueden compartir el mismo cuadro de diálogo simplemente invocando a uno u otro tipo en tiempo de ejecución (no es posible indicarlo en tiempo de diseño). Para ello se dispone de los métodos siguientes***: ShowColor, ShowFont, ShowHelp, ShowOpen, ShowPrinter*** y ***ShowSave.*** En ocasiones interesará introducir varios controles diferentes por motivos de claridad o para que ciertas propiedades sean distintas.

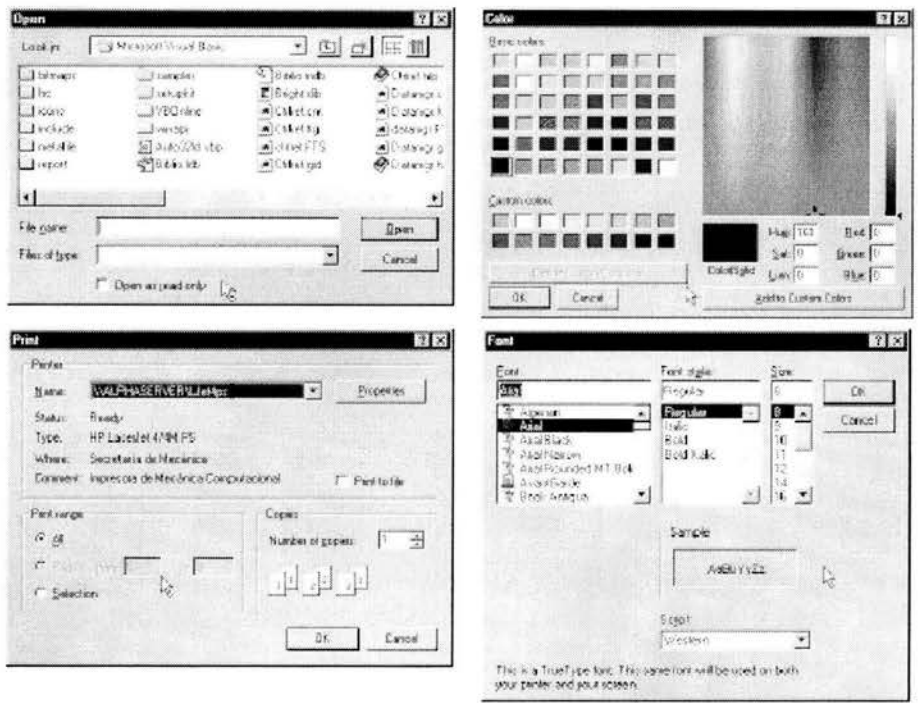


Figura 2.2 Controles Common Dialog.

En la Figura 2.2 se pueden observar distintos tipos de control ***Common Dialog***. Por ejemplo, si se desea visualizar un cuadro de diálogo para abrir un fichero, habrá que escribir:

```
dlgAbrir.ShowOpen
```

Donde ***dlgAbrir*** es el nombre asignado al control ***Common Dialog***.

Las principales ***propiedades*** de este control en cada una de sus variantes se explican en los apartados siguientes. La propiedad ***Flag*** existe para todos los controles y determina algunas de sus características más importantes.

2.6.1 Open/Save Dialog Control

Las propiedades más importantes de este control son:

- ***DefaultExt***: Es la extensión por defecto a utilizar para abrir/salvar archivos. Con ***Save***, si el nombre del fichero se teclea sin extensión, se añade esta extensión por defecto.
- ***DialogTitle***: Devuelve o da valor al título de la caja de diálogo (cadena de caracteres).
- ***FileName***: Nombre completo del archivo a abrir/salvar, incluyendo el *path*.
- ***FileTitle***: Nombre del archivo a abrir/salvar pero sin la ruta de acceso correspondiente.
- ***Filter***: Contiene los filtros de selección que aparecerán indicados en la parte inferior de la pantalla en la lista de tipos de archivo. Pueden indicarse múltiples tipos de archivo, separándolos mediante un barra vertical (***Alt Gr +< I>***). Su sintaxis es la siguiente: Objeto.Filter = "(descripción a aparecer en la listbox)|filtro" Por ejemplo: "Texto (*.txt)|*.txt|Imágenes(*.bmp;*.ico)|*.bmp;*.ico"
- ***FilterIndex***: Indica el índice (con respecto a la lista de tipos) del filtro por defecto. Se empiezan a numerar por "1".

- ***InitDir***: Contiene el nombre del directorio por defecto. Si no se especifica, se utiliza el directorio actual.
- ***Flags***: Esta propiedad puede tomar muchos valores con objeto de fijar los detalles concretos de este control (por ejemplo, abrir un fichero en modo ***read only***, avisar antes de escribir sobre un fichero ya existente, etc.). Estos valores están definidos por constantes de ***Visual Basic 6.0*** cuyos nombres empiezan con las letras ***cdl***. Para más información en el ***Help*** de ***Common Dialog Control*** buscar ***Properties, Flags Properties (Open, Save As Dialogs)***. Por ejemplo, el valor definido por la constante ***cdlOFNOverwritePrompt*** hace que antes de escribir en un fichero ya existente se pida confirmación al usuario. Para establecer varias opciones a la vez se le asigna a ***Flags*** la suma de las constantes correspondientes. Las distintas constantes disponibles se pueden encontrar en el ***Help*** buscando ***Constants/CommonDialog Control***.

2.6.2 Print Dialog Control

Las propiedades más importantes de este control son:

- ***Copies***: Determina el número de copias a realizar por la impresora.
- ***FromPage***: Selecciona el número de página a partir del cual comienza el rango de impresión.
- ***ToPage***: Selecciona el número de página hasta la cual llega el rango de impresión.
- ***PrinterDefault***: Cuando es ***True*** se imprime en el objeto ***Visual Basic Printer***. Además las opciones actuales de impresión que se cambien serán asignadas como las opciones de impresión por defecto del sistema.
- ***Flags***: Ver con ayuda del ***Help*** los posibles valores de esta propiedad.

2.7 MENUS

*Figura 2.3 Editor de menús de **Visual Basic**.*

Entre las capacidades de ***Visual Basic 6.0*** no podía faltar la de construir menús con gran facilidad. Sin embargo, hay algunas diferencias respecto al modo el que se construyen los controles. Para crear menús ***Visual Basic*** dispone de una herramienta especial que se activa mediante el comando ***Menu Editor*** del menú ***Tools***. El cuadro de diálogo que se abre se nuestra en la Figura 2.3 Más adelante se verá cómo se utiliza esta herramienta; antes, conviene recordar brevemente las características más importantes de los menús de ***Windows 95/98/NT***.

Los menús presentan sobre los demás controles la ventaja de que ocupan menos espacio en pantalla, pero tienen el inconveniente de que sus posibilidades no están a la vista más que cuando se despliegan.

La mayor parte de las aplicaciones de ***Windows 95*** utilizan menús. Aunque todo el mundo está familiarizado con sus funciones más básicas, conviene ver algunas posibilidades menos usuales. Se utilizarán para ello unas aplicaciones tan conocidas como ***Word*** y ***Excel***.

La Figura 2.4 recoge el aspecto del menú ***View*** de ***Word 97***, en el que conviene destacar las siguientes características:

1. Lo primero que llama la atención es que los menús aparecen divididos en grupos de opciones separados por líneas horizontales.
2. Algunos items como ***Page Layout*** tienen un icono resaltado a su izquierda. Esto quiere decir que ese ítem es la opción elegida entre los cuatro items de su grupo. En este sentido los menús se parecen a los

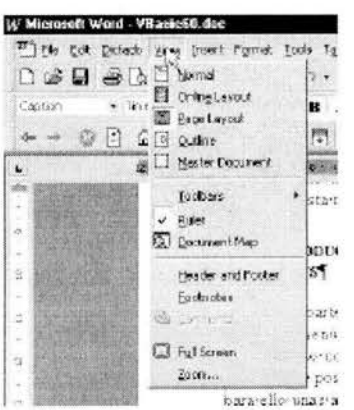


*Figura 2.4 El menú **View** de **Word 97**.*

controles ***OptionButton***. ***Visual Basic 6.0*** no permite hacer esto directamente, pero lo puede simular.

3. Otros items como ***Ruler*** tienen una marca de selección a su izquierda. En este caso el menú realiza la función de las cajas de selección (***CheckBox***).
4. Todas las opciones del menú tienen una letra subrayada. La finalidad es poder desplegar y activar los menús desde teclado, sin ayuda del ratón (con ***Alt*** y la letra subrayada).
5. También se observa que el ítem ***Comments*** aparece en gris claro. Esto quiere decir que en este momento no está activo y por tanto no es seleccionable.
6. Otros items como ***Toolbars*** están seguidos por un pequeño triángulo. Eso quiere decir que existe un menú secundario con más opciones. Otros items como ***Zoom*** aparecen seguidos por puntos suspensivos (...). Este es un convenio utilizado para indicar que eligiendo esa opción se abrirá un cuadro de diálogo en el que habrá que tomar otras decisiones.

Por lo que respecta al menú de ***Excel 97*** que aparece en la Figura 2.5 la característica más importante es que tiene ***sub-menús*** (señalados mediante un pequeño triángulo a su derecha), que se abren al colocar el cursor sobre el ítem correspondiente. Estos menús se suelen llamar ***menús en cascada***, y son muy frecuentes en ***Windows 95/98/NT***.

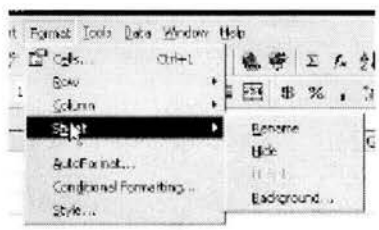


Figura 2.5 El menú ***Format/Sheet*** *de* ***Excel 97***.

Otra característica de los menús, que no aparece en las figuras anteriores, es la posibilidad de definir combinaciones de teclas que realizan la misma función que una opción del menú. Por ejemplo, en muchas aplicaciones ***Ctrl+C*** equivale a ***Edit/Copy*** y ***Ctrl+V*** a ***Edit/Paste***. Estas combinaciones de teclas se llaman *accesos rápidos* (***shortcut***) y hay que distinguirlas de acceder a los menús mediante la tecla ***Alt*** y las letras subrayadas de los nombres.

2.7.1 Editor de Menus

En la Figura 2.6 se vuelve a recoger –a mayor tamaño y con algunos elementos ya definidos- el editor de menús mostrado en la Figura 2.6, que se abre con ***Tools/Menu Editor*** o clicando en el botón correspondiente de la barra de herramientas.

Figura 2.6. Definición de menús con ***Menu Editor***.

Se llama ***título*** a cada elemento que aparece en la barra de menús y ***línea*** o ***ítem*** a cada elemento que aparece al desplegarse un título. Para introducir un nuevo título en la barra de menús hay que definir, en la caja de texto ***Caption*** de la Figura 2.6, el nombre con el que se quiere que aparezca. Si se desea acceder a dicho título mediante teclado (***Alt+letra***), la letra que se desea utilizar deberá ir precedida por el carácter (&). Además,
y al igual que todos los controles de ***Visual Basic 6.0***, conviene que el título tenga un *nombre* (caja de texto ***Name***) para que se pueda acceder a él desde programa. Los nombres de los títulos

de los menús suelen comenzar por las letras ***mnu***, como por ejemplo ***mnuFile***, ***mnuEdit*** o ***mnuHelp***.

En la Figura 2.6 la caja de texto ***Index*** hace referencia a la posibilidad de crear *arrays de menús*. Se puede definir también un ***shortcut*** en la caja de texto correspondiente. En esta figura aparecen cuatro ***checkButtons*** (***Enabled***, ***Checked***, ***Visible*** y ***WindowList***) con los que se pueden especificar algunas propiedades iniciales del menú, como por ejemplo que esté activado o que sea visible.

2.7.2 Adición de código a los menus

Los items de los menús *admiten un único evento*: el evento ***click***, que consiste en ser seleccionados por medio del ratón o del teclado. Para añadir el código correspondiente basta elegir en el menú, estando en modo diseño, el ítem correspondiente para que se abra la ventana de código en el procedimiento ligado a ese evento. También puede buscarse directamente el objeto y el evento correspondiente en las listas desplegables de la ventana de código.

En ocasiones habrá que cambiar las propiedades ***checked***, ***active*** y ***visible*** desde los procedimientos. A estas propiedades se accede del modo habitual, con el nombre del ítem y el operador punto (.)

2.7.3 Arrays de menús

De la misma manera que pueden definirse arrays de controles, también pueden definirse arrays de items (y de títulos) en un menú. La ventaja de definir arrays de items es que *basta definir un único procedimiento* que se haga cargo del evento ***click*** de todos los items del array. Este procedimiento recibe como parámetro la variable entera ***Index***, que indica que ítem del array ha sido seleccionado por el usuario. Dentro de este procedimiento se podrá utilizar por ejemplo la sentencia ***Select Case*** para tratar de forma adecuada cada uno de los casos.

2.8 GRAFICOS EN VISUAL BASIC

Visual Basic 6.0, además de hacer fácil la construcción de interfaces gráficas de usuario, tiene también grandes posibilidades gráficas en lo que se refiere a dibujo de líneas y formas geométricas, así como en lo referente a la introducción de gráficos y figuras realizados con otras aplicaciones. En este capítulo se presentarán brevemente las posibilidades gráficas más importantes de ***Visual Basic 6.0***.

TRATAMIENTO DEL COLOR

Antes de ver cómo se dibuja en ***Visual Basic 6.0*** se verá cómo se definen los colores. Al igual que en tantas aplicaciones informáticas, los colores de ***Visual Basic*** se definen por medio de las componentes fundamentales RGB (***Red***, ***Green*** and ***Blue***). La intensidad de cada color fundamental se define con un ***byte***, es decir con un número entero entre 0 y 255. Se utilizan pues tres bytes para definir los tres colores. ***Visual Basic 6.0*** utiliza un entero *long* (32 bits, 4 bytes) para guardar un color, lo cual quiere decir que existe un byte adicional donde se podrá guardar alguna otra información.

Representación hexadecimal de los colores

Para los números enteros entre 0 y 255 se utilizan dos dígitos hexadecimales. Con esta notación el cero es el "00" y el 255 el "FF". El número que indica el color va precedido por el carácter "&" y la letra "H". Así, el color verde se define en la forma: &H00FF00. Con esta notación es posible prescindir de los ceros situados a la izquierda. Por ejemplo, el color rojo se puede escribir como &H0000FF y como &HFF.

Nombre	Código HEX	Color
vbBlack	&H000000	Negro
vbRed	&H0000FF	Rojo.
vbGreen	&H00FF00	Verde.
vbYellow	&H00FFFF	Amarillo.
vbBlue	&HFF0000	Azul.
vbMagenta	&HFF00FF	Magenta.
vbCyan	&HFFFF00	Cyan.
vbWhite	&HFFFFFF	Blanco.

Visual Basic 6.0 dispone también de nombres para los colores fundamentales y los que son combinación de los colores fundamentales, según puede verse en la Tabla 2.4 .

Tabla 2.4 Nombres de colores.

Acceso a los colores del sistema

El cuarto byte (en el entero *long* que contiene el color) puede utilizarse para hacer referencia a los ***colores del sistema***. Los colores del sistema son aquellos colores con los que ***Windows 95/98/NT*** representa las ventanas y sus bordes, las barras de desplazamiento, etc. Dichos colores se eligen en el panel de control ***Display/Appearance***, y ***Visual Basic 6.0*** permite acceder a ellos a través de su nombre o de su valor hexadecimal, que empieza por "&H8" y utiliza el cuarto byte. La Tabla 2.5 muestra algunos de estos valores. Para una descripción completa buscar ***Color Constants*** en el ***Help*** de ***Visual Basic 6.0***.

Nombre	Valor	Descripción
vbScrollBars	&H80000000	Scroll bar color.
vbDesktop	&H80000001	Desktop color.
vbActiveTitleBar	&H80000002	Color of the title bar for the active window.
vbInactiveTitleBar	&H80000003	Color of the title bar for the inactive window.
vbMenuBar	&H80000004	Menu background color.
vbWindowBackground	&H80000005	Window background color.
vbWindowFrame	&H80000006	Window frame color.
vbMenuText	&H80000007	Color of text on menus.
vbWindowText	&H80000008	Color of text in windows.
vbTitleBarText	&H80000009	Color of text in caption, size box, and scroll arrow.
...	...	...

Tabla 2.5 Colores del sistema.

2.8.1 Formatos Gráficos

En un formulario de ***Visual Basic 6.0*** -y en los controles ***Image*** y ***PictureBox***- es posible insertar gráficos, tanto de tipo *bitmap* (los producidos por aplicaciones como ***Paint***, ***Paintbrush***, ***Paint Shop Pro***, etc.), como de tipo *vectorial* (los producidos por las herramientas gráficas de ***Word*** y ***PowerPoint***).

Visual Basic 6.0 admite varios formatos de ficheros gráficos: los ficheros ****.bmp*** y ****.ico*** para los gráficos de tipo *bitmap*, los ficheros ****.wmf*** (*Windows Meta File*) y ****.emf*** (***Enhanced Meta File***) para los gráficos de tipo vectorial y ****.jpg*** (*JPEG o Joint Photographic Experts Group*) y ****.gif*** (*Graphic Interchange Format*). Los ficheros ****.ico*** son ficheros *bitmap* de pequeño tamaño (32 por 32) destinados a contener iconos. Los ficheros JPEG y GIF son formatos gráficos comprimidos que soportan respectivamente color de 24 bit (~16 millones de colores) y 8 bit (256 colores). Ambos formatos son los utilizados en Internet para mostrar imágenes.

Si se desea insertar ficheros gráficos que estén en otros formatos, habrá que convertirlos previamente a uno de estos formatos usando el programa adecuado.

2.8.2 Controles Gráficos

Visual Basic 6.0 dispone de varios controles con los que insertar gráficos en un formulario. Algunos tienen más posibilidades que otros y es necesario conocerlos bien. A continuación se verán los controles ***Line***, ***Shape***, ***Image*** y ***PictureBox***.

2.8.3 Control Line

Es el control gráfico más elemental, ya que carece de propiedades como *Text*, *Caption* y *Value*. Además no reconoce ningún evento, por lo que su misión es casi exclusivamente decorativa.

El control ***Line*** permite dibujar líneas en un ***formulario*** o en un control ***PictureBox***. Las propiedades más importantes son las coordenadas de los puntos extremos (***X1***, ***Y1***, ***X2*** e ***Y2***), la anchura en pixels (***BorderWidth***), el estilo de la línea (***BorderStyle***) -continua, a trazos, etc.- que sólo está activo cuando la anchura es 1 pixel, el color (***BorderColor***) y el nombre (***Name***). La línea puede estar visible o no (***Visible***), y existe la propiedad ***Index***, que permite crear *arrays* de líneas.

2.8.4 Control Shape

Este control es en muchos aspectos similar al control ***Line***: tampoco tiene las propiedades *text*, *Caption* y *Value*, ni reconoce eventos. Se diferencia en que admite formas geométricas más complejas, que vienen definidas por la propiedad ***Shape***, que admite los valores siguientes: cuadrado (***Square***), rectángulo (***Rectangle***), círculo (***Circle***), elipse (***Oval***), cuadrado redondeado (***Rounded Square***) y rectángulo redondeado (***Rounded Rectangle***).

Además de las propiedades correspondientes al tamaño y posición, las propiedades más interesantes del control ***Shape*** son las siguientes: *BackColor*, *BackStyle*, *BorderColor*, *BorderStyle*, *BorderWidth*, *FillColor*, *FillStyle*, *DrawMode*. Un control ***Shape*** puede estar visible o no (***Visible***), y existe la propiedad ***Index***, que permite crear *arrays* de ***Shapes***.

2.8.5 Métodos Gráficos

Sólo los ***formularios*** y los controles ***PictureBox*** pueden albergar otros tipos de controles. Además es posible escribir texto y dibujar directamente sobre ellos por medio de ciertos ***métodos*** (Los ***métodos*** son procedimientos que ***Visual Basic*** ofrece ya programados. El usuario sólo tiene que llamarlos pasándoles los argumentos apropiados.)de ***Visual Basic***. Por defecto estos métodos actúan sobre el ***formulario activo***. Si se desea que actúen sobre un control ***PictureBox*** hay que precederlos por el nombre del control y el operador punto.

Método Print

En tiempo de ejecución se puede escribir texto en un ***formulario*** o en un control ***PictureBox*** por medio del método ***Print***. La forma general de este método es la siguiente:

objeto.Print {spc(n)|tab(n)} expresion poschar

donde ***spc(n)*** es opcional y sirve para insertar ***n*** caracteres en la salida; ***tab(n)*** es también opcional y sirve para posicionar la salida en una posición absoluta determinada por ***n*** con un tabulador. Si ***tab*** se utiliza sin argumentos lleva al comienzo de la siguiente ***región de salida***; (En ***Visual Basic*** se comienza una ***región de salida*** cada 14 caracteres, si se utiliza un tipo de letra de anchura constante. Con otros tipos de letra esta medida es sólo aproximada.) ***expresion*** representa cualquier expresión cuyo resultado sea un número o una cadena de caracteres. ***poschar*** indica dónde se imprimirá el siguiente carácter. Si es un ***punto y coma*** (;) la impresión se hace a continuación del último carácter impreso; si es un ***tab(n)*** o un ***tab*** tiene el efecto antes descrito; si se omite, la impresión comienza en una nueva línea.

El color, la fuente y el tamaño del texto se toman de las propiedades del formulario o control ***PictureBox***.

Dibujo de puntos: método PSet

El método ***PSet*** sirve para dibujar puntos en un formulario o en un control ***PictureBox***. Su forma general es la siguiente:

```
object.PSet Step (x, y), color
```

donde:

object es opcional y representa el objeto (***form*** o ***PictureBox***) en el que se va a dibujar elpunto. Si se omite, el punto se dibuja en el formulario activo (el que tiene el *focus*).

Step es opcional. Si se introduce las coordenadas que le siguen son relativas respecto a laspropiedades ***CurrentX*** y ***CurrentY*** de la ***PictureBox***. Al dibujar un punto, estas propiedades se actualizan a las coordenadas de dicho punto.

(x, y) son las coordenadas absolutas o relativas del punto a dibujar (expresiones, variables o constantes *single*). Tanto las coordenadas como la coma, como los paréntesis son obligatorios. Las unidades dependen de la propiedad ***ScaleMode*** del objeto en que se dibuja.

Color es opcional y es un nombre de color (***vbRed***, ***vbBlue***, etc.) o un ***long*** conteniendo el código de color hexadecimal (puede ser el valor de retorno de la función ***RGB***). Si se omite, se utiliza la propiedad ***ForeColor*** del objeto en el que se dibuja.

El tamaño del punto viene determinado por la propiedad ***DrawWidth*** del objeto en que se dibuja. Si el tamaño es mayor que uno, el punto se dibuja centrado en las coordenadas suministradas a ***PSet***. Si se desea eliminar un punto previamente dibujado es necesario volver a pintar ese punto con el color de fondo del objeto (***BackColor***).

Dibujo de líneas y rectángulos: método Line

El método ***Line*** dibuja líneas y - en ciertas condiciones- cajas rectangulares de lados horizontales y verticales. Su forma general es la siguiente:

```
object.Line Step (x1, y1) - Step (x2, y2), color, BF
```

donde ***object***, ***step*** y ***color*** tienen el mismo significado que en ***PSet***, y

(x1, y1) son opcionales y son las coordenadas del punto inicial de la línea. Si se omiten la línea comienza en las coordenadas definidas por ***CurrentX*** y ***CurrentY***.

(x2, y2) son obligados y contienen las coordenadas del punto final de la línea.

B es un carácter opcional. Si se incluye se dibuja un rectángulo (*Box*) con los puntos dados como extremos de una de sus diagonales.

F es también un carácter opcional, que sólo se puede incluir si se ha incluido ***B*** Si se incluye, la caja rectangular se rellena (*Fill*) con el mismo color del contorno. Si se omite la caja se rellena con las propiedades ***FillColor*** y ***FillStyle*** del objeto en el que se dibuja.

Después de ejecutarse este método las propiedades ***CurrentX*** y ***CurrentY*** tienen el valor del punto final de la línea. Es necesario introducir el carácter (-), aunque se omita el primero de los puntos que definen la línea.

Las propiedades ***DrawWidth*** y ***DrawStyle*** determinan cómo se dibujan las líneas rectas o curvas en ***Visual Basic 6.0***. Más en concreto, la propiedad ***DrawWidth*** determina el grosor en pixels, mientras que ***DrawStyle*** determina el tipo de línea. La Tabla 2.6 considera los posibles valores de la propiedad ***DrawStyle***.

Valor	Nombre	Estilo de línea
0	vbSolid	continua (valor por defecto)
1	vbDash	trazos (continua si w>1)
2	vbDot	puntos (continua si w>1)
3	vbDashDot	raya-pto (continua si w>1)
4	vbDashDotDot	raya-pto-pto (continua si w>1)
5	vbInvisible	transparente (continua si w>1)
6	vbInsideSolid	continua interna

Tabla 2.6 Valores de ***DrawStyle****.*

Los tipos de raya discontinua no permiten que el grosor sea mayor que 1 pixel. Si el grosor es superior, la línea se dibuja de modo continuo.

2.8.6 Sistema de Coordenadas

Un punto de particular importancia con ***Visual Basic 6.0*** es el que hace referencia a la posición y tamaño de los formularios y de los demás controles, así como a las unidades en que se expresan y determinan.

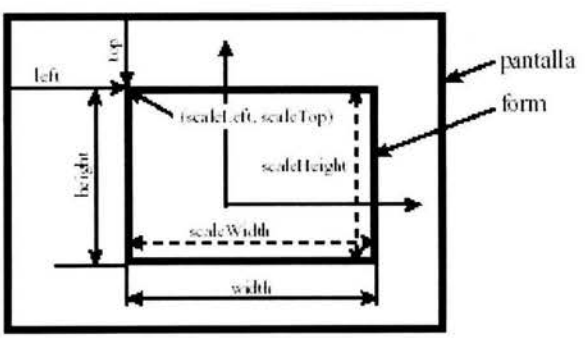
Visual Basic 6.0 permite elegir entre distintas unidades, e incluso utilizar distintas unidades para distintos elementos de la aplicación. Las unidades se especifican con la propiedad ***ScaleMode***, que es propia de los formularios y controles ***PictureBox***.

Valor	Nombre	Unidades
0	vbUser	definidas por el usuario
1	vbTwips	twips (1440 por pulgada)
2	vbPoints	puntos (72 por pulgada)
3	vbPixels	pixels
4	vbCharacters	caracteres (120x240 twips)
5	vbInches	pulgadas
6	vbMillimeters	milímetros
7	vbCentimeters	centímetros

Tabla 2.7 Valores de ScaleMode.

La Tabla 2.7 especifica los posibles valores de esta propiedad. La unidad por defecto es el ***twip***, que es la vigésima parte del ***punto*** o ***pixel***.

Figura2.7. Posición y tamaño de una caja ***PictureBox****.*



En un formulario las propiedades relacionadas con la escala y las dimensiones, agrupadas de cuatro en cuatro, son las siguientes: (***top***, ***left***, ***height*** y ***width***) y (***scaleTop***, ***scaleLeft***, ***scaleHeight*** y ***scaleWidth***). Su significado de explica a continuación con ayuda de la Figura 2.7

En esta figura se muestra la *pantalla* y un *formulario*. La posición y dimensiones del formulario vienen dadas por las propiedades (***top***, ***left***, ***height*** y ***width***). Para un formulario, estas propiedades se definen ***siempre*** en ***twips***. Obsérvese que se miden a partir de la esquina superior izquierda.

Sin embargo, el formulario puede tener su propio sistema de coordenadas interno, definido por las propiedades (***scaleTop***, ***scaleLeft***, ***scaleHeight*** y ***scaleWidth***), para lo cual su propiedad ***ScaleMode*** debe estar puesta a cero.

Las propiedades ***scaleLeft*** y ***scaleTop*** determinan las coordenadas de la esquina superior izquierda en el *propio sistema de coordenadas*, mientras que ***scaleWidth*** y ***scaleHeight*** determinan su anchura y altura en dichas coordenadas. En realidad estas propiedades determinan indirectamente la posición del origen de coordenadas y la escala y orientación de los ejes. Si ***scaleHeight*** es positiva el eje de ordenadas va hacia abajo, mientras que si es negativa estará orientado hacia arriba. El eje horizontal va hacia la derecha si ***scaleWidth*** es positiva, y hacia la izquierda si es negativa.

El método ***Scale*** permite establecer las cuatro propiedades (***scaleTop***, ***scaleLeft***, ***scaleHeight*** y ***scaleWidth***) conjuntamente, como se verá en el siguiente apartado. Sólo los formularios y los controles ***PictureBox*** pueden tener las propiedades ***scaleTop***, ***scaleLeft***, ***scaleHeight*** y ***scaleWidth***.

Si las propiedades (***top***, ***left***, ***height*** y ***width***) no se aplican a un formulario sino a un control, ya no es obligatorio medirlas en ***twips***, sino que se miden en las unidades determinadas por la propiedad ***scaleMode*** del ***formulario*** o ***pictureBox*** que las contiene. Cuando estas propiedades se utilizan sin anteponerles el nombre de un objeto se aplican al formulario activo. Para que se apliquen a un objeto cualquiera basta anteponerles el nombre del objeto separado por el operador punto (.).

2.8.7 Eventos y Propiedades relacionados con gráficos

El evento Saint

El evento ***Paint*** se ejecuta cuando un objeto -de tipo ***form*** o ***PictureBox***- se hace visible. Su finalidad es que el resultado de los ***métodos gráficos*** y del método ***Print*** aparezcan en el objeto correspondiente. Hay que tener en cuenta que si se introducen métodos gráficos en el procedimiento ***Form_load*** su resultado no aparece al hacerse visible el formulario (es como si se dibujara sobre el formulario antes de que éste existiera). Para que el resultados de ***Print*** y de los métodos gráficos aparezcan al hacerse visible el formulario, deben introducirse en el procedimiento ***Paint_form***. También los controles ***pictureBox*** tienen evento ***Paint***, que se ejecuta al hacerse visibles.

El evento ***Paint*** tiene mucha importancia en relación con el refresco de los gráficos y con la velocidad de ejecución de los mismos. En los apartados siguientes se completará la explicación de este tema.

La propiedad DrawMode

Esta es una propiedad bastante importante y difícil de manejar, sobre todo si se quieren realizar cierto tipo de acciones con los métodos gráficos. La opción por defecto es la nº 13: ***Copy Pen***.

La propiedad ***DrawMode*** controla cómo se dibujan los controles ***Line*** y ***Shape***, así como los resultados de los métodos gráficos ***PSet***, ***Line*** y ***Circle***. La opción por defecto hace que cada elemento gráfico se dibuje con el color correspondiente (por defecto el ***foreColor***) sobre lo dibujado anteriormente. En ocasiones esto no es lo más adecuado pues, por ejemplo, si se superponen dos figuras del mismo color o si se dibuja con el ***backColor***, los gráficos resultan indistinguibles.

Para entender cómo funciona ***DrawMode*** es necesario tener claros los conceptos de ***color complementario*** y ***combinación de dos colores***. El color complementario de un color es el color que sumado con él da el ***blanco*** (&HFFFFFF&). Por ejemplo, el color complementario del ***rojo*** (&H0000FF&) es el ***cyan*** (&HFFFF00&).

El ***color complementario*** se puede obtener mediante la simple resta del color blanco menos el color original. Por su parte la ***combinación de dos colores*** es el color que resulta de aplicar el operador lógico ***Or***: el color resultante tiene sus *bits* a 1 si alguno o los dos de los colores originales tiene a 1 el *bit* correspondiente. La explicación de los distintos valores de la propiedad ***DrawMode*** que se obtiene del ***Help*** es la siguiente:

Constant	Setting	Description
VbBlackness	1	Blackness.
VbNotMergePen	2	Not Merge Pen — Inverse of setting 15 (Merge Pen).
VbMaskNotPen	3	Mask Not Pen — Combination of the colors common to the background color and the inverse of the pen.
VbNotCopyPen	4	Not Copy Pen — Inverse of setting 13 (Copy Pen).
VbMaskPenNot	5	Mask Pen Not — Combination of the colors common to both the pen and the inverse of the display.
VbInvert	6	Invert — Inverse of the display color.
VbXorPen	7	Xor Pen — Combination of the colors in the pen and in the display color, but not in both.
VbNotMaskPen	8	Not Mask Pen — Inverse of setting 9 (Mask Pen).
VbMaskPen	9	Mask Pen — Combination of the colors common to both the pen and the display.
VbNotXorPen	10	Not Xor Pen — Inverse of setting 7 (Xor Pen).
VbNop	11	Nop — No operation — output remains unchanged. In effect, this setting turns drawing off.
VbMergeNotPen	12	Merge Not Pen — Combination of the display color and the inverse of the pen color.
VbCopyPen	13	Copy Pen (Default) — Color specified by the ForeColor property.
VbMergePenNot	14	Merge Pen Not — Combination of the pen color and the inverse of the display color.
VbMergePen	15	Merge Pen — Combination of the pen color and the display color.
VbWhiteness	16	Whiteness.

El explicar más a fondo las distintas aplicaciones de esta propiedad esta fuera del alcance de este manual introductorio.

Planos de dibujo (Layers)

Visual Basic 6.0 considera ***tres planos superpuestos*** (layers): el ***plano frontal***, el ***plano intermedio*** y el ***plano de fondo***. Es importante saber en qué plano se introduce cada elemento gráfico para entender cuándo unos elementos se superpondrán a otros en la pantalla. En principio, los tres planos se utilizan del siguiente modo:

1. En el ***plano frontal*** (***Front***) se dibujan *todos los controles*, excepto los controles gráficos y las labels.
2. En el ***plano intermedio*** se representan los *controles gráficos* y *labels*.
3. En el ***plano de fondo*** se representa el *color de fondo* y el resultado de los *métodos gráficos*.

Estas reglas tienen excepciones que dependen de la propiedad ***AutoRedraw***, de la propiedad ***ClipControl*** y de si los métodos gráficos se utilizan o no asociados al evento ***Paint***.

La propiedad AutoRedraw

Esta propiedad tiene una gran importancia. En principio, todas las aplicaciones de ***Windows*** permiten superponer ventanas y/u otros elementos gráficos, recuperando completamente el contenido de cualquier ventana cuando ésta se selecciona de nuevo y viene a primer plano (es la ventana activa). A esto se llama ***redibujar*** (*redraw*) la ventana. Cualquier aplicación que se desarrolle en ***Visual Basic 6.0*** debe ser capaz de redibujarse correctamente, pero para ello el programador debe conocer algo de la propiedad ***AutoRedraw***.

Por defecto, ***Visual Basic 6.0*** redibuja siempre los controles que aparecen en un formulario. Esto no sucede sin embargo con el resultado de los ***métodos gráficos*** y de ***Print***. Para que la salida de estos métodos se redibuje es necesario adoptar uno de los dos métodos siguientes:

Si en el ***form*** o ***pictureBox*** la propiedad ***AutoRedraw*** está en ***False***:

- Si los ***métodos gráficos*** y ***Print*** están en el procedimiento correspondiente al evento ***Paint*** se redibujan en el *plano de fondo* (los métodos vuelven a ejecutarse, por lo que el proceso puede ser lento en ciertos casos).
- Si los ***métodos gráficos*** y ***Print*** están fuera del evento ***Paint*** no se redibujan.

1. Si en el ***form*** o ***pictureBox*** la propiedad ***AutoRedraw*** está en ***True***:
   - Si los ***métodos gráficos*** y ***Print*** están en el evento ***Paint*** se ignoran.

- Si los ***métodos gráficos*** y ***Print*** están fuera del evento ***Paint*** se redibujan guardando en memoria una copia de la zona de pantalla a refrescar. Este es la forma más rápida de conseguir que los gráficos y el texto se redibujen. Tiene el inconveniente de necesitar más memoria.

La propiedad ***AutoRedraw*** de los ***forms*** y de las ***pictureBox*** es independiente, por lo que las dos formas anteriores de conseguir que los gráficos se redibujen se pueden utilizar conjuntamente, por ejemplo una en el formulario y otra en las ***pictureBox***.

La propiedad ClipControl

Por defecto esta propiedad de las ***forms*** y ***pixtureBox*** está en ***True***. En este caso los controles están siempre por encima de la salida de los métodos gráficos, por lo que nunca por ejemplo una línea se dibujará sobre un botón o una barra de desplazamiento (los controles están siempre en el plano frontal o en el plano intermedio, según se ha explicado antes).

Cuando la propiedad ***ClipControl*** se pone a ***False*** se produce una doble circunstancia:

- Los métodos gráficos situados en un evento ***Paint*** siempre se dibujan en el *plano de fondo* y por tanto respetan los controles.
- Los métodos gráficos situados fuera de un evento ***Paint*** se dibujan sobre cualquier elemento que esté en la pantalla, incluidos los controles.

2.9. ARCHIVOS Y ENTRADA/SALIDA DE DATOS

En este inciso se van a describir varias formas de introducir información en el programa, así como de obtener resultados en forma impresa o mediante escritura en un fichero. Se va a presentar una nueva forma interactiva de comunicarse con el usuario, como son las cajas de diálogo ***MsgBox*** e ***InputBox***. Particular interés tiene la lectura y escritura de datos en el disco, lo cual es necesario tanto cuando el volumen de información es muy importante (la memoria RAM está siempre más limitada que el espacio en disco), como cuando se desea que los datos no desaparezcan al terminar la ejecución del programa. Los ficheros en disco resuelven ambos problemas.

También se verá en este capítulo cómo obtener resultados alfanuméricos y gráficos por la impresora.

2.9.1 Cajas de diálogo InputBox y MsgBox

Estas cajas de diálogo son similares a las que se utilizan en muchas aplicaciones de ***Windows***. La caja de mensajes o ***MsgBox*** abre una ventana a través de la cual se envía un mensaje al usuario y se le pide una respuesta, por ejemplo en forma de clicar un botón ***O.K./Cancel***, o ***Yes/No***. Este tipo de mensajes son muy utilizados para confirmar acciones y para decisiones sencillas. La caja de diálogo ***InputBox*** pide al usuario que teclee una frase, por ejemplo su nombre, un título, etc.

La forma general de la función ***MsgBox*** es la siguiente:

```
respuesta = MsgBox("texto para el usuario", tiposBotones, "titulo")
```

donde ***respuesta*** es la variable donde se almacena el valor de retorno, que es un número indicativo del botón clicado por el usuario, de acuerdo con los valores de la Tabla 2.8. La constante simbólica que representa el valor de retorno indica claramente el botón clicado. Los otros dos argumentos son opcionales. El parámetro ***tiposBotones*** es un entero que indica la combinación de botones deseada por el usuario; sus posibles valores se muestran en la Tabla 2.9. También en este caso la constante simbólica correspondiente es suficientemente explícita. Si este argumento se omite se muestra sólo el botón ***O.K.*** El parámetro ***titulo*** contiene un texto que aparece como título de la ventana; si se omite, se muestra en su lugar el nombre de la aplicación.

Valor de retorno	Constante simbólica
1	vbOK
2	vbCancel
3	vbAbort
4	vbRetry
5	vbIgnore
6	vbYes
7	vbNo

Tabla 2.8 Botón clicado por el usuario.

Valor tiposBotones	Constante simbólica
0	vbOKOnly
1	vbOKCancel
2	vbAbortRetryIgnore
3	vbYesNoCancel
4	vbYesNo
5	vbRetryCancel

*Tabla 2.9 Botones mostrados en **MsgBox**.*

Se puede modificar el valor de ***tiposBotones*** de modo que el botón que se activa por defecto cuando se pulsa la tecla ***Intro*** (el botón que tiene el *focus*) sea cualquiera de los botones de la caja. Para ello basta sumar a ***tiposBotones*** otra constante que puede tomar uno de los tres valores siguientes: ***0*** (*vbDefaulButton1*, que representa el primer botón), ***256*** (*vbDefaulButton2*, que representa el segundo botón) y ***512*** (*vbDefaulButton3*, que representa el tercer botón).

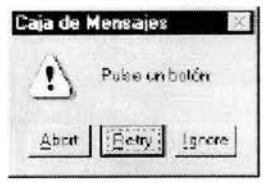


*Figura 2.8 Ejemplo de caja **MsgBox**.*

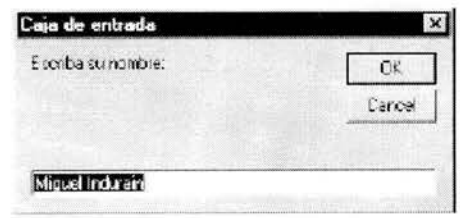


*Figura 2.9 Ejemplo de caja de **InputBox**.*

Finalmente, se puede incluir en el mensaje un *icono ad-hoc* por el mismo procedimiento de sumarle al argumento ***tiposBotones*** una nueva constante numérica con los siguientes valores y significados definidos por la constante simbólica apropiada: 16 (*vbCritical*), 32 (*vbQuestion*), 48 (*vbExclamation*) y 64 (*vbInformation*). Es obvio que, por los propios valores considerados, al sumar estas constantes o las anteriores al argumento ***tiposBotones***, la información original descrita en la Tabla 2.9 no se pierde. La Figura 2.8 muestra un ejemplo de caja ***MsgBox*** resultado de ejecutar el comando siguiente:

```
lblBox.Caption = MsgBox("Pulse un botón: ", 2 + 256 + 48, _
"Caja de mensajes")
```

donde el "2" indica que deben aparecer los botones ***Abort***, ***Retry*** y ***Cancel***, el "256" indica que el *botón por defecto* es el segundo (***Retry***) y el "48" indica que debe aparecer el *icono de exclamación*.

Por otra parte, la forma general de la función ***InputBox*** es la siguiente:

```
texto = InputBox("texto para el usuario", "titulo", "default", left, top)
```

Donde ***texto*** es la variable donde se almacena el valor de retorno, que es el texto tecleado por el usuario. Los parámetros "*texto para el usuario*" y ***titulo*** tienen el mismo significado que en ***MsgBox***. El parámetro ***default*** es un texto por defecto que aparece en la caja de texto y que el usuario puede aceptar, modificar o sustituir; el contenido de esta caja es lo que en definitiva esta función devuelve como valor de retorno. Finalmente, ***left*** y ***top*** son las coordenadas de la esquina superior izquierda de la ***InputBox***; si se omiten, ***Visual Basic 6.0*** dibuja esta caja centrada en horizontal y algo por encima de la mitad de la pantalla en vertical. La Figura 2.9 muestra un ejemplo de caja ***InputBox*** resultado de ejecutar el comando siguiente:

```
lblBox.Caption = InputBox("Escriba su nombre: ", _
"Caja de entrada", "Miguel Indurain")
```

Donde el nombre que aparece por defecto es el del mejor ciclista de los últimos tiempos. Este nombre aparece seleccionado y puede ser sustituido por otro que teclee el usuario.

2.9.2 Método Print

Este método permite escribir texto en ***formularios***, cajas ***pictureBox*** y en un objeto llamado ***Printer***

1. El método ***Print*** recibe como datos una *lista de variables y/o cadenas de caracteres*. Las cadenas son impresas y las variables se sustituyen por su valor.
2. Hay dos tipos básicos de ***separadores*** para los elementos de la lista. El carácter ***punto y coma*** (;) hace que se escriba inmediatamente a continuación de lo anterior. La ***coma*** (,) hace que se vaya al comienzo de la siguiente ***área de salida***. Con letra de paso constante como la Courier las áreas de salida empiezan cada 14 caracteres, es decir en las columnas 1, 15, 29, etc. Con letras de paso variable esto se hace sólo de modo aproximado.
3. Las constantes numéricas positivas van precedidas por un espacio en blanco y separadas entre sí por otro espacio en blanco. Si son negativas el segundo espacio es ocupado por el signo menos (-).
4. El tipo y tamaño de letra que se utiliza depende de la propiedad ***Font*** del formulario, objeto ***PictureBox*** u objeto ***Printer*** en que se esté escribiendo.

Existen otros separadores tales como ***Tab(n)*** y ***Spc(n)***. El primero de ellos lleva el punto de inserción de texto a la columna ***n***, mientras que el segundo deja ***n*** espacios en blanco antes de seguir escribiendo. ***Tab*** sin argumento equivale a la coma (,). Estos espaciadores se utilizan en combinación con el punto y coma (;), para separarlos de los demás argumentos.

Por defecto, la salida de cada método ***Print*** se escribe en una nueva línea, pero si se coloca un punto y coma al final de un método ***Print***, el resultado del siguiente ***Print*** se escribe en la misma línea.

Puede controlarse el lugar del formulario o control donde se imprime la salida del método ***Print***. Esta salida se imprime en el lugar indicado por las propiedades ***CurrentX*** y ***CurrentY*** del formulario o control donde se imprime. Cambiando estas propiedades se modifica el lugar de impresión, que por defecto es la esquina superior izquierda. Existen unas funciones llamadas ***TextWidth(string)*** y ***TextHeight(string)*** que devuelven la anchura y la altura de una cadena de caracteres pasada como argumento. Estas funciones pueden ayudar a calcular los valores más adecuados para las propiedades ***CurrentX*** y ***CurrentY***.

La función ***str(valor_numérico)*** convierte un número en cadena de caracteres para facilitar su impresión. En realidad, es lo que ***Visual Basic 6.0*** ha hecho de modo implícito en los ejemplos anteriores. En versiones anteriores del programa era necesario que el usuario realizase la conversión de modo explícito.

CAPITULO 3.

DISEÑO DEL SISTEMA DE ADQUISICION DE DATOS PARA EL MONITOREO DE LA ESTACION METEOROLOGICA

CAPITULO 3. DISEÑO DEL SISTEMA DE ADQUISICION DE DATOS

3.1. CARACTERÍSTICAS DE UNA ESTACIÓN METEOROLÓGICA

¿Qué es la meteorología?

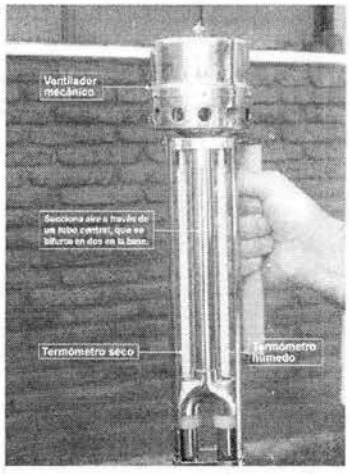
La meteorología es la rama de la física que aborda los fenómenos que ocurren en la atmósfera. Estos se refieren a una gran variedad de procesos, incluyendo entre otros aspectos el movimiento de la atmósfera (meteorología dinámica), su interacción con los flujos de energía radiactiva (radiación solar e infrarroja), los procesos termodinámicos que llevan a la formación de las nubes y la generación de la precipitación en cualquiera de sus formas (lluvia, nieve y granizo), los intercambios de energía con la superficie (transportes de calor y vapor de agua), las reacciones químicas (formación de la capa de ozono, generación de contaminantes por reacciones fotoquímicas), los fenómenos eléctricos (rayos) y los efectos ópticos (arco iris, espejismos, halos en el Sol y la Luna).

Los fenómenos físicos en la atmósfera ocurren en todas las escalas espaciales y temporales y sus impactos son relevantes para muchas actividades.
Están por una parte los fenómenos de escala espacial muy pequeña, como por ejemplo el intercambio de vapor de agua entre las plantas y la atmósfera que ocurre a nivel de los estomas de las hojas. Por otra parte, la evaluación de riesgo de heladas o de disponibilidad de energía eólica requiere del conocimiento de fenómenos que presentan una variabilidad espacial de cientos de metros o de algunos kilómetros. Los procesos que condicionan la dispersión de contaminantes involucran escalas espaciales del tamaño de una región, al igual que el desarrollo de sistemas de brisas costeras o de valle. En la escala de algunos miles de kilómetros se desarrollan sistemas organizados de nubosidad y precipitación asociado a los frentes fríos y cálidos, en tanto que las condiciones meteorológicas anómalas asociadas a los fenómenos El Niño y La Niña tienen que ver con perturbaciones en el comportamiento de la atmósfera en una escala hemisférica.
Desde el punto de vista de la variabilidad temporal de los fenómenos atmosféricos, los meteorólogos analizan una gran variedad de situaciones, aparte de aquellas forzadas por los ciclos astronómicos diario y anual. Están por una parte los fenómenos de muy corta duración como por ejemplo los procesos turbulentos de pequeña escala que explican el transporte de calor en los primeros cientos de metros sobre la superficie, la formación de torbellinos de diversos tamaños o la ocurrencia de rayos. A una escala de minutos a horas ocurren fenómenos tales como la formación de las nubes o el desarrollo de tormentas severas. En la escala de los días se observa el desarrollo de frentes y en general de fenómenos que permiten caracterizar las condiciones de "tiempo" atmosférico en un cierto lugar. En una escala de tiempo todavía mayor está la variabilidad atmosférica intraestacional, que explica por ejemplo la ocurrencia de un periodo relativamente prolongado de buen tiempo en un invierno anormalmente riguroso, y la variabilidad interanual, que da cuenta de los cambios de las condiciones medias meteorológicas de un año al siguiente.
La CLIMATOLOGIA es la rama de la meteorología que se preocupa de estudiar la evolución de las condiciones medias de la atmósfera en periodos relativamente largos, incluyendo cambios que ocurren en periodos de décadas (variabilidad decadal) o de siglos (variabilidad secular).
Tal como se describe, la meteorología se preocupa de una gran diversidad de problemas, además del pronóstico del tiempo, que es la tarea que más típicamente se asocia a esta disciplina.

3.2 SENSORES

Humedad atmosférica (higrometría)

Existen diversas formas para medir el contenido de vapor de agua en la atmósfera. La medición más frecuente es la de humedad relativa, que corresponde a la fracción porcentual entre la presión parcial del vapor de agua y la presión de vapor de agua en el punto de saturación a la temperatura ambiente.



PSICROMETRO

El psicrómetro está formado por dos termómetros. El bulbo de uno de ellos está envuelto en un tejido que se mantiene siempre humedecido. Ambos termómetros se exponen a una corriente de aire, ya sea mediante un ventilador mecánico o por agitación. La evaporación en el tejido que envuelve al bulbo húmedo hace descender la temperatura. Si la atmósfera está saturada (humedad relativa = 100%) la evaporación es nula y por lo tanto ambas temperaturas coinciden. La relación entre la diferencia de temperatura que miden los dos termómetros y la humedad relativa no se directa, ya que depende de la temperatura real del aire, y de la presión atmosférica. En el diagrama adjunto se muestra esta relación para una presión de 1000 hPa (mb).

HIGROMETROS MECANICOS

Están basados en la propiedad de algunos materiales (cabello humano, algodón, seda, papel, etc.) de cambiar su dimensión física dependiendo de la humedad relativa del aire. El cabello humano fue ampliamente utilizado como sensor de humedad relativa en los higrógrafos de estaciones meteorológicas convencionales, así como también en los primeros equipos de radiosondeo.

HIGROMETRO BASADO EN EL USO DE COMPONENTE ELECTRONICA

Se utiliza la capacidad de ciertos materiales de absorber moléculas de vapor de agua a través de su superficie. Este proceso, al modificar las propiedades eléctricas de una componente de un circuito electrónico (resistencia o condensador), permite crear una señal eléctrica que es proporcional a la humedad. Este tipo de sensor se utiliza en estaciones meteorológicas automáticas y en equipos de radiosondeos.

HIGROMETRO ESPECTROSCOPICO

Son equipos relativamente caros pero de alta precisión. Se basan en la propiedad del vapor de agua en la atmósfera de absorber radiación infrarroja en bandas específicas del espectro electromagnético. La mayor o menor radiación absorbida se relaciona con el nivel de humedad del aire. Se utilizan en mediciones de humedad donde se requiere una alta tasa de muestreo (por ejemplo en mediciones de turbulencia del vapor de agua).

PRECIPITACIÓN (PLUVIOMÉTRICA) Y EVAPORACIÓN.

En las mediciones de precipitación se mide la tasa de acumulación de lluvia o nieve, por unidad de área horizontal. Una acumulación de 1 mm corresponde a un volumen de 1 litro por metro cuadrado de superficie.

PRECIPITACION LIQUIDA

El instrumento se llama pluviómetro. Su componente central es un recipiente que acumula el agua, hasta que es vaciada y medida, ya sea en forma manual o automática. En la mayoría de las estaciones meteorológicas la precipitación se mide una vez al día, de modo que el sistema de almacenamiento está diseñado de modo de evitar la evaporación. Para esto el agua captada escurre por un pequeño agujero en el fondo del recipiente de captación hacia un contenedor de acumulación.

Un pluviómetro de amplia utilización en estaciones meteorológicas automáticas tiene un sistema de registro que incluye dos pequeños recipientes que alternan sus posiciones para recibir el agua que se recibe de la zona de captación. La capacidad de llenado de cada recipiente equivale a una cierta cantidad de precipitación (por ejemplo 0.2 mm). Cuando uno de los recipientes se llena, el sistema se vuelca por gravedad, vaciándose el

recipiente lleno y quedando el otro en posición de llenado. Mediante un contacto eléctrico se registra la frecuencia de volcamientos, lo cual es proporcional a la intensidad de la precipitación.

TEMPERATURA (TERMOMETRÍA)

En la escala Celsius de temperatura, el cero de la escala corresponde a la temperatura del punto de congelamiento del agua, y el 100 a su temperatura de ebullición, ambos a nivel del mar. Las mediciones se realizan en un ambiente con buena ventilación pero protegido de la radiación solar, para lo cual se utiliza el típico cobertizo meteorológico de madera que caracteriza las estaciones meteorológicas convencionales. En las estaciones automáticas se utilizan cobertizos de menor tamaño.

TERMOMETROS LIQUIDOS (mercurio o alcohol)
Se mide en una escala graduada la expansión o compresión que experimenta un líquido (mercurio o alcohol) en el interior de un tubo de vidrio, como respuesta a los cambios de la temperatura ambiente. El mercurio permite medir temperaturas superiores a -39 C, mientras que los termómetros de alcohol pueden medir valores por encima de -62 C.

En el termómetro de máxima, que se instala en forma horizontal, cuando la temperatura aumenta el mercurio se expande pasando por un punto de estrangulación en la columna cerca de la base del bulbo del termómetro. Cuando la temperatura empieza a descender, la columna de mercurio se corta en ese punto, quedando marcada la posición de máxima expansión. Una vez leída la temperatura máxima, la columna se devuelve a la posición más comprimida mediante agitación mecánica del termómetro.

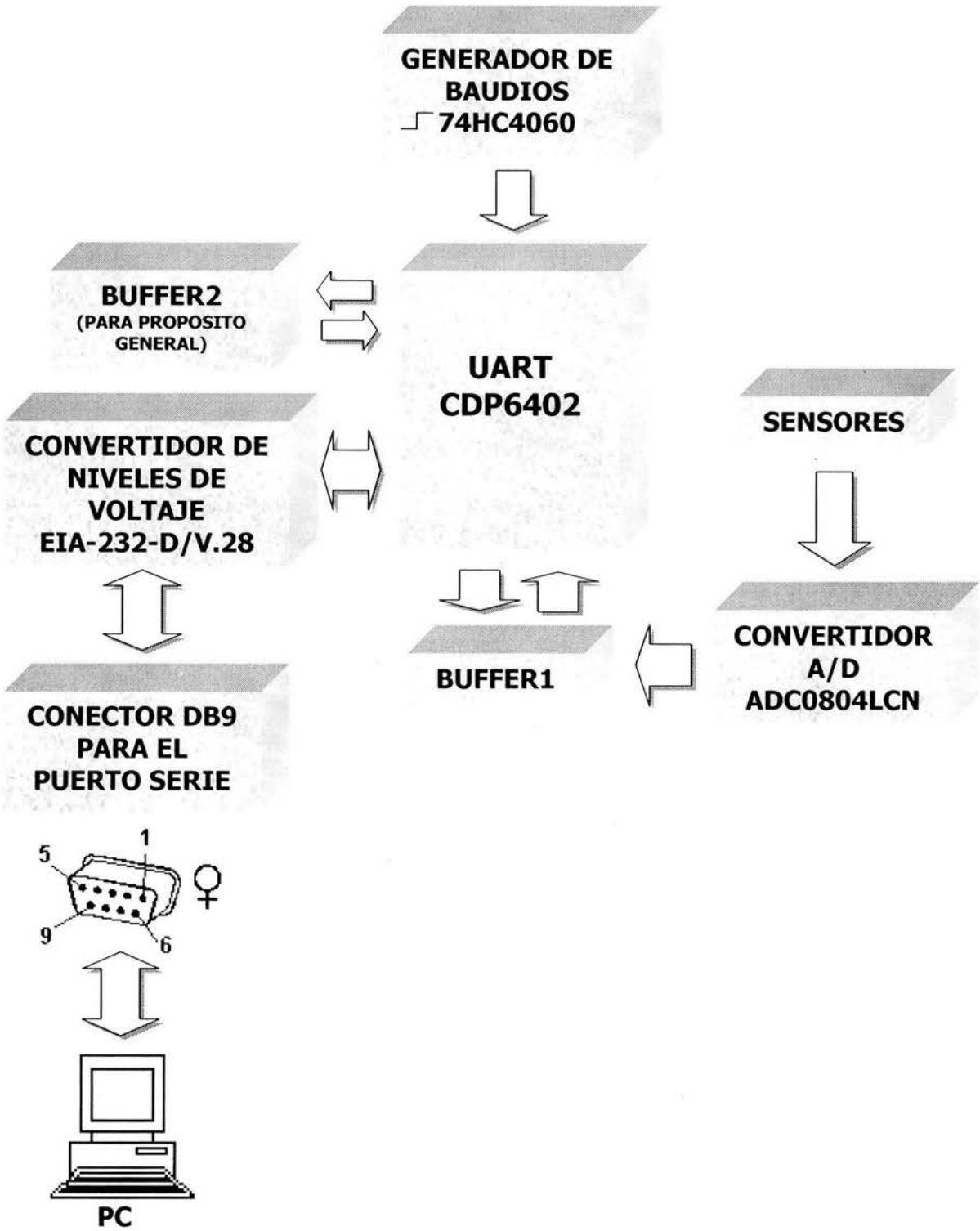
En el termómetro de mínima, que se instala en forma horizontal, existe un pequeño vástago que es arrastrado por el extremo de la columna que se comprime al descender la temperatura (se utiliza alcohol en lugar de mercurio). Si la temperatura aumenta, el vástago permanece estacionario en la posición alcanzada en la temperatura mínima.

LAMINA BIMETALICA
El sensor de temperatura está constituido por dos láminas de metales distintos que están pegadas entre sí. Como los coeficientes de expansión térmica de ambos metales no son iguales, las láminas se flectan al cambiar la temperatura. Mediante un sistema de palancas, la deflexión se amplifica y se registra en una banda de papel. Este método fue ampliamente utilizado en estaciones meteorológicas convencionales (termógrafo), así como también en los primeros sistemas de radiosondeo para medir perfiles verticales de temperatura en la atmósfera (radiosonda).

TERMOCUPLA
El principio de medición se basa en el hecho que en el punto de contacto de dos metales distintos se produce una diferencia de potencial eléctrico, cuya magnitud depende de la temperatura a la cual se encuentra dicho punto. El cambio de potencial es del orden de 40 microvolts por cada grado de temperatura. Los metales más frecuentemente utilizados en la construcción de termocuplas son el Cobre y el Constantan.

SENSORES BASADO EN EL USO DE COMPONENTES ELECTRONICAS
En las décadas más recientes se ha producido un notable desarrollo tecnológico de sensores meteorológicos basados en el uso de circuitos electrónicos. En el caso de la temperatura, se utiliza la propiedad de algunos metales de modificar su resistencia eléctrica con la temperatura. El metal más utilizado es el platino. También se utilizan termistores, que están basados en semiconductores cuya resistencia varía con la temperatura.

3.3 DIAGRAMA A BLOQUES DEL SISTEMA DE ADQUISICION DE DATOS



3.4 UART CDP6402

La UART (Transmisor/Receptor Universal Asíncrono) CDP6402 es un circuito CMOS. Este dispositivo sirve para comunicar microprocesadores o computadoras con canales de datos seriales asíncronos. La UART CDP402 esta diseñada para proporcionar el formato necesario y el control entre los canales de datos serie y paralelo. El lado receptor convierte el dato con formato serial, el bit de arranque, bit de paridad y bit de parada en un dato con formato en paralelo, verificando que la transmisión, los bits de paridad y los bits de parada sean los correctos. Por su parte, el lado transmisor convierte el dato paralelo en dato en forma serial y automáticamente les añade los bits de arranque, paridad y de parada. En la fig. 3.1 se ve la configuración de pines.

La palabra de datos puede ser de 5, 6, 7 u 8 bits de longitud. La paridad puede ser par, impar o ninguna. Los bits de parada pueden ser 1, 1.5, o 2 (cuando se transmite código de 5-bits).

La UART CDP6402 puede ser usada en un gran número de aplicaciones incluyendo módem, impresoras, periféricos, terminales de video, sistemas remotos de adquisición de datos, entre otros.

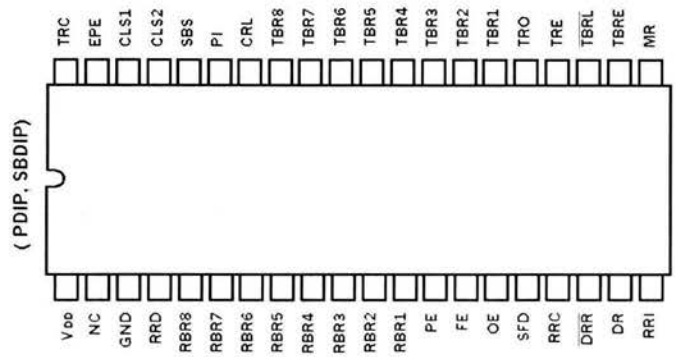


Fig. 3.1 La UART CDP6402

TABLA 3.1 DEFINICION DE LA FUNCION DE CADA PIN DE LA UART CDP6402

PIN	SIMBOLO	DESCRIPCION
1	VDD	Polo positivo de la fuente de voltaje.
2	N/C	No conectado.
3	GND	Tierra (Vss)
4	RRD	RECEIVER REGISTER DISABLE (Deshabilitar Registro Receptor) Un nivel alto en la entrada RRD pone a las salidas del registro receptor RBR1- RBR8 en un estado de alta impedancia.
5	RBR8	El contenido del Registro del Buffer Receptor aparece en estas salidas tri-estadas. Si la palabra es menor que 8 caracteres esta se justifica a partir de la derecha de RBR1.
6	RBR7	Ver pin 5 - RBR8
7	RBR6	
8	RBR5	
9	RBR4	
10	RBR3	
11	RBR2	
12	RBR1	
13	PE	PARITY ERROR (Error de Paridad). Un nivel alto en PE indica que la paridad recibida no es igual a la programada por el control de bits.
14	FE	FRAMING ERROR (Error de Trama). Un nivel alto en FE indica que el primer bit de parada fue invalidado. FE permanece en estado activo hasta que el bit de parada de los siguientes caracteres recibidos sea valido.
15	OE	OVERRUN ERROR (Error de Desbordamiento). Un nivel alto en OE indica que la bandera del dato recibido no se limpio antes de que el último caracter fuera transferido al registro del buffer receptor. El error es limpiado en el bit de parada de los siguientes caracteres si DRR ha sido activado (DRR se activa en bajo)

PIN	SIMBOLO	DESCRIPCION
16	SFD	STATUS FLAGS DISABLE (Deshabilitar el Estado de las Banderas). Un nivel alto en SFD pone las salidas PE, FE, OE, DR y TBRE en un estado de alta impedancia.
17	RRC	RECEIVER REGISTER CLOCK (Registro Receptor del Reloj). El RRC es 16X con respecto al dato.
18	DRR	Un nivel bajo en DATA RECEIVED RESET (Reiniciar Dato Recibido) limpia la salida del dato recibido (DR), a un nivel bajo.
19	DR	Un nivel alto en DATA RECEIVED (Dato Recibido) indica que un caracter ha sido recibido y transferido al registro del buffer receptor.
20	RRI	El dato serial en RECEIVER REGISTER INPUT (Entrada del Registro Receptor) es sincronizado dentro del registro receptor.
21	MR	Un nivel alto en MASTER RESET (MR) limpia PE, FE, OE y DR, e inicializa TRE, TBRE, y TRO. TRE es realmente colocado en el primer pulso de reloj de TRC después de que MR se pone en alto.
22	TBRE	Un nivel alto en TRANSMITTER BUFFER REGISTER EMPTY (Registro del Buffer Transmisor Vacío) indica que el registro del buffer transmisor ha transferido sus datos al registro transmisor y esta listo para un nuevo dato.
23	TBRL	Un nivel bajo en TRANSMITTER BUFFER REGISTER LOAD (Carga del Registro del Buffer Transmisor) coloca el dato presente en las entradas TBR1 – TBR8 en el Registro del Buffer Transmisor. Una transición de nivel bajo a alto en TBRL envía el dato al Registro Transmisor. Si el Registro Transmisor esta ocupado, la transferencia se retarda automáticamente hasta que los dos caracteres sean transmitidos completamente.
24	TRE	Un nivel alto en TRANSMITTER REGISTER EMPTY (Registro Transmisor Vacío) indica la transmisión completa de un caracter incluyendo los bits de parada.
25	TRO	El caracter del dato, el bit de inicio y el bit de parada aparecen en serie en el TRANSMITTER REGISTER OUTPUT (Salida del Registro Transmisor).
26	TBR1	El caracter del dato es cargado en el TRANSMITTER BUFFER REGISTER (Registro del Buffer Transmisor) mediante las entradas TBR1 – TBR8.
27	TBR2	Ver pin 26 – TBR1
28	TBR3	
29	TBR4	
30	TBR5	
31	TBR6	
32	TBR7	
33	TBR8	
34	CRL	Un nivel alto en CONTROL REGISTER LOAD (Carga del Registro de Control) carga el registro de control.
35	PI♦	Un nivel alto en PARITY INHIBIT (Inhibir Paridad) inhibe la generación de paridad, la revisión de paridad y pone la salida PE en nivel bajo.
36	SBS♦	Un nivel alto en STOP BIT SELECT (Selección de Bits de Parada) selecciona 1.5 bits de parada para un formato de caracter de 5 bits de dato y 2 bits de parada para otras longitudes.
37	CLS2♦	Estas entradas programan el CHARACTER LENGTH SELECTED (Selección de longitud de caracter). CLS1 en bajo CLS2 en bajo 5 bits; CLS1 en alto CLS2 en bajo 6 bits; CLS1 en bajo CLS2 en alto 7 bits; CLS1 en alto CLS2 en alto 8 bits.
38	CLS1♦	Ver Pin 37 – CLS2
39	EPE♦	Cuando PI esta en bajo, un nivel alto en EVEN PARITY ENABLE (Habilitar Paridad Par) genera y revisa la paridad par. Un nivel bajo selecciona paridad impar.
40	TRC	El TRANSMITTER REGISTER CLOCK (Registro Transmisor del Reloj). El RRC es 16X con respecto al dato.

♦ Ver Tabla 3.2 (Funciones de la palabra de control)

TABLA 3.2 FUNCIONES DE LA PALABRA DE CONTROL

PALABRA DE CONTROL					BITS DE DATOS	BIT DE PARIDAD	BIT(S) DE PARADA
CLS2	CLS1	PI	EPE	SBS			
L	L	L	L	L	5	PAR	1
L	L	L	L	H	5	PAR	1.5
L	L	L	H	L	5	IMPAR	1
L	L	L	H	H	5	IMPAR	1.5
L	L	H	X	L	5	NINGUNO	1
L	L	H	X	H	5	NINGUNO	1.5
L	H	L	L	L	6	PAR	1
L	H	L	L	H	6	PAR	2
L	H	L	H	L	6	IMPAR	1
L	H	L	H	H	6	IMPAR	2
L	H	H	X	L	6	NINGUNO	1
L	H	H	X	H	6	NINGUNO	2
H	H	L	L	L	7	PAR	1
H	H	L	L	H	7	PAR	2
H	L	L	H	L	7	IMPAR	1
H	L	L	H	H	7	IMPAR	2
H	L	H	X	L	7	NINGUNO	1
H	L	H	X	H	7	NINGUNO	2
H	H	L	L	L	8	PAR	1
H	H	L	L	H	8	PAR	2
H	H	L	H	L	8	IMPAR	1
H	H	L	H	H	8	IMPAR	2
H	H	H	X	L	8	NINGUNO	1
H	H	H	X	H	8	NINGUNO	2

Nota: X = No importa.

DESCRIPCIÓN DE OPERACIÓN

Inicialización y control

Un pulso positivo en la entrada del MASTER RESET, reinicia los valores del control, estado de las banderas, y los registros del buffer receptor. Además, pone la salida serial (TRO) en estado Alto. La sincronización es generada desde las entradas de reloj RRC y TRC en una frecuencia 16 veces proporcional en razón del número de bits del dato serial. Las entradas RRC y TRC pueden ser manejadas por un reloj común, o se pueden manejar de forma independiente por dos relojes diferentes. La entrada del Registro de Control de Carga, CONTROL REGISTER LOAD (CRL), es utilizada para cargar los bits de control con la finalidad de: Inhibir paridad, PARITY INHIBIT (PI), Habilitar Paridad Par, EVEN PARITY ENABLE (EPE), Selector de Bits de Parada, STOP BIT SELECTS (SBS), y Selector de Longitud de Caracter, CHARACTER LENGTH SELECTS (CLS1 y CLS2). Estas entradas pueden conectarse a VSS o VDD con CRL conectado a VDD. Cuando la inicialización esta completada, la UART esta lista para realizar operaciones de transmisión y/o recepción. En la fig. 3.2 podemos ver el diagrama a bloques de la UART.

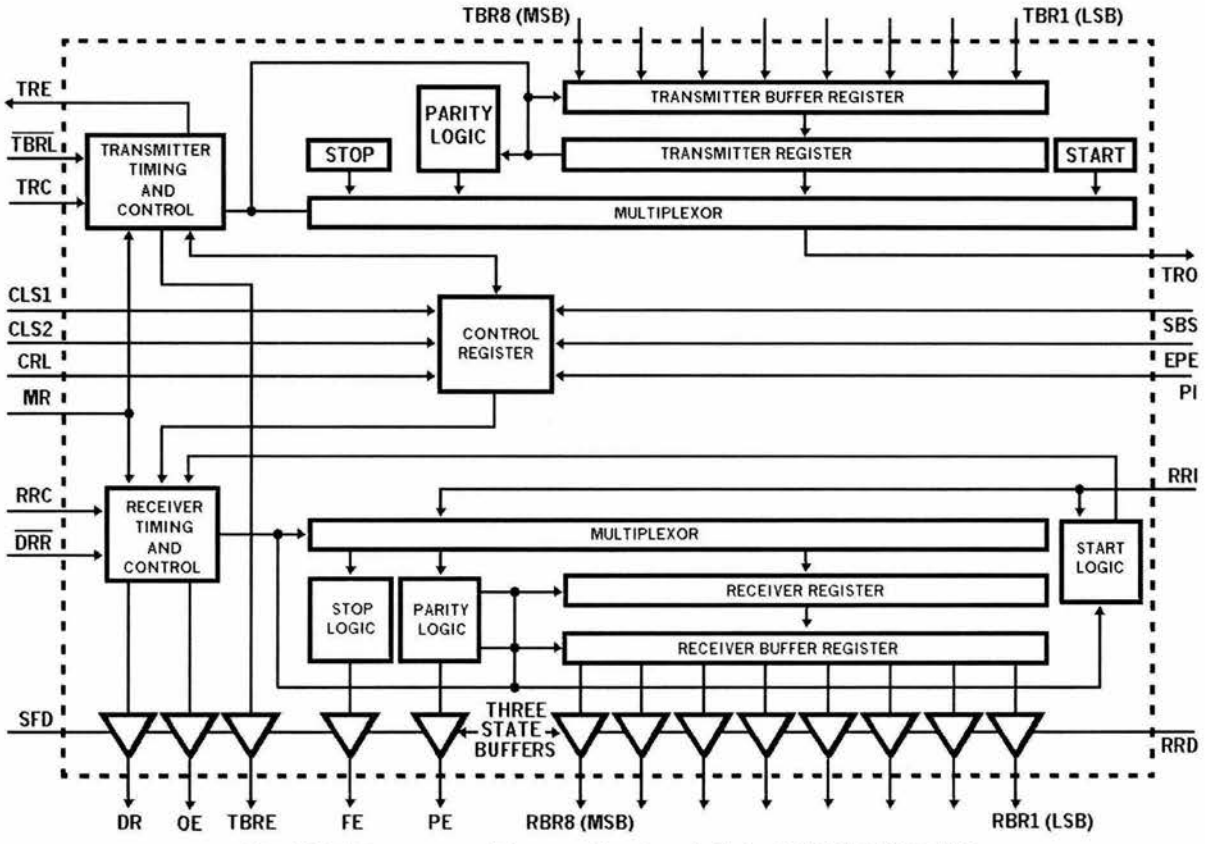


Fig. 3.2 Diagrama a bloques funcional de la UART CDP6402

Operación de Transmisión

La sección transmisora de la UART admite datos en paralelo, y los transmite en forma serial a través del terminal TRO. (Ver fig. 3.3).

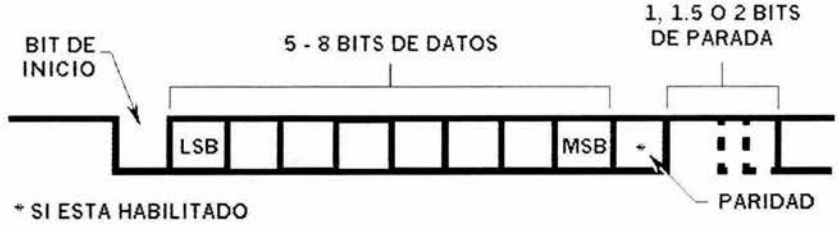


Figura 3.3 Formato de Datos Serial.

Las formas de onda y el periodo de transmisión se muestran en la figura 3.4.

(A) El dato es cargado al registro transmisor a través de las entradas TBR1 a TBR8 por un nivel lógico bajo en la entrada $\overline{\text{TBRL}}$. Si la Palabra utilizada es menor de 8 bits, solo se usan los bits menos significativos. El caracter es justificado hacia la derecha a partir del bit menos significativo, TBR1.

(B) El rizo de $\overline{\text{TBRL}}$ limpia el TBRE (Registro Del Buffer Transmisor Vacío). Un 1/2 ciclo a 11/2 ciclo después, dependiendo de cuando ocurre el pulso $\overline{\text{TBRL}}$ con respecto a TRC, el dato es transferido al Registro del Buffer Transmisor y TRE es limpiado. Un ciclo después, TRE es puesto a un nivel lógico alto.

La salida de datos es sincronizado por TRC. El rango de la frecuencia de reloj es 16 veces proporcional al dato.

(C) Un segundo pulso en $\overline{\text{TBRL}}$ carga el dato en el Registro del Buffer Transmisor. La transferencia del dato al Registro Transmisor es retardada hasta que la transmisión del caracter este completa.

(D) El dato es transferido automáticamente al Registro Transmisor y la transmisión de este caracter inicia.

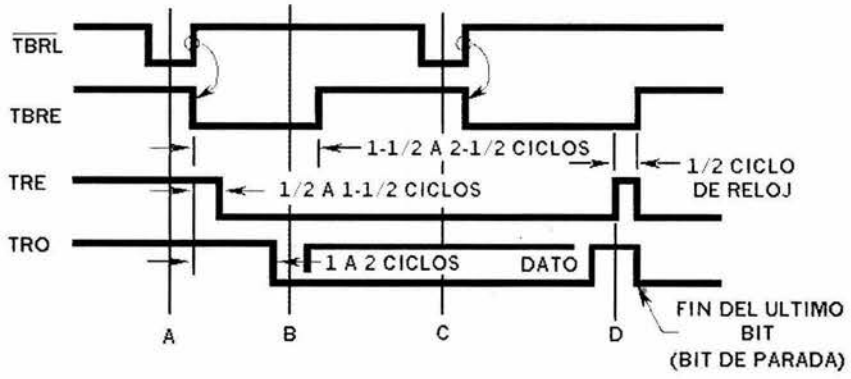


Fig. 3.4 Formas de Onda de Transmisión

Operación de Recepción

Las formas de onda y el periodo de recepción se muestran en la figura 3.5.

El dato es recibido en forma serial en la entrada RRI. Cuando el dato no esta siendo recibido, la entrada RRI debe permanecer en ALTO. El dato es sincronizado a través de RRC. La proporción del reloj es 16 veces la proporción de datos. Las formas de onda y el periodo se muestran en la figura 3.

(A) Un nivel bajo en $\overline{\text{DRR}}$ limpia la línea DR

(B) Durante el primer bit de parada el dato es transferido desde el registro receptor al registro RB. Si la palabra es menor que 8 bits, los bits mas significativos sin usar tendrán un nivel lógico BAJO. El caracter de salida es justificado a la derecha del bit menos significativo RBR1. Un nivel lógico ALTO en OE indica un desbordamiento. Un desbordamiento ocurre cuando DR no ha sido limpiado antes de que el caracter presente fuera transferido al RBR.

(C) ½ ciclo de reloj después DR es inicializado a un nivel lógico ALTO y FE es evaluado. Un nivel lógico ALTO en FE indica que se recibió un bit de parada inválido.

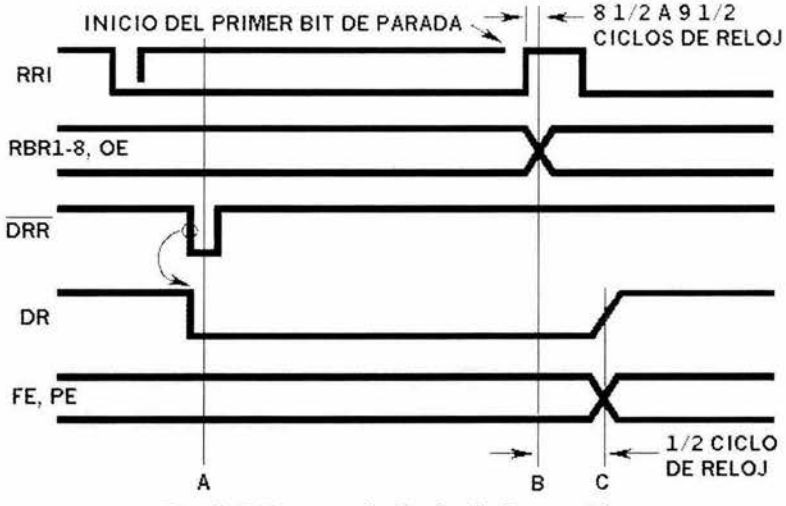


Fig. 3.5 Formas de Onda de Recepción.

Detección del bit de Inicio.

El receptor usa un reloj con una frecuencia 16 veces proporcional al número de bits del dato serial, para la sincronización (Fig. 3.6). El bit de inicio puede haber ocurrido como máximo antes de que este fuera detectado, como esta indicado por la porción sombreada. El centro del bit de inicio esta definido como 7 ½ ciclos de reloj. Si el reloj receptor es una onda simétrica cuadrada, el centro de el bit de inicio estará ubicado dentro de ±½ de ciclo de reloj, ± 1/32 de bit o ±3.125%. El receptor inicia la búsqueda para el siguiente bit de inicio en 9 ciclos de reloj dentro del primer bit de parada.

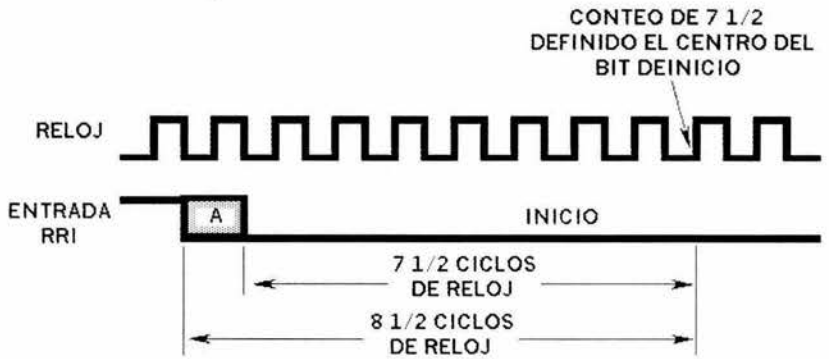


Fig. 3.6 Formas de Onda de la Detección del Bit de Inicio.

3.5 GENERADOR DE BAUDIOS 74HC4060

El 74HC4060 (Fig. 3.7 y 3.8) es un dispositivo de tecnología Si-gate CMOS de alta velocidad. Este circuito contiene contadores/divisores de 14 etapas de onda con acarreo y osciladores con tres terminales del oscilador (RS, R_{TC} y C_{TC}), diez salidas moderadas (Q_3 a Q_9 y Q_{11} a Q_{13}) y un master reset primario (MR).

La configuración del oscilador esta diseñada para utilizar cualquier circuito RC o circuitos con oscilador del cristal. El oscilador puede ser reemplazado por una señal externa de reloj en la entrada RS. En este se deben mantener los otros pines del oscilador (R_{TC} y C_{TC}) flotando.

El contador avanza en la transición negativa de RS. Un nivel ALTO en MR vuelve a inicializar el contador (Q_3 a Q_9 y Q_{11} a Q_{13} = BAJO), independiente de otras condiciones de entrada.

TABLA 3.4 DESCRIPCIÓN DE PINES DEL 74HC4060

PIN	DESCRIPCION	NOMBRE Y FUNCION
1, 2, 3	Q_{11} a Q_{13}	Salidas del contador
7, 5, 4, 6, 14, 13, 15	Q_3 a Q_9	Salidas del contador
8	GND	Tierra (0 V)
9	C_{TC}	Conexión externa del capacitor
10	R_{TC}	Conexión externa de resistencia
11	RS	Entrada de Reloj/Pin Oscilador
12	MR	Master Reset
16	Vcc	Positivo de la Fuente de Voltaje

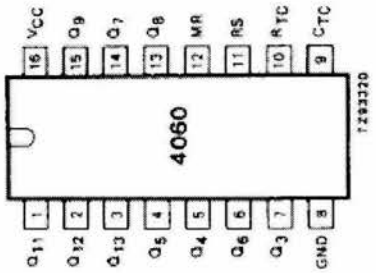


Fig. 3.7 Configuración de Pines

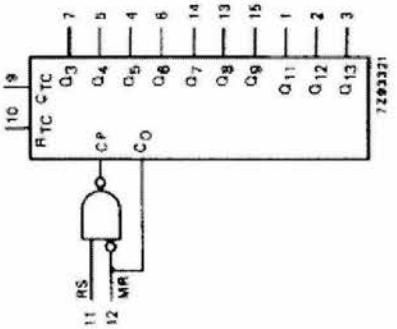


Fig. 3.8 Símbolos Lógicos

3.6. CONVERTIDOR DE NIVELES DE VOLTAJE EIA-232-D/V.28

El MC145406 (Fig. 3.9) es un circuito integrado CMOS de silicio que combina 3 controladores y 3 receptores para cumplir con las especificaciones eléctricas de las normas EIA-232-D y CCITT V28.

APLICACIONES

- Interfaz de MODEM.
- Interfaz telefónica de Voz/Datos
- Computadoras Lap-top
- Interfaz de UART

CARACTERISTICAS

- **Controladores**
- Fuente de voltaje de ± 5V a ± 12V
- Impedancia de 300 Ω
- Limitación de corriente de salida
- Compatible con TTL
- Máximo razón de cambio 30V/μs
- **Receptores**
- Rango de voltaje de entrada de ± 25V
- Impedancia de entrada de 3 a 7 kΩ
- **General**
- Muy bajo consumo de corriente para una larga vida de batería

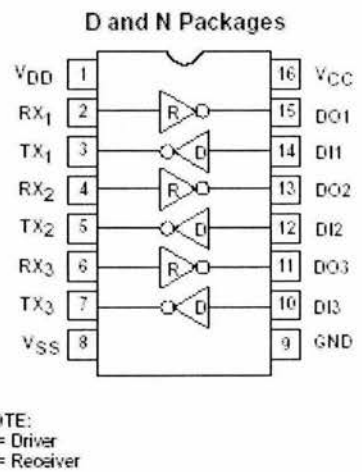


Fig. 3.9 Configuración de pines

TABLA 3.5 DESCRIPCION DE PINES DEL CONVERTIDOR DE NIVELES DE VOLTAJE EIA-232-D/V.28.

# PIN	SIMBOLO	DESCRIPCION DE PIN
1	V_{DD}	**Polo positivo de la fuente de voltaje.** El pin positivo de la fuente de voltaje, el cual esta típicamente entre 5 y 12 Volts.
8	V_{SS}	**Polo negativo de la fuente de voltaje.** El pin negativo de la fuente de voltaje, el cual esta típicamente entre -5 y -12 Volts.
16	V_{CC}	**Polo positivo digital** (Máximo +5.5V)
9	GND	**Tierra digital.**
2, 4, 6	RX_1, RX_2, RX_3	**Receptor de Entradas de Datos.** Estos son los pines de entrada de recepción de las señales EIA-232-D cuyos voltajes pueden estar en el rango de +25V y -25V. Un voltaje entre +3V y +25V es decodificado como un ESPACIO y ocasiona que el correspondiente pin DO cambie a Tierra (0V); Un voltaje entre -3V y -25V es decodificado como una MARCA y ocasiona que el correspondiente pin DO cambie a Vcc. La impedancia nominal de entrada es de 5kΩ. Un pin de entrada en circuito abierto o puesto a tierra es interpretado como una marca, forzando al pin DO a Vcc.
11, 13, 15	DO1, DO2, DO3	**Salida de Datos.** Estos son los pines de las salidas digitales del receptor, los cuales varían de Vcc a GND. Un ESPACIO en el pin RX, ocasiona que en el pin DO se produzca un CERO lógico; una MARCA produce un UNO lógico. Cada pin de salida es capaz de controlar una carga de entrada LSTTL.
10, 12, 14	DI1, DI2, DI3	**Entrada de Datos.** Estos son los pines de entrada de alta impedancia a los controladores.
3, 5, 7	TX1, TX2, TX3	**Transmisor de Salida de Datos.** Estos son los pines de salida de transmisión de las señales EIA-232-D cuyos voltajes oscilan entre V_{DD} y V_{SS}. Un UNO lógico en la entrada DI ocasiona que su salida TX correspondiente cambie a V_{SS}. Un CERO lógico en la entrada DI ocasiona que su salida TX correspondiente cambie a V_{DD}. (Los voltajes de salida serán ligeramente menores que V_{DD} o V_{SS} dependiendo de la carga de salida). Cuando el MC145406 está apagado ($V_{DD} = V_{SS} = Vcc = GND$), la impedancia mínima de salida es 300Ω.

RANGOS MAXIMOS ABSOLUTOS

SIMBOLO	PARAMETRO	RANGO	UNIDADES
Vcc	Fuente de voltaje	-0.5 a +6.0	V
V_{DD}	Fuente de voltaje	-0.5 a +13.5	V
V_{SS}	Fuente de voltaje	+0.5 a -13.5	V
V_{IR}	Rango de voltaje de entrada Entradas RX_{1-3} Entradas DI_{1-3}	 $(V_{SS} - 15)$ a $(V_{DD} + 15)$ -0.5 a (Vcc + 0.5)	V
	Corriente DC por PIN	±100	mA
T_A	Rango de Temperatura de Operación	0 a +70	°C

NOTA: Este dispositivo contiene circuitos de protección para resguardar las entradas contra daños producidos por voltajes estáticos altos o por campos eléctricos; Sin embargo, se recomienda tomar las precauciones necesarias con el fin de evitar el uso de voltajes mas altos que el rango máximo permitido para este circuito en alta impedancia. Para la operación correcta, es recomendado que los voltajes en los pines DI y DO estén en el rango $GND<V_{DI}<V_{DD}$ y $GND<V_{DO}<V_{CC}$. También, el voltaje en el pin RX debería estar limitado a + 25V, y el pin TX debería estar limitado para $V_{SS} < V_{TX1-3} < V_{DD}$. Las entradas sin uso siempre deben estar conectadas con un nivel lógico apropiado (por ejemplo, GND o V_{CC} para DI, y V_{SS} o V_{DD} para RX) de voltaje.

3.7. CONVERTIDOR ANALÓGICO / DIGITAL ADC0804LCN

La conversión de datos analógicos a digitales, para que la información pueda ser procesada se consigue con un convertidor analógico/digital de la serie ADC080X (Fig. 3.10) , estos convertidores A/D disponen de una salida de 8 bits, son de tecnología CMOS y son muy rápidos con una velocidad de conversión de aproximadamente 100µS. Trabajan con una fuente de voltaje de 5V, dispone de reloj interno o externo, aunque lo mas fácil es poner un puente R/C entre los pines 19,4 y GND.

TABLA 3.6 CONFIGURACIÓN DE PINES DEL CONVERTIDOR ADC0804LCN

PIN	NOMBRE	FUNCION	LOGICA
1	CS- Chip Select	Habilita el chip	I / 0
2	RD- Salida autorizada	Lee la información	I / 0
3	WR- Start conversion	Iniciar conversión	I / 0
4	CLKIN	Entrada de reloj	
5	INTR	Indicador fin conversión	I / 0
6	V +	Señal positiva analógica	-0.3/16V
7	V -	Señal negativa analógica	0
8	A GND	Tierra analógica	0
9	Vref/2	1/2 máximo del Pin 6	
10	D GND	Tierra digital	
11/18	DB7 a DB0	Salidas digitales	I / 0
19	CLK R	Salidas reloj interno	
20	Vcc	Alimentación	hasta 6,5V

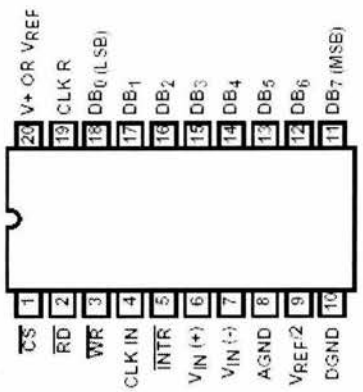


Fig. 3.10 El convertidor ADC0804LCN

Las instrucciones para el convertidor A/D son:

CS (Chip Select) autoriza el funcionamiento del convertidor
WR (Write) Da la orden de inicio del convertidor
RD (Read) Efectúa la lectura de los datos
INTR (Indicador de fin conversión)

Con CS y WR con posición lógica 1 el convertidos A/D se bloquea y no actúa. La conversión empieza con la llegada de un pulso 1 a la entrada de WR si la entrada de CS esta a 0. Durante la transición de 1 a 0 de la señal en la entrada del WR o del CS, se reinician el controlador interno, y el registro de datos y la salida del INTR se ponen a 1.

Después de que la conversión sea completa la patita INTR realiza una transición de 1 a 0, esto puede ser usado para interrumpir un microprocesador o señalar la posibilidad de un nuevo resultado para otra conversión. Una operación de lectura del RD con CS a 0 limpia la INTR y autoriza los latch de salida.

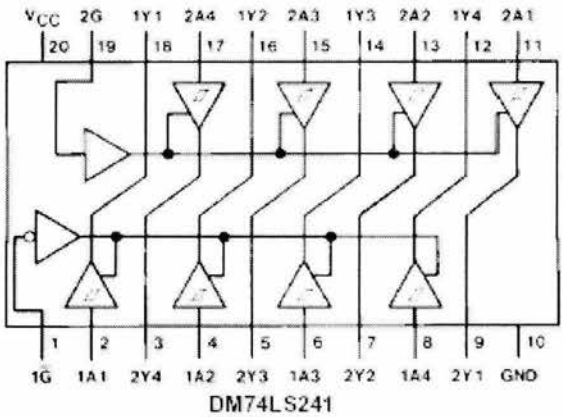
Los periodos entre transiciones de 0 a 1 o de 1 a 0 deben ir precedidos de unos periodos de tiempo en espera de 0,5 mS para permitir la adecuación de todos los circuitos internos del convertidor A/D. Este tiempo puede ser menor aunque es cuestión de controlar las tablas de tiempos según el proceso que se este realizando.

Para conseguir una conversión en continuo CS y RD deben de estar a 0 y la patita INTR conectada a la entrada de WR. Esta conexión INTR/WR fuerza a 0 la patita de WR y asegura la operación del circuito.

3.8 BUFFER 74LS241 Y COMPUERTA AND 74LS08

CARACTERISTICAS DEL BUFFER 74LS241

- Salidas de 3-Estados directamente
- Controlador de Línea
- Receptor de Línea



DM74LS241

Entradas				Salidas	
G	$\overline{G}$	1A	2A	1Y	2Y
X	L	L	X	L	
X	L	H	X	H	
X	H	X	X	Z	
H	X	X	L		L
H	X	X	H		H
L	X	X	X		Z

Fig. 3.11 El cto. 74LS241

CARACTERISTICAS DEL 74LS208

- El cto. contiene cuatro compuertas AND de 2 entradas TTL.

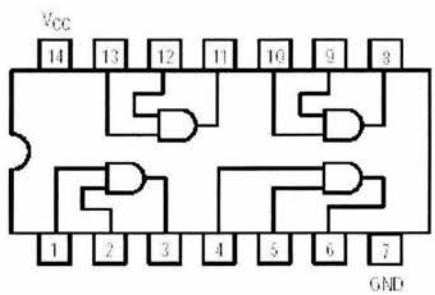


Fig.3.12 El cto. 74LS08.

3.9 SENSOR DE TEMPERATURA LM335Z

El LM335Z (Fig. 3.13) es un sensor de temperatura de precisión, fácil de calibrar. Operando como un zener de 2 terminales el LM335 tiene un voltaje de rompimiento directamente proporcional a una temperatura absoluta de 10 mV/°K. Cuando lo calibramos el LM335 a 25°C, este tiene un error típico menor a 1°C sobre un rango de temperatura de 100°C. El LM335 opera en un rango desde -40°C a 100°C.

Características:

- Calibrado directamente en °K
- Precisión inicial disponible de 1°C
- Opera desde 400 µA a 2 mA
- Impedancia dinámica menor a 1 Ω
- Fácil calibración
- Amplio rango de temperatura de operación
- Rango de 200°C
- Bajo Costo

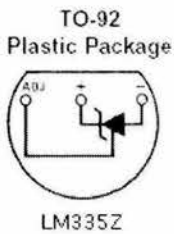


Fig. 3.13 El sensor LM335Z

3.10. EL CIRCUITO Y SU FUNCIONAMIENTO

DESCRIPCION GENERAL

El sistema de adquisición de datos esta diseñado para monitorear variables ambientales como temperatura, humedad, presión atmosférica, velocidad del viento, entre otras. En la práctica solo implementamos el sensor de temperatura LM335Z, por ser el de menor costo, pero el circuito tiene la flexibilidad de que se le puedan conectar más dispositivos electrónicos y a su vez puede ser utilizado para aplicaciones de control digital.

El circuito consta de las siguientes etapas:

- La etapa de alimentación, que es la que suministra los diferentes voltajes para el circuito.
- El generador de baudios, que es el encargado de suministrar los pulsos de reloj a la UART.
- El conector DB9 para el puerto serial.
- La etapa de conversión de niveles de voltaje (niveles EIA-232-2 a niveles de voltaje TTL)
- La UART CDP6402, que es la encargada de convertir los datos de formato paralelo a formato serial y viceversa.
- El Buffer de recepción, que nos permite manejar 2 entradas digitales de 7 bits en formato paralelo.
- El convertidor A/D que convierte la señal analógica de los sensores, en un dato digital en formato paralelo.
- Los sensores (Temperatura, Humedad, Presión, etc.)

Los parámetros que utilizamos para la comunicación con el puerto serial son los siguientes:
8 bits de datos, sin paridad, 1 bit de parada, velocidad de transmisión de 9,600 bps y manejamos el protocolo XON/XOFF, el cual veremos mas adelante.

ETAPA DE ALIMENTACION

Esta etapa consta de un circuito de puente de diodos y de un regulador de voltaje. En la fig. 314 se observa que el circuito se puede alimentar con CA o CD. El valor de la fuente de voltaje AC que se utilizó es de 16 V_{AC}, o bien se puede utilizar una fuente de DC con un rango entre 12 y 25 V_{DC}.

Para convertir el voltaje AC en DC utilizamos un arreglo de puente de diodos para rectificar la señal y un capacitor electrolítico para filtrar la señal. El regulador de voltaje LM7805 lo utilizamos para mantener la señal de Vcc en un nivel de $5V_{DC}$.

V_{DD} y V_{SS} los obtenemos de un divisor de voltaje de dos resistencias del mismo valor como se ve en la figura. V_{DD} y V_{SS} son los niveles de voltaje que utilizamos para el puerto serial y Vcc y GND los utilizamos para alimentar a nuestros circuitos CMOS y TTL.

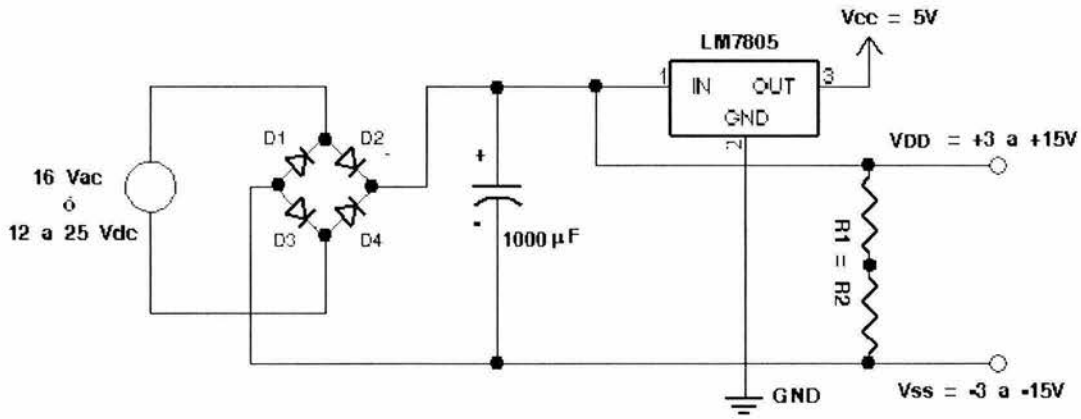


Fig. 3.14 Circuito de la etapa de alimentación

GENERADOR DE BAUDIOS (RELOJ)

La UART requiere un pulso reloj con un rango que sea 16 veces más alto que el número de bps que se utilice para la interfaz de comunicación entre el circuito y la computadora. De esta manera no podemos obtener rangos muy altos usando cristales comunes, como los de 1.8432 MHz. y 2.4576 MHz.

La tabla 3.7 nos indica los posibles valores de baudios que podemos obtener usando el 74HC4060.

TABLA 3.7 VALORES DE BAUDIOS DEL 74HC4060

Salida	1.8432 MHz	2.4546 MHz
Out 2	115.2 KBPS	153.6 KBPS
Q4	7200 BPS	9600 BPS
Q5	3600 BPS	4800 BPS
Q6	1800 BPS	2400 BPS
Q7	900 BPS	1200 BPS
Q8	450 BPS	600 BPS
Q9	225 BPS	300 BPS

La figura 3.15 nos indica la manera en que se conecta el 74HC4060 para obtener una razón de baudios de 9600 bps.

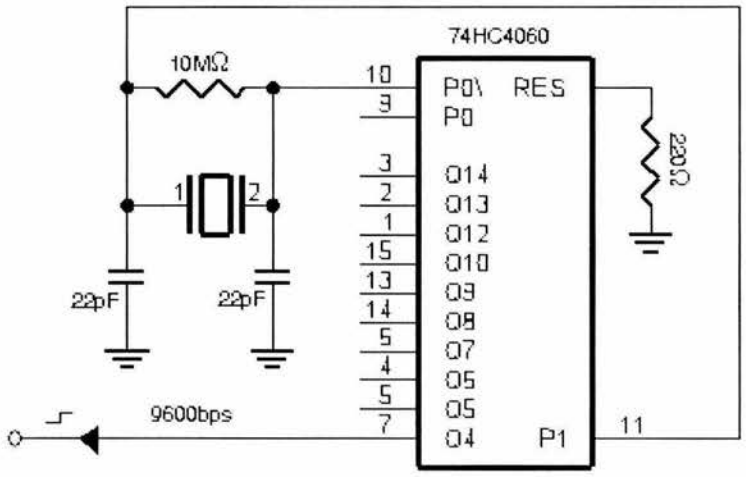


Fig. 3.15 Circuito del generador de baudios

CONFIGURACION DEL DB-9 (PUERTO SERIAL)

En la figura podemos observar lo siguiente:

- El pin1 del DB-9 (CD) esta conectado a Vss que nos indica un estado binario de MARCA (1 binario en las señales RS-232), y le indica a la computadora que se le esta enviando una señal de portadora.
- El pin 6 (DSR) se conecta a Vss para indicarle a la computadora que el sistema está listo para intercambiar información.
- El pin 8 (CTS) esta en un nivel lógico 1 para indicarle a la computadora que el sistema esta listo para transmitir información.
- El pin 9 (RI) lo conectamos a 1 lógico ya que de estar en 0 lógico se produciría una interrupción en la comunicación.
- El pin 2 (RD) esta conectado a la entrada TRO de la UART para recibir los datos en formato serial por parte de esta.
- El pin 3 (TD) esta conectado a la entrada RRI de la UART para transmitir información en formato serial por parte de esta.
- El pin 7 (RTS) esta conectado al pin TBRL de la UART y lo utilizamos para enviar una señal de reloj a la UART para su funcionamiento.
- Los pines 4 (DTR) y 5 (SG) no se conectan.

Cabe recordar que los pines del DB-9 no están conectados directamente a la UART sino que pasan por el convertidor de niveles de voltaje.

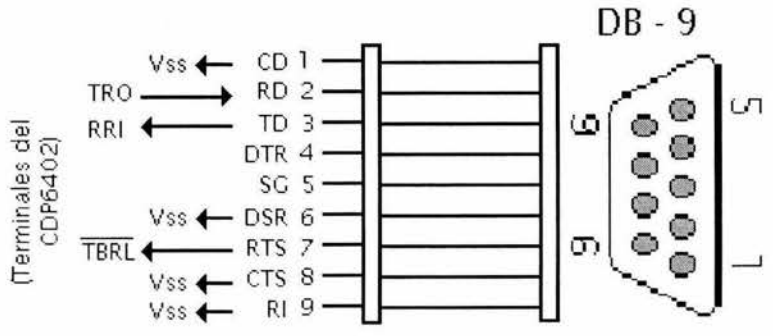


Fig. 3.16 Conexión del DB-9

CONVERSION DE LOS NIVELES DE VOLTAJE

Esta etapa del circuito la utilizamos para convertir los niveles de voltaje lógicos (0 y 5V), en niveles de voltaje EIA-232-D y viceversa.

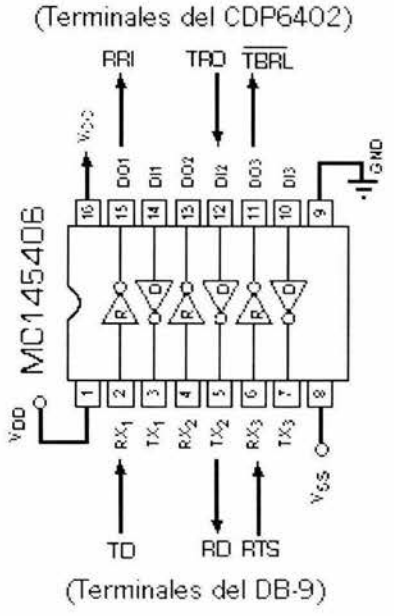


Fig. 3.17 Conexión del MC145406

Como se puede ver en la figura 3.17, el cto. MC145406 lo utilizamos como interfaz entre el CDP6402 y el puerto serial del PC, debido a que trabajan con voltajes diferentes.

LA UART CDP6402

La UART CDP6402 trabaja con un voltaje de 5 Vdc. En la figura 3.15 se observa que los pines 1 y 3 son de alimentación. Como vimos anteriormente, los pines 38 (CLS2), 39 (CLS1), 35 (PI), 39 (EPE), 36 (SBS) son para configurar la palabra de control. Para este sistema en particular, utilizaremos un formato de datos de 8 bits, sin paridad, con un bit de parada y una velocidad de transmisión de 9600 bps.

La configuración de la palabra de control para este formato es la siguiente:

PALABRA DE CONTROL					BITS DE DATOS	BIT DE PARIDAD	BIT(S) DE PARADA
CLS2	CLS1	PI	EPE	SBS			
H	H	H	X	L	8	NINGUNO	1

- ⇨ El pin 4 (RRD) esta en bajo, ya que un nivel alto en la entrada RRD retiene a las salidas del registro receptor RBR1- RBR8 en un estado de alta impedancia.

- ⇨ El pin 16 (SFD) esta en bajo, ya que un nivel alto en SFD pone las salidas PE, FE, OE, DR y TBRE en un estado de alta impedancia.

- ⇨ El pin 18 (DRR) esta en alto, ya que un nivel bajo en DRR pone la salida del dato recibido (DR), a un nivel bajo.

- ⇨ El pin 21 (MR), esta en bajo, debido a que en nivel alto se reinicia el circuito.

- ⇨ El pin 34 (CRL), esta en un nivel alto, ya que así se carga el registro de control.
- ⇨ Los pines 17 (RRC) y 40 (TRC) están conectados al terminal Q4 del 74HC4060, el cual les suministra una señal de reloj para trabajar a 9600 bps.

Las terminales 20 (RRI) y 25 (TRO) son para la transmisión y recepción de datos en forma serial hacia el PC. RRI y TRO están conectadas al DI2 y al DO1 del MC145406 respectivamente, que a su vez esta conectado a las terminales TD y RD del puerto serial.

El pin 23 (TBRL), se utiliza para la Carga del Registro Transmisor. Un nivel bajo en TBRL coloca el dato presente en las entradas TBR1 – TBR8, en el Registro del Buffer Transmisor. Una transición de nivel bajo a alto en TBRL envía el dato al Registro Transmisor. Si el Registro Transmisor esta ocupado, la transferencia se retarda hasta que los caracteres sean transmitidos completamente. TBRL esta conectado al DO3 del MC145406, que a su vez esta conectado al RTS del puerto Serial.

Las transiciones de alto a bajo y viceversa del TBRL , son controladas mediante Software y se transfieren mediante el terminal RTS del puerto serial.

Por último, las terminales 5 a 12 (TBR8 a TBR1) están conectadas a las salidas del Buffer de Recepción (Que veremos mas adelante), que su vez esta conectado al convertidor A/D ADC0804 a través del Buffer1, y son las encargadas de recibir el dato en forma paralela, para que posteriormente este sea transmitido en forma serial.

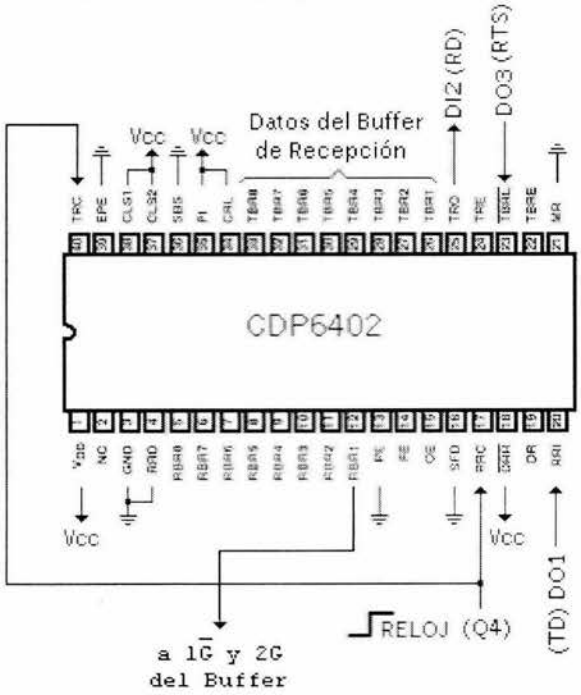


Fig. 3.18 Conexión de la UART CDP6402

EL BUFFER DE RECEPCION

En la fig. 3.19 se observa que el **Buffer1** esta constituido por las terminales 1A1 a 1A4 del cto. 74LS241 (a) y los pines 1A1 a 1A3 del 74LS241 (b) siendo el pin señalado como B7 el del bit más significativo y B1 el del bit menos significativo. Por otro lado, el **Buffer2** esta constituido por los pines 2A1 a 2A4 del cto. 74LS241 (a) y los pines 2A1 a 2A3 del 74LS241 (b) siendo el pin señalado con B7 el bit más significativo y B1 el menos significativo.

El terminal $1\overline{G}$ y el 2G del cto. 74LS241 (a) están conectados entre sí y a su vez están conectados a los pines $1\overline{G}$ y 2G del cto. 74LS241 (b). Al momento de seleccionar un Buffer mediante software, si se selecciona el Buffer1 se envía un 0 al pin RBR1 de la UART y este hace que el pin $1\overline{G}$ este en 0, por lo que el Buffer1 se activa y el Buffer2 se pone en estado de alta impedancia al estar 2G en 0. Si se selecciona el buffer2 se envía un 1 a RBR1, el Buffer2 se activa y el Buffer1 pasa a un estado de alta impedancia.

Para poder conectar nuestros 2 buffers a la UART utilizamos compuertas AND que nos sirven como una especie de filtro, al dejar pasar el dato de la señal que esta activa. Cabe mencionar que las compuertas TTL toman a Z (Alta Impedancia) como un 1 lógico.

TABLA 3.8 CONFIGURACIÓN DEL BUFFER

RBR1	1A	2A	Salida
0	0	Z	0
0	1	Z	1
1	Z	0	0
1	Z	1	1

RBR1 = 0 Se activa el Buffer1
RBR1 = 1 Se activa el Buffer2

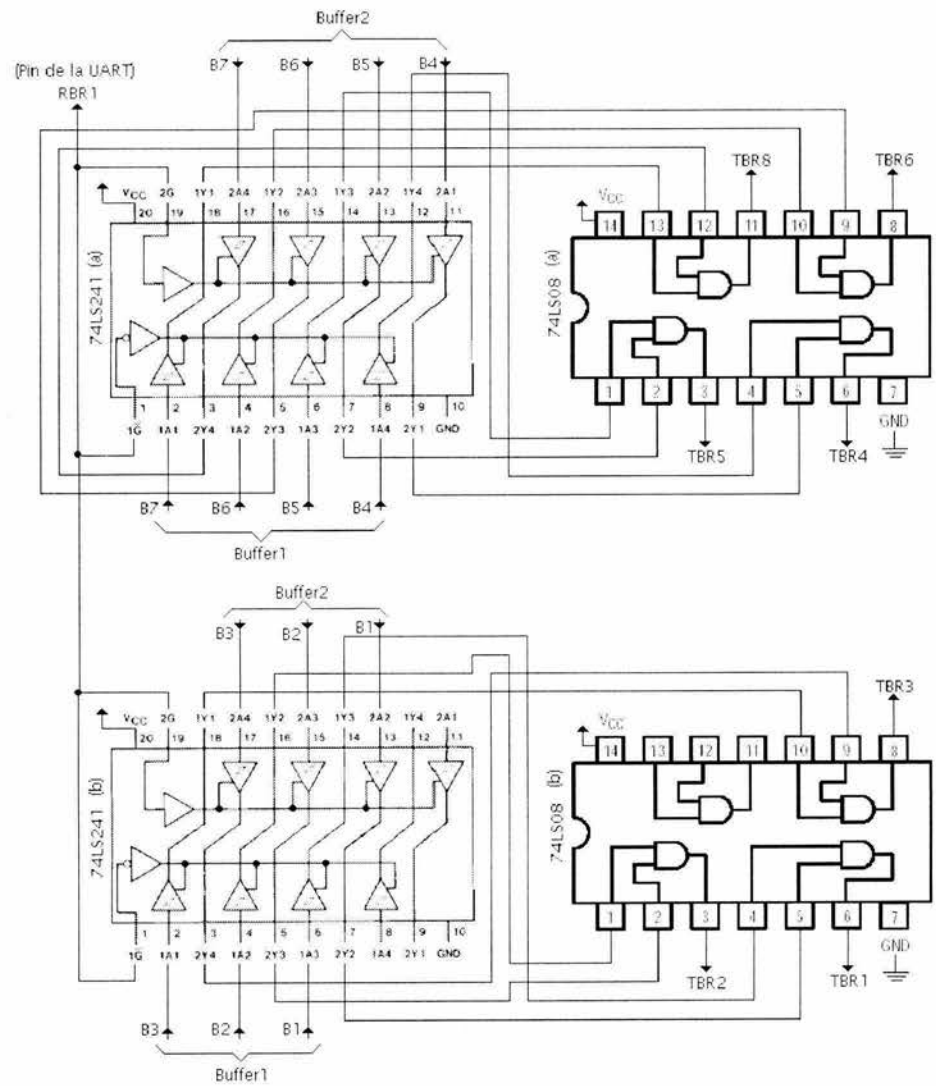


Fig. 3.19 Conexión del Buffer de recepción.

EL CONVERTIDOR A/D ADC0804

El funcionamiento y configuración del convertidor ADC0804 se explico anteriormente. En la figura 3.20 se puede observar la conexión y configuración del cto. ADC0804. Para nuestro fin, el circuito esta configurado para realizar una conversión en continuo, por lo que $\overline{CS}$ y $\overline{RD}$ están a 0 y el pin $\overline{INTR}$ esta conectado a la entrada de $\overline{WR}$. La resistencia y el capacitor en las entradas CLKR y CLIN son para el funcionamiento del reloj. Las terminales DB0 a DB7 pertenecen al dato digital y estas van al CDP6402 a través del Buffer1. Las terminales VIN(+) y VIN(-) están conectadas aun multiplexor analógico, el cual nos suministra la señal analógica del sensor (En la práctica utilizamos un sensor de temperatura).

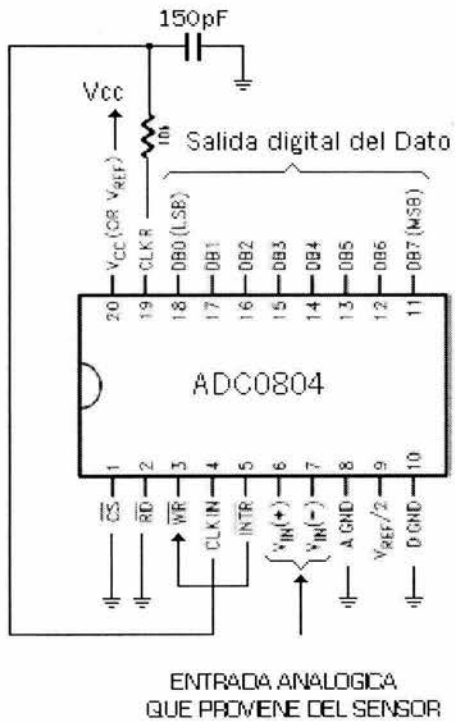


Fig. 3.20 Conexión y configuración del ADC0804

EL SENSOR DE TEMPERATURA LM335Z

En la figura 3.21, se muestra como se conecta el LM335 para obtener una salida de 10 mV por cada °K de variación de temperatura.

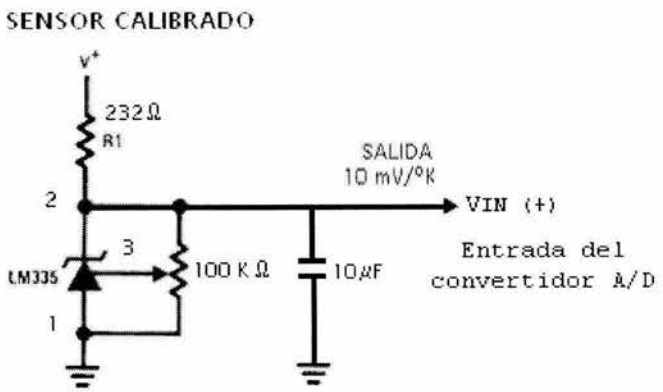


Fig. 3.21 Conexión del sensor de temperatura LM335.

FUNCIONAMIENTO DEL CIRCUITO

La figura 3.22 nos muestra el circuito total del Sistema de Adquisición de Datos.

El sistema recibe del **sensor de temperatura** una señal analógica con un rango de 0v a 5v, esta señal pasa al **convertidor A/D** el cual la muestrea y nos proporciona una señal digital altamente confiable, la señal digital es canalizada al **Buffer1**, este **Buffer1** tiene como función la intercomunicación en paralelo entre el **convertidor A/D** y la **UART**. Por otro lado el **Buffer2** nos amplia la posibilidad de canalizar mas variables para nuestro sistema de adquisición de datos. (En forma general el buffer1 y buffer2 nos permiten ampliar el número de bits de recepción de la **UART** de 7 a 14 bits). El **Buffer1** y el **Buffer2** son seleccionados mediante software, solo uno puede estar activo.

La **UART CDP6402** recibe el dato digital del **Buffer** (1 o 2) en forma paralela y lo convierte en formato serial añadiéndole el bit de inicio y el bit de parada, este dato digital pasa al **convertidor de niveles de voltaje** que lo adapta en niveles de voltaje EIA-232 para poder ser interpretado por la computadora desde el **puerto serie (RS-232)**, este dato digital es procesado mediante el **software** desarrollado y este a su ver lo despliega en pantalla.

El **software** de la computadora a su vez envía un dato digital a través del **puerto serial RS-232** para seleccionar que **Buffer** se desea monitorear. El dato pasa por el **convertido de niveles de voltaje** y es convertido en niveles de voltaje TTL-CMOS para poder ser interpretado por la **UART**.
La UART recibe el dato en formato serial enviado por la computadora con niveles TTL-CMOS y lo convierte en forma paralela. La UART analiza la petición del software para seleccionar al Buffer que se desea monitorear, y comienza el ciclo de lectura de datos por parte de la computadora.

Cabe mencionar que la computadora envía una señal de reloj a la UART a través del pin RTS del puerto serial vía software para el funcionamiento de la UART (TBRL). Esta señal pasa por el mismo proceso conversión de niveles de voltaje que las señales de transmisión y recepción que vimos anteriormente.

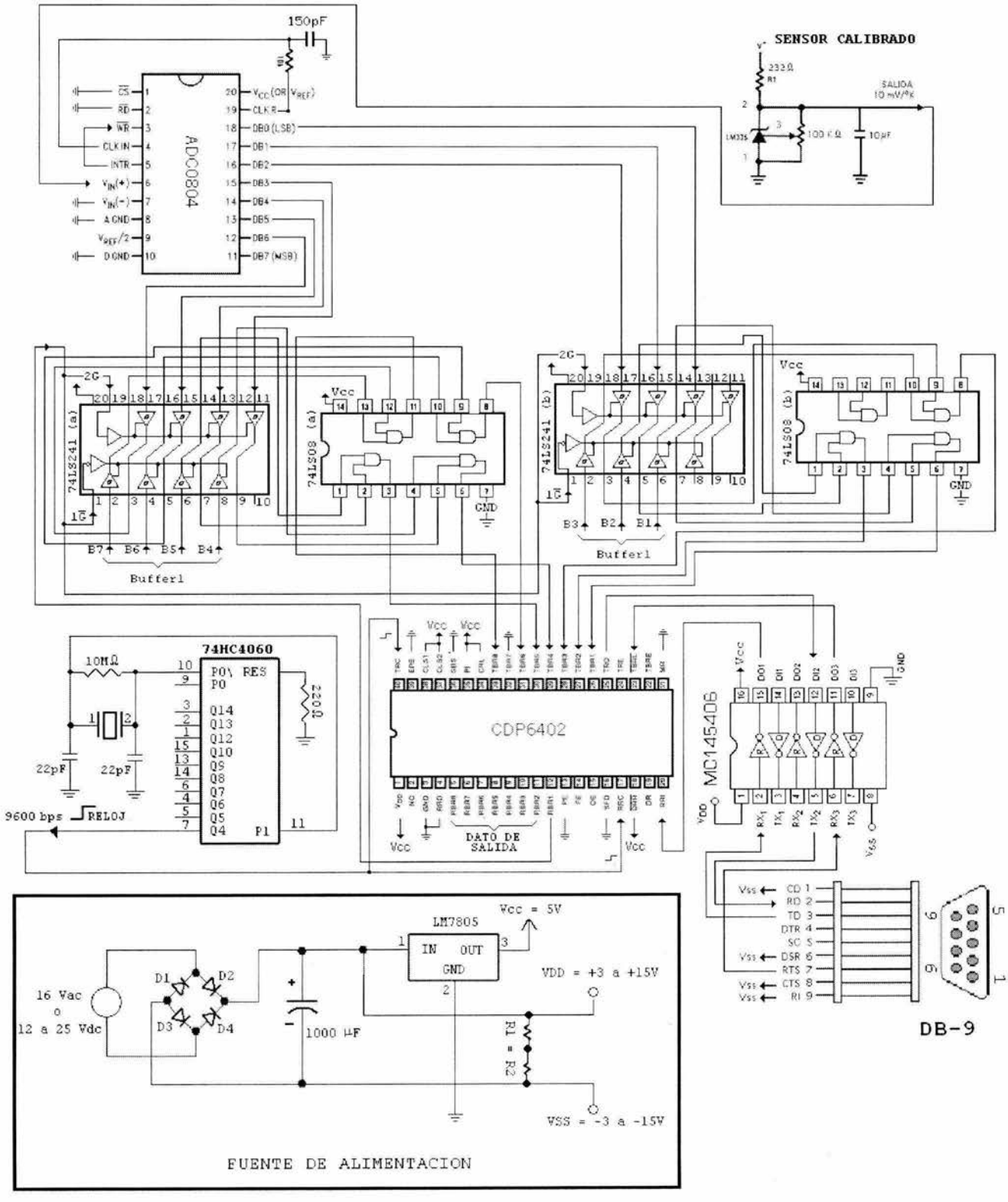


Fig. 3.22 CIRCUITO DEL SISTEMA DE ADQUISICION DE DATOS

CAPITULO 4.

EL PROGRAMA PARA ENVIAR Y RECIBIR DATOS POR EL PUERTO SERIE

CAPITULO 4. EL PROGRAMA PARA ENVIAR Y RECIBIR DATOS POR EL PUERTO SERIE

4.1 EL CONTROL DE COMUNICACIONES MSCOMM

INTRODUCCIÓN

La programación de las comunicaciones con Visual Basic y Visual C++ pueden llevarse a cabo utilizando las funciones de la API de Windows, no obstante, este procedimiento resulta mucho más complicado que utilizar el propio control de comunicaciones MSCOMM.VBX que Microsoft incluye con su lenguaje de programación.

Dado que este control puede ser utilizado tanto desde Visual Basic como desde Visual C++, las exposiciones que aquí se hacen son válidas para ambos lenguajes, salvo en aquellos casos en los que se indican expresamente la diferencia, como es el caso de la sintaxis de las instrucciones.

PROGRAMACIÓN CON VISUAL BASIC Y VISUAL C++

La programación con los lenguajes visuales esta basada en crear ventanas (llamadas de forma general formularios, *forms*) y colocar sobre ellas los objetos (controles) sobre los que posteriormente se van a escribir los códigos para controlar todos los hechos o acontecimientos que puedan ocurrir durante la ejecución del programa.

Cada objeto (control) está ligado a una serie de códigos que permanecen inactivos hasta que ocurre el suceso (evento) que lo activa. Por ejemplo, un suceso puede ser hasta un clic del ratón sobre un objeto botón.

Los objetos, o mejor dicho, los controles que se pueden utilizar para formar las distintas ventanas (*forms*) que forman la aplicación son proporcionados por las propias herramientas (*Toolbox*) de Visual Basic y C++. Estos controles son los botones, textos, gráficos, iconos, líneas, listas, etc. Que dan vida a cada ventana de la aplicación. Uno de estos controles, aunque no presente en todas las versiones de Visual Basic y C++, es el control de comunicaciones, el cual se encarga de controlar a los puertos serie de la computadora.

Además de los controles que acompañan al propio lenguaje de programación, cualquier programador puede añadir a sus aplicaciones nuevos controles (*Custom Control*) de los muchos que hay comercializados por distintas casas de software.

Los controles vienen en ficheros con la extensión .VBL El fichero del control de comunicaciones tiene el nombre MSCOMM.VBX, y fue desarrollado por la empresa Crescent Software Inc. para ser incluido en estos lenguajes de Microsoft (tanto Visual Basic edición profesional como Visual C++).

Cada control tiene distintas propiedades, como son el color, la forma, el tamaño, si está visible o no, o cualquier otra característica relacionada con el *mismo.* De *la* misma forma, a cada control (objeto) le pueden ocurrir distintos eventos (acontecimientos) predefinidos Por ejemplo, a un botón le pueden hacer un clic, un doble clic, o sencillamente puede pasar el cursor por encima de él.

PROGRAMAR CON LAS FUNCIONES API DE WINDOWS

Como se ha comentado anteriormente, si no se dispone del control de comunicaciones MSCOMM.VBX, se puede utilizar las funciones de la API de Windows. Y los pasos son:

Abrir el puesto de comunicaciones con la función **OpenComm**. En esta operación se deben definir el puerto de comunicaciones y los tamaños de los buffers de entrada y salida. Esta función devuelve un número que identifica al puerto de comunicaciones. En el caso de ocurrir un error, devolverá un número negativo.

Por ejemplo:

nCID=OpenComm ("COM1", 1024,1024)

Construir un bloque de control de datos (DCB) mediante la función BuildCommDCB. Los argumentos son la definición de los parámetros del puerto y la estructura DCB. La definición de los parámetros del puerto se realiza con el mismo formato que el comando MODE del DOS. Por ejemplo:

BuildCommDCB("COM1:2400,n,8,1",&dcb)

Establecer las características del puerto mediante la función SetCommState. El argumento es la estructura DCB, en la que previamente se guardaron dichas características. Por ejemplo:

SetCommState(&dcb)

Cuando se desean enviar datos, se utiliza la función WriteComm. Esta función tiene los siguientes parámetros: el número identificativo de **dispositivo** dévuelto por OpenComm, la cadena de caracteres a enviar número de bytes a transmitir. Por ejemplo:

WriteComm (nCID, BufferSAL$,Len (BufersSAL$))

Establecer un bucle que permanezca a la espera de los datos que se puedan recibir. Para leer los datos se utiliza la función ReadComm con los siguientes parámetros: el valor ID que identifica al puerto de comunicaciones, una cadena de caracteres que contendrá los datos recibidos numero de caracteres recibidos. Por ejemplo:

NumCar=ReadComm (nCID, LpBuf, Len(LpBuf))

Si ocurre un error, Windows bloqueará el puerto correspondiente hasta se llame a la función GetCommError. Los parámetros de esta función el valor ID que identifica al puerto de comunicaciones y una estructura tipo COMSTAT que contendrá el estado del dispositivo. Generalmente llama a GetCommError después de llamar a ReadComm. Por eiemplo:
Rerror=GetCommError(nCid, lpStar)

Por ultimo, cuando ya no se vaya a utilizar más el puerto de comunicaciones, se cierra con la función CloseComm. Si el puerto no se pudiera cerrar alguna causa, esta función devolvería un valor negativo. Por ejemplo:

Vr=CloseComm (nCid)

EL CONTROL DE COMUNICACIONES MSCOMM.VBX

El control de comunicaciones permite que se puedan realizar transmisiones y recepciones de datos a través del puerto serie. Cada control de comunicaciones sólo puede ser utilizado para controlar un puerto serie, por lo que si se necesita acceder a más de un puerto serie, se debe usar más de un control.

El control de comunicaciones tiene predefinido un solo evento para manejar las comunicaciones. Este evento, llamado *OnComm,* se dispara cada vez que ocurre cualquier cosa relacionada con el puerto de comunicaciones (se recibe un nuevo carácter, se detecta un cambio de las señales de la interfaz o se produce cualquier tipo de error en la comunicación).

Los verdaderos artífices que permiten un control completo del puerto serie son las propiedades relacionadas con el control de comunicaciones. Son las siguientes: *Break, CDHolding, CDTimeout, CommEvent, CommID, CommPort, CTSHolding, ClSTimeout, DSRHolding, DSRT¡meou¡, DTREnable, Handshaking, InBufferCount,*

JnBufferSize, Index, !nput, JnputLen, Interval, Left, Name, NuIIDiscard, OutBufferCount, OurBufferSize, Output, ParityReplace, PortOpen, RThreshold, RTSEnable, Settings, SThreshold. Tag, y Top.

De estas propiedades, salvo *Left, Name, Tag y Top,* que son utilizadas también por otros controles, el resto son específicas del control de comunicaciones. De ellas, la propiedad *Índex* está sólo disponible en Visual Basic (y no en Visual C++), y la propiedad *Name,* en Visual C++ recibe el nombre de *CtlName,* al igual que en Visual Basic 1 0.

A pesar de que la lista de propiedades del control de comunicaciones es bastante larga, las principales con las que hay que familiarizarse primero son las siguientes:

CommPort	Determina al número del puerto de comunicaciones a utilizar.
Settings	Determina los parámetros de comunicación (velocidad, paridad, bits de datos y bits de parada).
PortOpen	Permite abrir y cerrar el puerto de comunicaciones definido con CommPort
Input	Lee y elimina los caracteres del buffer receptor
Output	Escribe una cadena de caracteres en el buffer transmisor.

Al igual que con otros lenguajes de programación, cuando se programa sobre Windows con Visual Basic y Visual C++, el control de comunicaciones permite programar las comunicaciones de dos formas:

Por sondeo. Realizando sondeos de los eventos o los errores de comunicaciones que se produzcan. Eso se lleva a cabo continuamente el valor de la propiedad *CommEvent.* Este método puede ser adecuado en el caso de que se realice una pequeña aplicación, como puede ser un simple marcador telefónico; en cualquier otro caso, el mejor método es el de controlar los eventos.
Controlar los eventos. Con este método, cada vez que ocurre cualquier evento o error relacionado con el puerto de comunicaciones, es detectado (evento *OnComm)* y se actúa en consecuencia. Para ver una lista de todos los posibles eventos y errores de comunicaciones consultar la descripción de la propiedad *CommEvent.*

4.1.1. EVENTOS ONCOMM

El evento *OnComm* se genera cada vez que se produce un evento o un error relacionado con las comunicaciones. El evento *OnComm* siempre va acompañado de un cambio de valor de la propiedad *CommEvent.* El valor de *CommEvent* nos indica la naturaleza del evento o error ocurrido.

La sintaxis es la siguiente:

V. Basic	Sal MSCommOnCommO
V. C++	void *CMyDialog::OnMSCommOnComm(UINT,lnt,CWnd*,LPVOID)*

La propiedad *CommEvent* contiene el código numérico del error o del evento ocurrido. Hay que tener en cuenta que sí se fijan las propiedades *RThreshold o SThreshold* a 0, los eventos MSCOMM_EV_RECEIVE o MSCOMM_EV_SEND respectivamente, no se mostrarán.

El siguiente ejemplo muestra como manejar los errores de comunicaciones y los eventos. Después de cada instrucción *Case* se puede insertar un código para manejar cada error o evento particular.

```
Sub Comm_OnComm( )
Select Case Comm1.CommEvent
'Errores
Case MSCOMM_ER_BREAK 'Error de corte
'Código para manejar el error de corte
Case MSCOMM_ER_CDTO 'Temporizador de CD)
'Código para manejar este error
Case MSCOMM_ER_CTSTO 'Temporizador de CTS
```

```
'Código para manejar este error
Case MSCOMM_ER_DSRTO 'Temporizador de DSR
'Código para manejar este error
Case MSCOMM_ER_FRAME 'Error de trama
'Código para manejar este error
Case MSCOMM_ER_OVERRUN 'Saturación. del puerto
'Código para manejar este error
Case MSCOMM_ER_RXOVER 'Saturación del buffer receptor 'Código para manejar este error
Case MSCOMM_ER_RXPARITY 'Error de paridad
'Código para manejar este error
Case MSCOMM_ER_TXFULL 'Buffer transmisor lleno Código para manejar este error
'Eventos
Case MSCOMM_EV_CD       'Cambio en la línea CD) 'Código para manejar este evento
Case MSCOMM_EV_CTS      'Cambio en la línea CTS 'Código para manejar este evento
Case MSCOMM_EV_DSR      'Cambio en la línea DSR 'Código para manejar este evento
Case MSCOMM_EV_RING 'Detectada señal de llamada 'Código para manejar este evento.
Case MSCOMM_EV_RECEIVE 'Recibido RThreshold caracteres 'Código para manejar este evento
Case MSCOMM_EV_SEND 'SThreshold caract. en Buff .TX. 'Código para manejar este evento
End Select
EndSub
```

4.1.2. PROPIEDADES DEL CONTROL MSCOMM

PROPIEDADES

Las propiedades del control de comunicaciones permiten controlar cualquier característica o evento que tenga que ver con el puerto de comunicaciones. A continuación vamos a ver una descripción de cada una de las propiedades específicas del control de comunicaciones.

Break

Esta propiedad es utilizada pera activar o desactivar el estado de la señal de corte. Esta propiedad puede tomar los valores TRUE, para activar el estado de la señal de corte, o FALSE, para desactivarlo La señal de corte suspende la transmisión de caracteres y fija la línea de transmisión en estado de corte hasta que *se le* da el valor FALSE a la propiedad Break.

La sintaxis es la siguiente:

```
V. Basic    [form.]MSComm.Break[={True|False}]
V. C++      pMSComm->GetNumProperty("Break")
            pMSComm->SetNumProperty("Break",{TRUE|FALSE})
```

En el ejemplo siguiente se envía un estado de corte durante una décima de segundo:

```
'Se activa la condición de corte
Comm1.Break=True
Se fija la duración a 1/10 segundo
Duración!=Timer+.1
Do Until Timer>Duración!
Dummy=DoEvents()
Loop
'Se desactiva la condición de corte
Comm1.Break=False
```

CDHolding

Nos indica si la línea CD (detección de portadora) del puerto serie activa o no. Esta línea, como se sabe, es utilizada por el módem para indicarle al ordenador que hay una conexión establecida. Si la conexión se interrumpe por algún motivo (por ejemplo, el terminal remoto cuelga), dicha línea pasa a estar desactivada.

La sintaxis es la siguiente:

V. Basic *[form.]MSComm.CDHolding*
V. C++ pMSComm>GetNumProperty("CDHolding")

Los valores devueltos son TRUE, para indicar que el estado de la línea es activo, y FALSE, para indicar que dicha línea está desactivada.

Cuando se realiza una petición del estado de la ¡inca CD, el control de comunicaciones espera un tiempo como máximo *(CDTimeout)* a que dicha línea encuentre en estado activo. Si pasado ese tiempo la línea CD no se encuentra en estado activo, el control de comunicaciones generará un evento *OnComm* para indicar dicho error. El error se indica fijando la propiedad *CommEvent* al valorMSCOMM_ER_CDTO.

CDTimeout

Nos sirve para fijar el número de milisegundos de temporización que el control de comunicaciones esperará a que la línea CD esté activa antes de generar un error de temporización. El error generado puede ser identificado comprobando si la propiedad *CommEvent* ha tomado el valor MSCOMM_ER_CDTO.

En Visual Basic, si se llama a la propiedad sin asignarle ningún valor, se obtiene como respuesta el valor que tenía asignado dicha propiedad con anterioridad.

La sintaxis es la siguiente:

V. Basic [form.]MSComm.CDTimeout[=milisegundos&]

V. C++ *pMSComm->GetNumProperty("CDTimeout")*
pMSComm->GetNumProperty("CDTimeout",milisegundos)

CommEvent

Nos indica qué tipo de evento de comunicaciones o qué tipo de error ha ocurrido recientemente. Cualquier evento o error que ocurra relacionado con el puerto de comunicaciones genera un evento *OnComm* y hará que *CommEvent* tome un valor determinado, que puede ser utilizado para identificar lo ocurrido. La propiedad *CommEvent* es de sólo lectura.

La sintaxis es la siguiente:

V. Basic *[form.]MSComm.CommEvent*V.
C++ *pMSComm->GetNumProperty("CommEvent")*

Los posibles códigos de error o de evento devueltos por la propiedad CommEvent *(incluidos en los ficheros CONSTANT.TXT o CONSTANT.H) son siguientes:*

Códigos de errores
MSCOMM_ER_BREAK 1001
Error de corte. Fue recibida una señal de corte.

MSCOMM_ER_CDTO 1007

Error de temporización de CD. La línea CD estuvo desactivada por un número de milisegundos superior a CTSTimeout cuando se intentaba transmitir carácter.

MSCOMM_ER_DSRTO 1002
Error de temporizaciín de CTS. La línea CTS estuvo desactivada por un número de milisegundos superior a DSRTimeout cuando se intentaba transmitir un carácter.

MSCOMM_ER_DSRTO 1003
Error de temporización de DSR. La línea DSR estuvo desactivada por un número de milisegundos superior a DSRTimeout cuando se intentaba transmitir un carácter.

MSCOMM_ER_FRAME 1004
Error de trama. El hardware detectó un error de trama. En este caso, se debe comprobar el número de bits de datos, bits de paridad o bits de parada establecida.

MSCOMM_ER_OVERRUN 1006
Error de SATURACIÓN del puerto. Se recibió un nuevo carácter antes de que pudiese ser leido el carácter anterior.

MSCOMM_ER_RXOVER 1008
Error de saturación del buffer del receptor. No hay espacio en el buffer receptor.

MSCOMM_ER_RXPARITY 1009
Error de paridad El hardware detectó un error dc paridad.

MSCOMM_ER_TXFULL 1010
Error por buffer transmisor lleno. El buffer del transmisor se llenó al intentar enviar un nuevo carácter.

CÓDIGOS DE *EVENTOS:*

MSCOMM_EV_CD 5
Cambio en la línea CD

MSCOMM_EV_CTS 3
Cambio en la CTS

MSCOMM_EV_DSR 4
Cambio en la línea DSR. Este evento sólo ocurre cuando DSR cambia de –1 a 0.

MSCOMM_EV__EOF 7
Se recibió el carácter EOF (End of File, carácter ASCII 26).

MSCOMM_EV_RING 6
Detecta una señal de llamada (algunas UART no soportan este evento).

MSCOMM_EV_RECEIVE 2
Recibido el número de caracteres fijos por la propiedad Rthreshold. Una vez ocurrido, este evento es generado continuamente hasta que se usa la propiedad input para leer y limpiar los datos del Buffer receptor.

MSCOMM_EV_SEND 1
El buffer transmisor contiene un número de caracteres menor que el fijado por la propiedad Sthreshold.

CommID

Devuelve un manejador que identifica al dispositivo de comunicación. Este es el valor devuelto por la API de Windows de la función OpenComm, y se utiliza para las rutinas de comunicación internas de la API de Windows.

La sintaxis de esta propiedad es la siguiente:

V. Basic [form.]MSComm.CommID
V. C++ pMSComm->GetNumProperty("CommID")

CommPort

Determina el número de puerto de comunicaciones que va a ser utilizado por el control de comunicaciones. Teóricamente, este parámetro puede tomar cualquier valor entre 1 y 99; no obstante, si se intenta abrir con *PortOpen* un puerto de comunicaciones que no existe, el control de comunicaciones generará el error 68 (dispositivo no disponible). El valor por defecto fijado para *CommPort es* 1 (COM1).

La propiedad CommPort debe fijarse antes de abrir un puerto con *PortOpen.* En Visual Basic, sí se llama a la propiedad sin asignarle ningún valor, se obtiene como respuesta el valor que tenía asignado dicha propiedad con anterioridad.

La sintaxis es la siguiente:

V. Basic *[form.]MSComm.CommPort[=CommPort%]*
V. C++ pMSComm->GetNumProperty("CommPort")

CTSHolding

Nos informa del estado de la línea CTS. La señal CTS es enviada por el módem al ordenador para indicarle que está listo pera transmitir *(Clear To Send,* listo para enviar). Esta propiedad nos puede devolver el valor TRUE, indicando que la línea tiene un valor alto (módem no listo para transmitir), o FALSE, indicando que la línea tiene un valor bajo (módem listo pera transmitir). Cuando CTS tiene un valor alto durante un número de milisegundos mayor que el indicado por *CTSTimeout,* el control de comunicaciones genera un evento OnComm y le asigna a CommEvent el valor MSCOMM_ER_CTSTO.

La línea CTS es utilizada pera el control de flujo hardware RTS/CTS.

La sintaxis es la siguiente.
V. Basic *[form.]MSComm.CTSHolding*

V. C++ *pMSComm->GetNumProperty("CTSHolding")*

CTSTimeuot

Determina el número milisegundos que hay que esperar una señal CTS antes de que el control de comunicaciones genere un evento OnComm y le asigne a *ComngEvent* el valor MSCOMM_ER_CTSTO. Si la línea CTS está con valor alto durante un número de milisegundos mayor que el indicado por CTSTimeout, el control de comunicaciones genera un evento *OnComm,* y asigna a *CommEvent* el valor MSCOMM_ER_CTSTO.

En Visual Basic, si se llama a la propiedad sin asignarle ningún valor, se obtiene como respuesta el valor que tenía asignado dicha propiedad con anterioridad.

Esta propiedad es útil cuando se escribe una rutina de control de flujo RTS/CTS.

La sintaxis es la siguiente:

V. Basic [form.]MSComm.CTSTimeout[=milisegundos&]
V. C++ pMSComm->SetNumProperty("CTSTimeout")

DSRHolding

Nos informa del estado de la línea DSR. La señal DSR es enviada por un módem a su ordenador para indicar que está listo para operar *(Data Set Ready,* rnódem preparado). Los valores posibles para esta propiedad son TRUE, para indicar que la línea DSR está con valor alto (módem no preparado), y FALSE, para indicar lo contrario (módem preparado). Si la línea DSR está con valor alto durante un número de milisegundos mayor que el indicado por *DSRTimeout,* el control de comunicaciones genera un evento *OnComm* y asigna a *CommEvent* el valor MSCOMM_ER_ DSRTO.

La sintaxis es la siguiente:

V. Basic *[form.]MSComm.DSRHolding*
V. C++ *pMSComm->GetNumProperty("DSRHolding")*

DSRTimeout

Determina el número de milisegundos que hay que esperar una señal DSR antes de que el control de comunicaciones genere un evento *OnComm* y le asigne a *CommEvent* el valor MSCOMM_ER_DSRTO. Si la línea DSR está con valor alto durante un número de milisegundos mayor que el indicado por DSRTimeout, el control de comunicaciones genera un evento OnComm y asigna a CommEvent el valor MSCOMM_ER_DSRTO.

En Visual Basic, si se llama a la propiedad sin asignarle ningún valor, se obtiene como respuesta el valor que tenía asignado dicha propiedad con anterioridad.

Esta propiedad es útil cuando se escribe una rutina de control de flujo RTS/CTS.

La sintaxis es la siguiente:

V. Basic [form.]MSComm.DSRTimeout[=milisegundos&]
V. C++ pMSComm->GetNumPrpperty("DSRTimeout")
pMSComm->GetNumPrpperty("DSRTimeout",milesegundos)

DTREnable

Activa o desactiva la línea DTR durante la comunicación. La señal DTR es utilizada por el ordenador para indicarle a su módem que está listo para recibir datos. Los valores posibles de esta propiedad son TRUE, para activar la línea, y FALSE, para desactivarla. Este último es el valor por defecto.

La sintaxis es la siguiente:
V. Basic [form.]MSComm.DTREnable[={True|False}]
V. C++ pMSComm->GetNumProperty("DTREnable")
pMSComm->GetNumProperty("DTREnable",{TRUE|FALSE})

Handshaking

Determina el método de control de flujo que se va a utilizar durante la comunicación. Handshaking hace referencia al procedimiento de comunicaciones interno por el cual los datos son transferidos del puerto serie (y por tanto del módem) al buffer receptor. Cuando un carácter de datos llega al puerto serie, el dispositivo de comunicaciones tiene que moverlo al buffer receptor, de forma que pueda leerlo el programa de comunicaciones. Si los datos llegan al puerto serie más rápidamente de lo que el programa de comunicaciones es capaz de asimilar, al final se acabarán perdiendo datos, debido a una saturación del buffer receptor. Para evitar eso es para lo que se utilizan los procedimientos de control de flujo.

En Visual Basic, si se llama a la propiedad sin asignarle ningún valor, se obtiene como respuesta el valor que tenía asignado dicha propiedad con anterioridad.

La sintaxis es la siguiente:

V. Basic [form.]MSComm.Handshaking[=control_flujo%]

V. C++ pMSComm->GetNumProperty("Handshaking")
pMSComm->GetNumProperty("Handshaking",control_flujo)

Los valores válidos de control_flujo son los siguientes

MSCOMM_HANDSHAKE_NONE	0	Sin control de flujo (defecto)
MSCOMM_HANDSHAKE_XONXOFF	1	Control de flujoXon/Xoff
MSCOMM_HANDSHAKE_RTS	2	Control de flujo RTS/CTS
MSCOMM_HANDSHAKE_RTSXONXOFF	3	Ambos RTS/CTS y Xon/Xoff

InBufferCount

Nos informa del número de caracteres que han sido recibidos por el módem y que están esperando en el buffer receptor para ser leídos. El buffer receptor se puede limpiar poniendo a 0 la propiedad InBufferCount.

La sintaxis es la siguiente:

V. Basic [form.]MSComm.InBufferCount[=cuenta%]

V. C++ pMSComm->GetNumProperty("InBufferCount")
pMSComm->GetNumProperty("InBufferCount",cuenta)

InBufferSize

Determina el tamaño total en bytes del buffer receptor. El valor por defecto es de 1024 bytes.

En Visual Basic, si se llama a la propiedad sin asignarle ningún valor, se obtiene como respuesta el valor que tenía asignado dicha propiedad con anterioridad.

V. Basic [form.]MSComm.InBufferSize[=número_bytes%]

V. C++ pMSComm->GetNumProperty("InBufferSize")
pMSComm->GetNumProperty("InBufferSize",número_bytes)

Lee y elimina los caracteres contenidos en el buffer receptor. El número de caracteres que son leídos por Input viene fijado por la propiedad InputLen. Si se fija InputLen a 0, Input leerá el contenido completo del buffer receptor.

La sintaxis es la siguiente:

V. Basic [form.]MSComm.Input

V. C++ pMSComm->GetNumProperty("Input")

IpData apunta al buffer de datos para mantener la entrada de datos, y cdData es el tamaño del buffer en bytes. Este mensaje devolverá el número actual de bytes de la entrada de usuario.

Este ejemplo muestra cómo leer los datos en el buffer receptor en Visual Basic:

```
'Lee todos los datos
Comm1.InputLen=0
If Comnl.tnBuffercount Then
'Lee los datos
InString$=Comm1.Input
End If
```

InputLen

Determina el número de caracteres que se leen en el buffer receptor cada vez que se utiliza la propiedad Input. El valor por defecto de InputLen es 0, lo cual significa que el control de comunicación leerá el contenido entero del buffer receptor cada vez que se use Input.

En Visual Basic, si se llama a la propiedad sin asignarle ningún valor, se obtiene como respuesta el valor que tenía asignada dicha propiedad con anterioridad.

La sintaxis es la siguiente:

V. Basic [form.]MScomm.InputLen[=número_caracteres%]

V. C++ pMSComm->GetNumProperty("InputLen")
pMSComm->GetNumProperty("InputLen",número_caracteres)

Si no hay disponible en el buffer receptor el número de caracteres fijados por InputLen la propiedad Input devolverá una cadena de longitud cero (""). Eso quiere decir que en el caso de utilizar esta propiedad con un número distinto de O, se deberla comprobar con InBufferCount si el buffer contiene el minimo número de caracteres requeridos antes de hacer un Input.

La propiedad InputLen es útil cuando se leen datos formateados en bloques de caracteres de longitud fija.

El siguiente ejemplo muestra cómo leer 10 caracteres de datos con Visual Basic:

```
'Determina 10 caracteres por bloque Comm1.InputLen=10 .
'Lee los datos
CommData$Comm1.Input
```

Interval

Esta propiedad sólo es necesaria en aplicaciones que se ejecuten en el entorno Windows 3.0. Interval fija el intervalo en milisegundos con el que el control de comunicaciones sondeara el puerto serie para ver si se han recibido nuevos datos. El valor por defecto de la propiedad Interval es 1000(1 segundo).

La sintaxis es la siguientes:

V. Basic [form.]MSComm.Interval[=milisegundos&]

V. C++ pMSComm->GetNumProperty("Interval")
pMSComm->GetNumProperty("Interval",milisegundos)

NullDiscard

Determina si los caracteres nulos son transferidos o no desde el puerto serie al buffer receptor. Se le llama carácter nulo a aquel que tiene el valor ASCII O. Los valores posibles para esta propiedad son: TRUE, para indicar que los caracteres nulos no son transferidos del puerto al buffer receptor, y FALSE, para indicar que los caracteres nulos son transferidos del puerto al buffer receptor. Éste es el valor por defecto.

La sintaxis es la siguiente:

V. Basic [form.]MSComm.NullDiscard[=TRUE|False]
V. C++ pMSComm->GetNumProperty("NullDiscard")

OutBufferCount

Nos informa del número de caracteres que están esperando en el buffer transmisor para ser transmitidos. Esta propiedad puede ser utilizada también para limpiar el buffer transmisor (dándole el valor 0).

La sintaxis es la siguiente:

V. Basic [form.]MSComm.OutBufferCount[=0]

V. C++ pMSComm->GetNumProperty("OutBufferCount")

OutBufferSize

Determina el tamaño, en caracteres, del buffer transmisor. El valor por defecto es de 512 bytes.

En Visual Basic, si se llama a la propiedad sm asignarle ningún valor, se obtiene como respuesta el valor que tenía asignado dicha propiedad con anterioridad.

La sintaxis es la siguiente:
V. C++ pMSComm->GetNumProperty("OutBufferSize")

V. Basic [form.]MSComm.OutBufferSize[=Número_bytes%]

Output

Escribe una cadena de caracteres en el buffer transmisor.

La sintaxis es la siguiente:

V. Basic [form.]MSComm.Output[=Cadena_Salida$]
V. C++ pMSComm->SetStrProperty("Output",Cadena_salida)

La sintaxis de Visual C++ sólo trabajará para cadenas que no contengan caracteres nulos (ASCII 0). Para cadenas que contengan caracteres nulos se debe usar la función ComOutput o bien se debe emplear la siguiente sintaxis:

#define UM_OUTPUT WN_USER+0x0B01
pMSComm->SendMessage(UM_OUTPUT,lpData,cdData)

lpData apunta a la cadena que está siendo enviada, y cbData es la longitud de la cadena en bytes. Este mensaje devolverá el número actual de bytes enviados.

ParityReplace

Determina el carácter que vas reemplazar a los caracteres dados como inválidos por tener error de paridad. Por defecto, el control de comunicaciones utiliza un carácter de interrogación (?). Si se fija la propiedad ParityReplace a una cadena vacía ("") se inhabilita la comprobación de paridad.

En Visual Basic, si se llama a la propiedad sin asignarle ningún valor, se obtiene como respuesta el valor que tenía asignado dicha propiedad con anterioridad.

La sintaxis es la siguiente:

V. Basic [form.]MSComm.ParityRepleca[=caracter$]

PortOpen

Abre o cierra el puerto de comunicaciones definido anteriormente con la propiedad ComPort.

En Visual Basic, si se llama a la propiedad sin asignarle ningún valor, se obtiene como respuesta el valor que tenía asignado dicha propiedad con anterioridad.

La sintaxis es la siguiente:

V. Basic [form.]MSComm.PortOpen[={True|alse}]

V. C++ pMSComm->GetNumProperty("PortOpen")

Para abrir el puerto de comunicaciones se le da a PortOpen el valor TRUE, y para cerrarlo se le da el valor FALSE. Cuando se cierra el puerto, también se limpia los buffer receptor y transmisor.

Si se intenta abrir un puerto cuyo valor CommPort no es válido, el control de comunicaciones generará el error 68 (Device Unavailable, dispositivo no disponible).

RThreshold

Determina el valor umbral de caracteres recibidos para generar un evento OnComm. Cuando el número de caracteres recibidos alcanza el valor RThreshold, el control de comunicaciones genera un evento OnComm y le da a CommEvent el valor MSCOMM_EV_RECEIVE.

Si, por ejemplo, se fija RThreshold a 1, cada vez que se reciba un carácter en el buifer receptor, el control de comunicaciones generará un evento.

En Visual Basic, si se llama a la propiedad sin asignarle ningún valor, se obtiene como el valor que tenía asignado dicha propiedad con anterioridad.

La sintaxis es la siguiente:

V. Basic [form.]MSComm.RThreshold[=numeros_caracteres%]
V. C++ pMSComm->GetNumProperty("RThreshold")

Si se fija la propiedad RThreshold a 0 (por defecto), se inhabilita la generación de eventos OnComm por recepción de caracteres.

RTSEnable

Activa o desactiva la señal RTS. La señal RTS es enviada por ordenador al módem para pedirle permiso para transmitir datos. Una vez hecho eso, y antes de empezar a transmitir datos, el ordenador debe recibir la señal CTS. La línea RTS es utilizada en el control de flujo hardware RTS/CST.

Los valores posibles de esta propiedad son TRUE, para habilitaría línea RTS, y FALSE, para inhabilitarla (valor por defecto).

La sintaxis es la siguiente:

V. Basic [form.]MSComm.RTSEnable[=True|False]

V. C++ pMSComm->GetNumProperty("RTSEnable")

Setting

Determina los parámetros de la comunicación, velocidad de transmisión, tipo de paridad, bits de datos y bit de parada.

En Visual Basic; si se llama ala propiedad sin asignarle ningún valor, se obtiene como respuesta el y valor que tenía asignado dicha propiedad con anterioridad.

La sintaxis es la siguiente:

V. Basic [form.]MSComm.Setting[=Setting$]

V. C++ pMSComm->GetNumProperty("Setting")

Parámetros tiene el formato: "BBBB,P,D,S". Donde BBBB es la velocidad en bps, P es la paridad, D es el número de bits de datos y 5 es el número de bits de parada. El valor por defecto de este parámetro es "9600,N,8,l". El control de comunicaciones admite unos valores predeterminados para cada uno de esos parámetros. Si se indica un valor no válido, cuando se intente abrir el puerto, se generará el error 380 (Invalid Property value, valor de la propiedad no válido).

4.2 CONTROL DE FLUJO DE LA INFORMACION

Las técnicas de control de flujo (*flow control*) compensan la diferencia de velocidad existente entre la llegada y la salida de datos de un dispositivo. Por ejemplo, imagínese un módem que utiliza la técnica de compresión; el módem está conectado con el otro extremo a 9600 bps, pero como está comprimiendo la información con una relación 2:1, el terminal le envía 19600 bps. Supóngase que en un momento dado, la información que recibe el terminal no es tan susceptible de ser comprimida como la anterior, y por tanto no consigue llegar a la relación de compresión 2:1. En este momento, el módem estaría recibiendo más información del terminal de la que puede transmitir. Si no existieran técnicas de control de flujo, toda esa información que recibe el módem de más se perdería.

Todos los módems que utilizan técnicas de detección y corrección de error o técnicas de compresión están forzados, por un lado, a incorporar una memoria intermedia (buffer), y por otro, a utilizar técnicas de control de flujo. El control de flujo es la técnica que previene que se sature la memoria intermedia y que se pierdan datos.

El control de flujo fija dos niveles de ocupación de la memoria intermedia: cuando la memoria intermedia alcanza el nivel alto, el módem le indica al terminal que no siga enviando información, y cuando el nivel de ocupación de la memoria intermedia alcanza el nivel bajo, el módem le indica al terminal que reanude el envío de información.

El control de flujo no sólo se produce en la dirección terminal-módem, sino que también se utiliza en la dirección contraria. El software de comunicaciones que opera en el terminal dispone de un área limitada para guardar los datos que recibe del módem. Si, por ejemplo, el terminal le va enviando los datos que recibe a un periférico, como puede ser una impresora, esos datos pueden llegar más de prisa de lo que la impresora es capaz de imprimirlos.

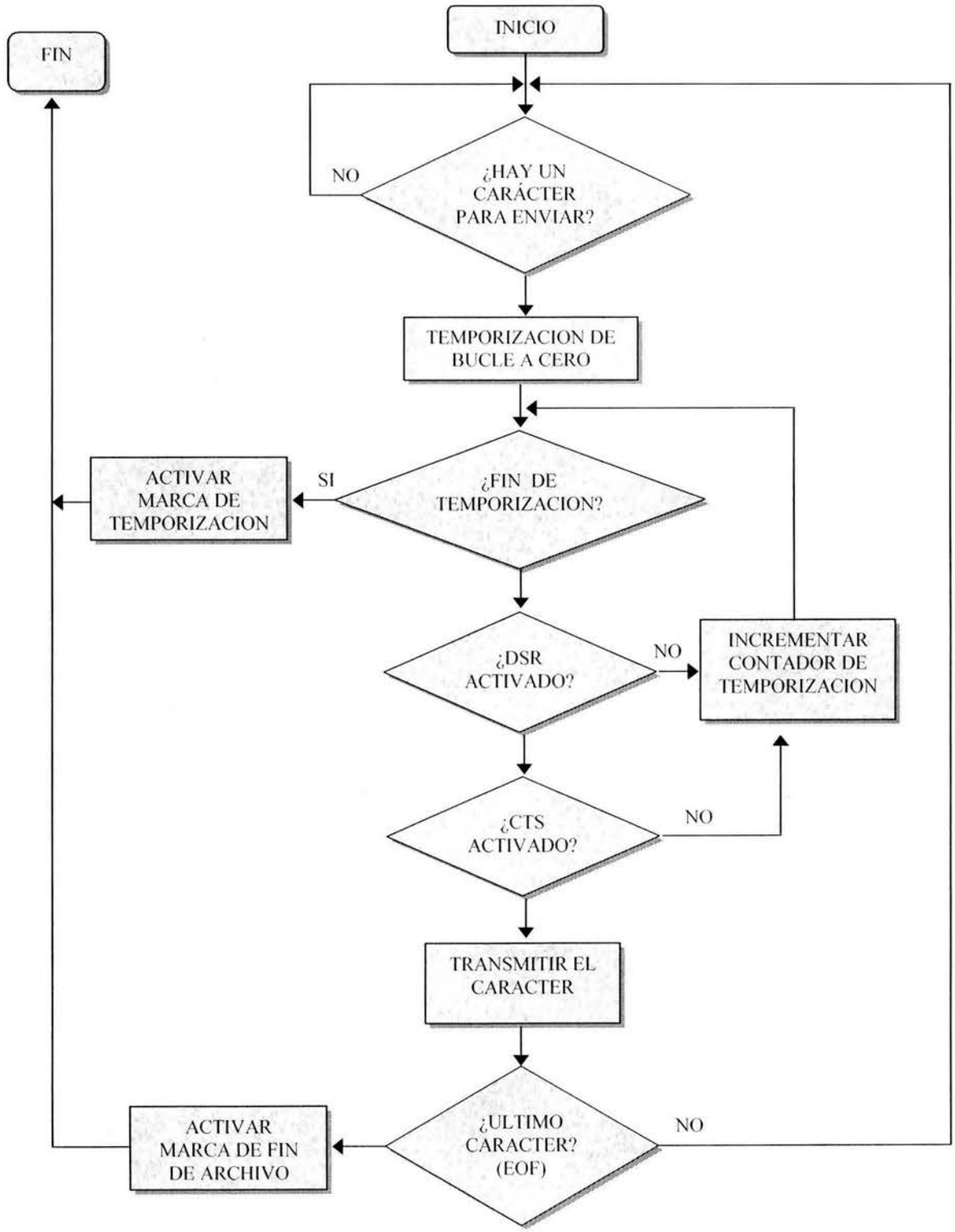
Las técnicas básicas de control de flujo son las siguientes:

- ⇨ RTS/CTS
- ⇨ XON/XOFF
- ⇨ ENQ/ACK

RTS/CTS

Cuando un ordenador se dispone a transmitir datos, le envía al módem una señal de petición de envío conocida como RTS (Request to Send). Esta señal consiste en poner a 1 el contacto 4 de la interfaz RS232. Sí el módem está listo para transmitir, responde con la señal de listo para enviar, CTS (Clear to Send). Esta señal consiste en activar el contacto 5 de la interfaz RS232. El terminal no transmitirá datos al módem si no está activa la señal CTS. Eso quiere decir que el módem puede controlar el flujo de datos del terminal simplemente activando o desactivando la señal CTS. A este sistema de control de flujo también se le conoce por el nombre de control de flujo hardware (*Hardware Flow Control* Fig. 4.1), debido a que el control se realiza por medio de un cable físico que une al ordenador con el módem.

FIG. 4.1 ORGANIGRAMA DEL CONTROL DE FLUJO RTS/CTS (*LADO TRANSMISOR*)

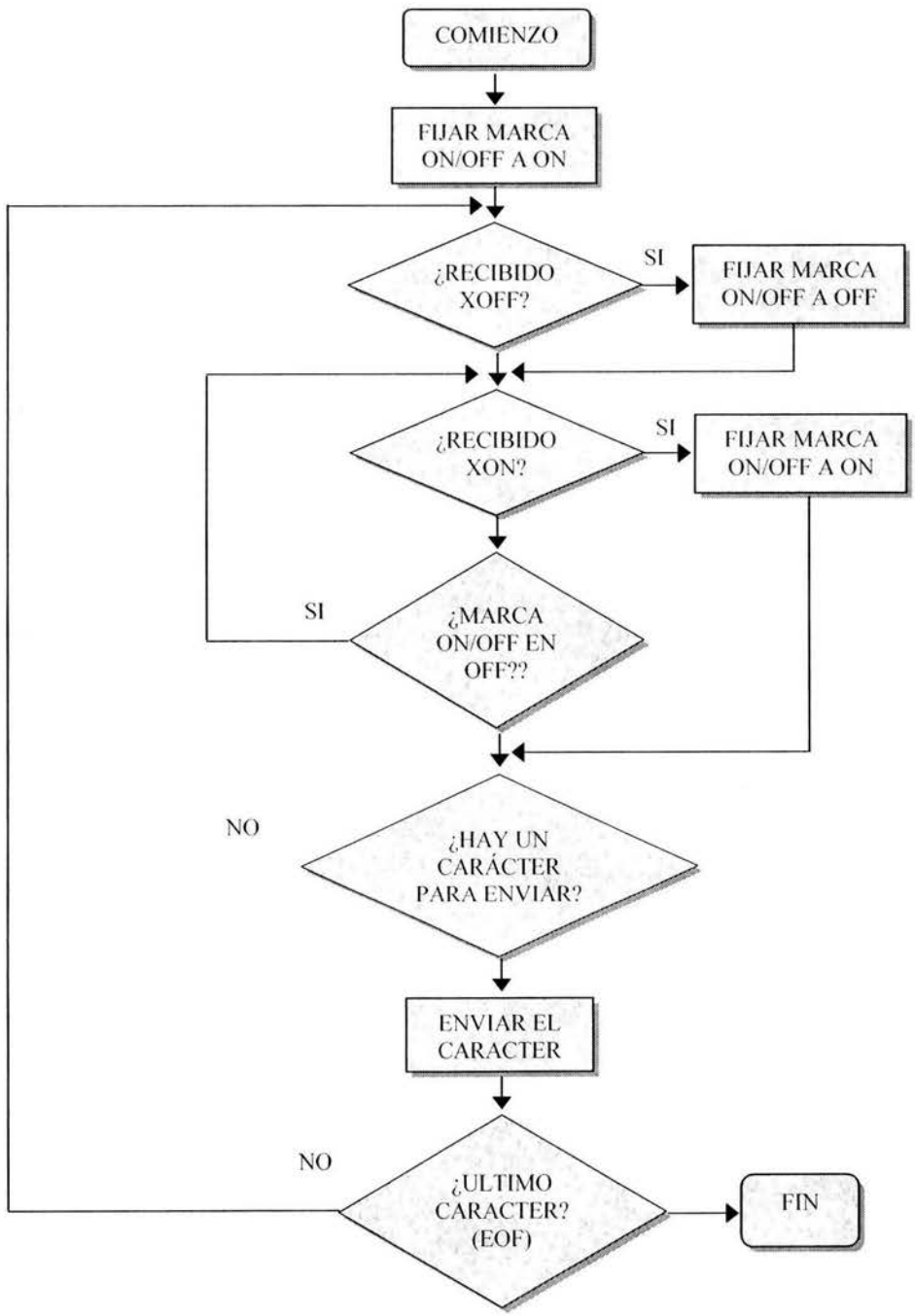


XON/XOFF

El sistema Xon/Xoff (Fig. 4.2) se basa en la existencia de dos caracteres de control, XON y XOFF, los cuales son utilizados por el módem para indicarle al terminal que detenga o reanude el envío de datos. Debido a que los caracteres XON y XOFF son generados mediante el software, a este procedimiento también se le conoce como control de flujo software (*Software Flow* Control o *Software Handshaking*).

El carácter de control XOFF, utilizado por el módem para suspender el flujo de datos, se corresponde con el carácter ASCII 19 (Ctrl-S), y también se conoce como carácter DC3 (*Device Control 3*, control de dispositivo 3). Por su parte, el carácter de control XON permite reanudar el envío de datos. Este carácter se corresponde con el código ASCII 17 (Ctrl-Q), y también se conoce como carácter DCl (*Device Control 1*, control de dispositivo 1).

FIG. 4.2 ORGANIGRAMA DEL CONTROL DE FLUJO XON/XOFF (*LADO TRANSMISOR*)



El control de flujo software presenta varios inconvenientes. El primero es que enviar señales XON y XOFF consume tiempo; esto es, mientras se envían las señales XON o XOFF no se puede enviar datos, lo cual disminuye el rendimiento. El segundo es que si los caracteres ASCII 17 o 19 de los caracteres de control XOFF y XON aparecen en los datos, el software tendrá que indicar de alguna forma que se trata de información de datos y no de los caracteres de control de flujo, lo cual implica más información redundante.

ENQ/ACK

El método de control de flujo ENQ/ACK (*enquire/acknowledge*, petición/aceptación) es utilizado por ciertos ordenadores de Hewlett-Packard. Este método consiste en que el terminal, antes de transmitirle datos al módem, le envía un mensaje ENQ (petición), a lo que el módem debe responder con un mensaje ACK (aceptación). Cada vez que el terminal recibe un mensaje ACK en respuesta a su mensaje ENQ, transmite un bloque de datos de aproximadamente 4000 caracteres. El módem controla el flujo con el mensaje ACK.

4.3. DESARROLLO DEL SOFTWARE DEL SISTEMA DE ADQUISICIÓN DE DATOS

DESCRIPCION GENERAL DEL PROGRAMA

El programa para el Sistema de Adquisición de Datos esta diseñado para poder recibir y transmitir datos a través del puerto serial.
Las características del programa ejecutable son las siguientes:

1.- Poder conectar o desconectar el puerto mediante un botón de opción.
2.- Seleccionar uno de los 2 buffers del circuito, (El Buffer1 nos proporciona los datos provenientes del convertidor A/D y el Buffer2 es de propósito general).
3.- Leer datos desde el puerto serial de manera continua, indicándonos en un cuadro de texto el valor del dato que se esta leyendo.
4.-Si esta seleccionado el Buffer1, se nos muestra una grafica del comportamiento de la temperatura ambiente o de algún objeto que se este sensando, con respecto al tiempo. Además se puede observar una barra que nos indica de manera visual el valor de la temperatura.

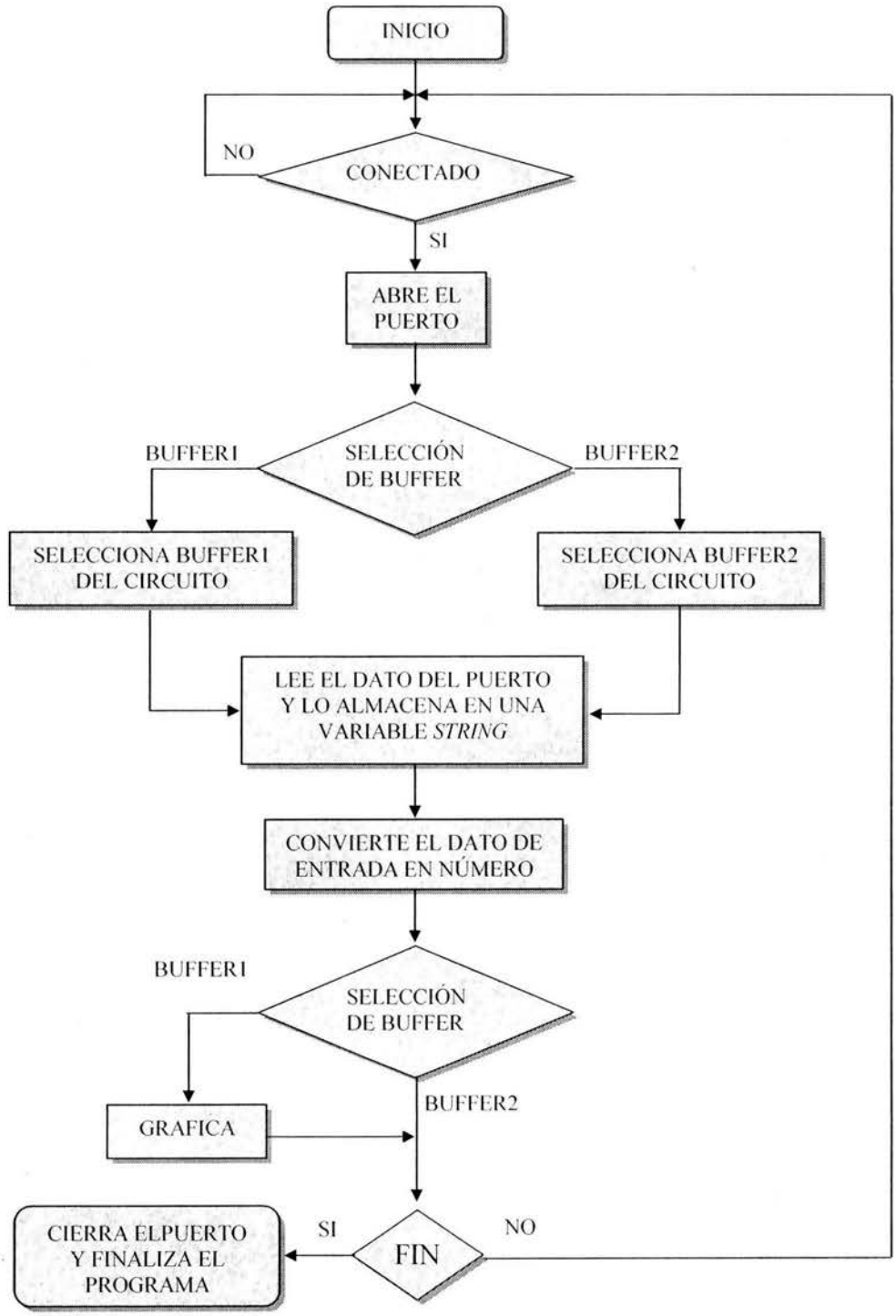
ESTRUCTURA INTERNA DEL PROGRAMA

El programa esta estructurado de la siguiente forma:

1.- Se configura el puerto de comunicaciones mediante las propiedades del control MSCOMM1. Los parámetros los podemos ver en la figura 4.3.
2.- Checa la opción de conectado, si esta habilitada se abre el puerto y se inicia el ciclo para la transmisión y recepción de datos.
3.- Si esta seleccionada la opción del Buffer1, se transmite el carácter ASCII(0), para habilitar el Buffer1 del circuito, y si esta habilitado el Buffer2, se transmite el carácter ASCII(1), para habilitar el Buffer2 del circuito.
4.- Posteriormente se envía un pulso de reloj a través del pin RTS del puerto serial, el cual, después de pasar por una conversión de voltaje a niveles TTL, es recibido por el pin 23 (TBRL) de la UART CDP6402, el cual utiliza este pulso para la Carga del Registro Transmisor. Cabe recordar que un nivel bajo en TBRL coloca el dato presente en las entradas TBR1 – TBR8, en el Registro del Buffer Transmisor. Una transición de nivel bajo a alto en TBRL envía el dato al Registro Transmisor.
5.-Después de esto, el programa lee le dato del puerto serial, y lo almacena en una variable para ser procesado posteriormente.
6.-Debido a las limitaciones de Visual Basic, y a que Windows tiene algunos caracteres reservados para uso especial y otros no pueden ser mostrados por este, el dato de entrada (que es un dato en formato de caracteres) es procesado matemáticamente mediante comparaciones binarias, para ajustar el dato a su valor real y convertirlo a su valor numérico de manera correcta.

7.- Si el Buffer2 es el que esta seleccionado, se muestra una gráfica con los valores que se van obteniendo, así como un termómetro virtual.
8.- El ciclo de lectura se repite de manera continua, hasta que se selecciona la opción 'Estado de Conexión DESCONECTADO' o se finaliza el programa.

DIAGRAMA DE FLUJO DEL PROGRAMA



CONFIGURACION DEL PUERTO SERIAL Y DEL CONTROL MSCOMM1.VBX

Los parámetros que utilizamos para la comunicación del PC con nuestro sistema de adquisición de datos son velocidad de transmisión **9,600 bps**, **sin paridad, 8 bits de datos y 1 de parada**, por ser estos los mas utilizados.

El puerto serial utilizado es el **COM1**, por ser el que tienen la mayoría de los PC's, aunque se puede seleccionar otro, mediante algunas modificaciones muy sencillas en el programa.

Las opciones **DTREnable** y **RTSEnable**, están habilitadas debido a que utilizamos el pin RTS para funciones de control del circuito. **EOF** (End of File, Fin de Archivo) esta inhabilitada, ya que no estamos recibiendo archivos, sino datos de manera continua. El modo de entrada **InputMode** es el **InputModeText** (Entrada en Modo Texto) ya que los datos recibidos se leen como caracteres, los cuales son mas fáciles de manejar que si se reciben en forma binaria. Además, el parámetro de **NullDiscard** (Descarga de caracteres nulos), esta inhabilitado para evitar errores de lectura. El valor de **InputLen** es 0 para leer todo el contenido del buffer receptor del puerto. Los valores de **InBufferSize**, **OutBufferSize** y **ParityReplace** son asignados por default. **RThreshold** y **SThreshold** tienen el valor 0 debido a que no utilizamos la opción de generar un evento al recibir o transmitir un determinado numero de caracteres.

Por último nuestro proyecto utilizamos el protocolo XON/XOFF, ya que nos permite un flujo de información mas sencillo y continuo. A continuación se describe el flujo que sigue nuestro programa, referido al protocolo XON/XOFF.

Comienzo	
Fijar marca ON/OFF a ON	
Recibido XOFF? Si o No?	**No**
Recibido XON? Si o No?	**No**
Marca ON/OFF en OFF? Si o No?	**No**
¿Hay un caracter para enviar?	**Si**
Enviar el caracter	
¿Ultimo carácter? EOF	**No**
Comienzo	

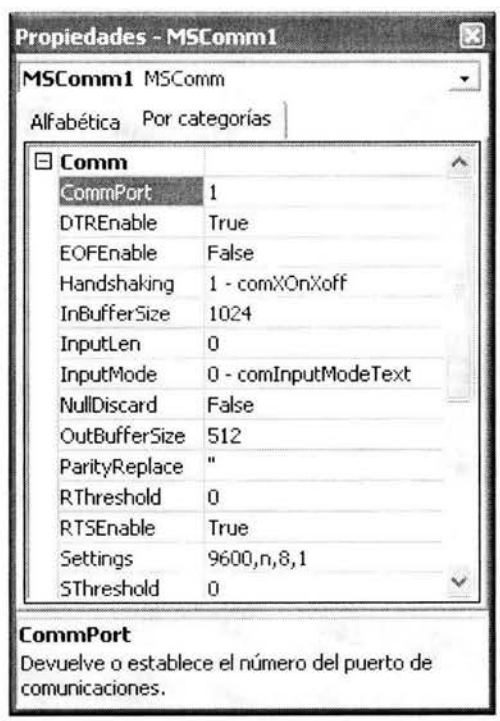


FIG. 4.3 PROPIEDADES DEL CONTROL MSCOMM1

PROGRAMA FUENTE DEL SISTEMA DE ADQUISICIÓN DE DATOS

```
'Define las variables que se van a utilizar en el programa
Public X As Integer, Y As Integer, X1 As Integer, X2 As Integer, i As Integer, YPoint As Integer
Public numero As Integer
Public cadena As String

Private Sub Form_Load()
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
"ABRE EL PUERTO Y SELECCIONA EL NUMERO DE PUERTO
'*********************************************************************************
'Define el numero de puerto
MSComm1.CommPort = 1
'Abre el puerto
MSComm1.PortOpen = True
'Habilita el Buffer2 del circuito (Dato digital)
lblOut.Caption = "1"
MSComm1.Output = Chr(1)
End Sub

Private Sub optConexion_Click(Index As Integer)
'++++++++++++++++++++++++++++++++++++++++++++++
'RUTINA PARA CONECTAR Y DESCONECTAR EL PUERTO
'**********************************************
'Abre el puerto si esta seleccionada la opción  conectado
If optConexion(0).Value = True Then

'Conecta el puerto
   If MSComm1.PortOpen = False Then
   MSComm1.PortOpen = True
   End If
   Image1.Visible = True
   lblConectado.Visible = True

'Habilita Timer1 para iniciar ciclo de Tx y Rx
   Timer1.Enabled = True
    Else

'Cierra el puerto si esta seleccionada la opción desconectado
    Image1.Visible = False
    lblConectado.Visible = False
      Timer1.Enabled = False
      Timer2.Enabled = False
      MSComm1.InBufferCount = 0
      MSComm1.PortOpen = False
      End If
End Sub

Private Sub optBuffer_Click(Index As Integer)
'************************************
'Selecciona el Buffer1 del circuito
MSComm1.InBufferCount = 0
If MSComm1.PortOpen = True Then
```

```
lblOut.Caption = "0"
MSComm1.Output = Chr(0)
Timer1.Enabled = True
Else
inutil = 0
End If
End Sub
```

```
Private Sub optBuffer2_Click()
'****************************************
'Selecciona el Buffer2 del circuito
MSComm1.InBufferCount = 0
If MSComm1.PortOpen = True Then
lblOut.Caption = "1"
MSComm1.Output = Chr(1)
Timer1.Enabled = True
Else
inutil = 0
End If
End Sub
```

```
Private Sub Timer1_Timer()
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'RUTINA PARA HACER EL RELOJ DE RTS
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'Cambia circulo1(shape1) a color Rojo
   Shape1(0).BackColor = vbGreen
'Habilita Timer 2 y pone la línea RTS a "0"
   MSComm1.RTSEnable = False
   Timer2.Enabled = True
   End Sub
```

```
Private Sub Timer2_Timer()
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'RUTINA PARA HACER EL RELOJ DE RTS  Y EMPEZAR EL CICLO DE LECTURA Y/O 'ESCRITURA
(2)
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'Cambia circulo1(shape1) a color Verde
   Shape1(0).BackColor = vbRed
'Deshabilita Timer 1 y pone la linea RTS a "1"
      MSComm1.RTSEnable = True
      Timer1.Enabled = False
'Llama al ciclo de lectura y/o escritura
Ciclo
End Sub
```

```
Function Ciclo()
Dim h As Integer
'++++++++++++++++++++++++++++++++++++++++++++++++
'CICLO PARA LEER Y ESCRIBIR DATOS
'------------------------------------------
'*********************************************************************************
'Checa que buffer esta seleccionado
'*********************************************************************************
```

```
'Si el Buffer2 esta seleccionado, empieza rutina para lectura
If optBuffer(0).Value = False Then
'Habilita el buffer2 del cto

'-Hace Invisible lo que no se esta sensando
'-----------------------------------------
   For d = 0 To 40
   LineGrap(d).Visible = False
   Next d
   lblIndicador(0).Visible = False
   txtNumero2.Visible = False
   txtEntrada2.Visible = False
   Picture1.Visible = False
   pbrTermometro.Visible = False
   For h = 0 To 24
   lblTemp(h).Visible = False
   Next h
   'Reinicia la variable i a 0 para la gráfica
   i = 0

'+Hace Visible lo que se esta sensando
'+++++++++++++++++++++++++++++++++++++++++++++++++
   lblIndicador(1).Visible = True
   txtNumero.Visible = True
   txtEntrada.Visible = True
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'Lee el Dato del Puerto Serie y lo pone en la variable 'cadena'
'*************************************************************
   lblIn.Caption = MSComm1.InBufferCount
   cadena = MSComm1.Input
   txtEntrada.Text = cadena
'Convierte el dato a valor numérico
   Conversion1
   X1 = X
   txtNumero.Text = X1

Else

'Si el Buffer1 esta seleccionado, empieza rutina para lectura del convertidor A/D
'-Hace Invisible lo que no se esta sensando
'-----------------------------------------
   lblIndicador(1).Visible = False
   txtNumero.Visible = False
   txtEntrada.Visible = False
'+Hace Visible lo que se esta sensando
'+++++++++++++++++++++++++++++++++++++++++++++++
   lblIndicador(0).Visible = True
   txtNumero2.Visible = True
   txtEntrada2.Visible = True
   Picture1.Visible = True
   pbrTermometro.Visible = True

   For h = 0 To 24
   lblTemp(h).Visible = True
   Next h
```

```
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'Lee el Dato del Puerto Serie y lo pone en la variable 'cadena'
'**********************************************************
   lblIn.Caption = MSComm1.InBufferCount
   cadena = MSComm1.Input
      txtEntrada2.Text = cadena

'Convierte el dato a valor numérico
   Conversion1
   X2 = X
   txtNumero2.Text = X2

'Llama a la función para graficar
   Termometro

End If

'Vuelve a iniciar el ciclo desde el reloj RTS
  Timer2.Enabled = False
  Timer1.Enabled = True
End Function
_____________________________________________________________________________
Function Conversion1()

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'RUTINA PARA CONVERTIR EL VALOR DE ENTRADA EN NUMERO (1)
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'Definición de variables
Dim caso As Integer, inutil As Integer
Dim caracter1 As String, caracter2 As String
Dim arroba As String, efe As String
arroba = Chr(64)
efe = Chr(102)

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'RUTINA para extraer 2 caracteres del Dato de entrada
'****************************************************
   caracter1 = Left(cadena, 1)
   caracter2 = Right(cadena, 1)
'Checa que no haya caracteres menores que @ = 64
If StrComp(caracter2, arroba, vbBinaryCompare) = -1 Then
caracter2 = Chr(255)
Else
inutil = 0
End If
'Checa que no haya caracteres mayores que f = 102 y que caracter2 no sea menor que caracter1
If StrComp(caracter1, efe, vbBinaryCompare) = 1 And StrComp(caracter2, caracter1, vbBinaryCompare) = -1
Then
caracter2 = Chr(255)
Else
inutil = 0
End If
```

```
'Checa que no haya caracteres nulos
If StrComp(caracter1, 0, vbBinaryCompare) = 1 And StrComp(caracter2, 0, vbBinaryCompare) = 1 Then

        'RUTINA PARA CORREGIR ERRORES EN EL DATO DE ENTRADA
        caso = StrComp(caracter1, caracter2, vbBinaryCompare)

        'Conversion del caracter a su valor numérico ASCII
        Select Case caso
              Case 0
              numero = Asc(caracter1)
             ConvNumero
              Case 1
              numero = Asc(caracter2)
             ConvNumero
              Case -1
              numero = Asc(caracter1)
              ConvNumero
     End Select

Else
   inutil = 0
End If

End Function
```

```
Function ConvNumero()

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'FUNCION PARA CONVERTIR EL VALOR DE ENTRADA EN NUMERO (2)
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
If numero < 128 Then
        X = numero - 64
     Else
        X = numero - 128
        End If
End Function
```

```
Function Termometro()
'Definición de variables
Dim d As Integer

'Control de la Barra de progreso que se usa como termometro
If X2 > 0 Then
   pbrTermometro.Value = X2 * 2
Else
   pbrTermometro.Value = 0
End If

'Control de los puntos para graficar
LineGrap(0).Y1 = 3355
LineGrap(0).Y2 = 3355 - (24 * X2)
If i < 40 Then
   i = i + 1
```

```
    YPoint = 3355 - (24 * X2)
    LineGrap(i).Y1 = LineGrap(i - 1).Y2
    LineGrap(i).Y2 = YPoint
    LineGrap(i).Visible = True
Else
    For d = 0 To 40
    LineGrap(d).Visible = False
Next d
    i = 0
    LineGrap(0).Y1 = YPoint
    LineGrap(0).Y2 = 3355 - (24 * X2)
    i = i + 1
    YPoint = 3355 - (24 * X2)
    LineGrap(i).Y1 = LineGrap(i - 1).Y2
    LineGrap(i).Y2 = YPoint
    LineGrap(i).Visible = True
End If
End Function

_____________________________________________________________________________
Private Sub cmdFin_Click()
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'FINALIZA EL PROGRAMA Y CIERRA EL PUERTO
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
If MSComm1.PortOpen = True Then
    MSComm1.PortOpen = False
End If
    Unload frmPuertoserie
    End
End Sub
```

'FIN DEL PROGRAMA

****

Nota: El dato que se lee del puerto serial se recibe como carácter ASCII, pero WINDOWS tiene algunos caracteres del código ASCII reservados para uso especial, por lo que el carácter de información que se recibe debe ser procesado para obtener el valor numérico que se transmitió de lado del Sistema de Adquisición de Datos.

Cabe mencionar que este programa, aunque es muy sencillo, tiene la flexibilidad para poder ser mejorado y tener un sistema mucho más potente y avanzado.

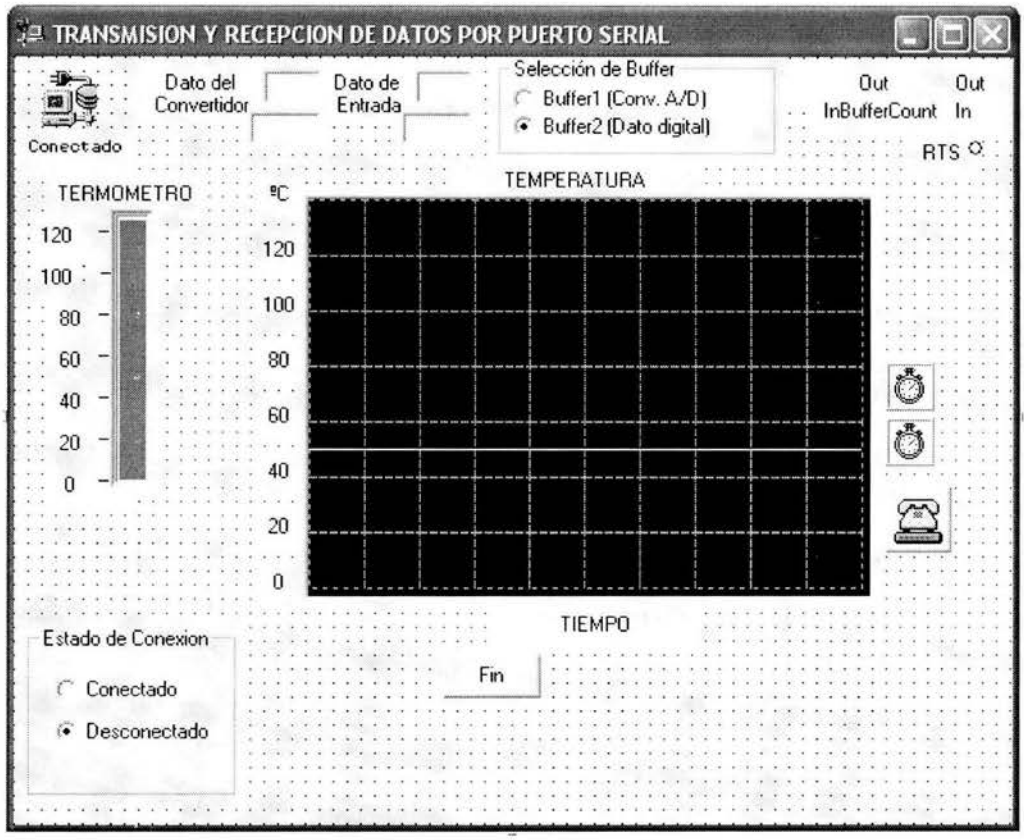
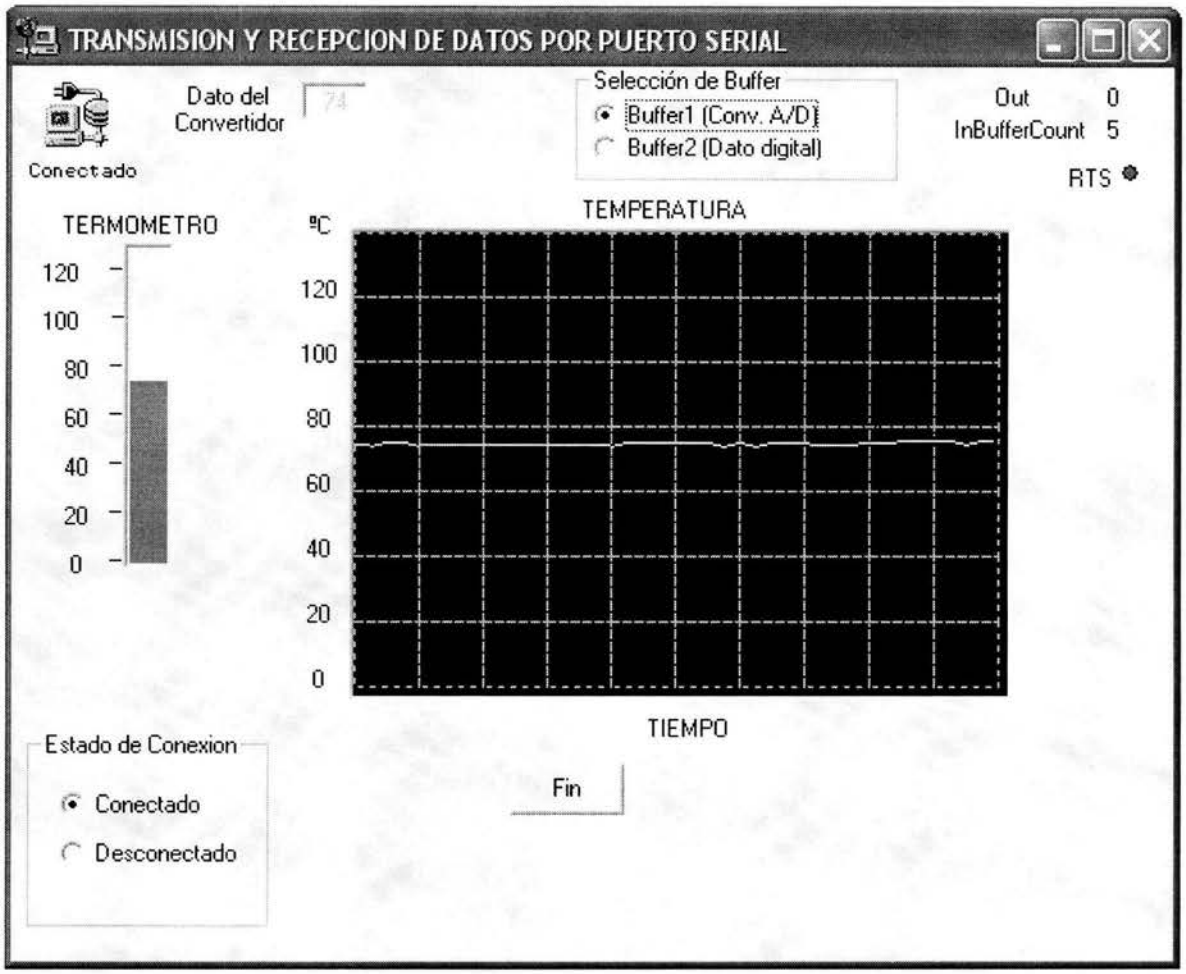


Tabla de código de caracteres ASCII (Multilingüe)

0		32		64	@	96	`	128	Ç	160	á	192	└	224	Ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
4	♦	36	Ş	68	D	100	d	132	ä	164	ñ	196	─	228	õ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	Þ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	þ
9	○	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13	♪	45	_	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	☼	47	/	79	O	111	o	143	Å	175	»	207	☼	239	´
16	►	48	0	80	P	112	p	144	É	176	░	208	ð	240	-
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	↕	50	2	82	R	114	r	146	ff	178	▓	210	Ê	242	=
19	‼	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	§	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	▬	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	↑	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	∟	60	<	92	\	124	¦	156	£	188	╝	220	▄	252	³
29	↔	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	▲	62	>	94	^	126	~	158	×	190	¥	222	ì	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	

VISUALIZACION DEL PROGRAMA EJECUTANDOSE



CONCLUSIONES

En este proyecto de tesis se logró diseñar un sistema de adquisición de datos que apoye directamente al "Prototipo Aragonés TIM", con la finalidad de crear una estación meteorológica, que preserve los equipos de medición y operación de dicho prototipo.

Uno de los problemas al cual nos enfrentamos fue la compatibilidad de nuestro sistema de adquisición de datos con computadoras con microprocesadores de frecuencias diferentes, esto se soluciono por medio de una programación exacta y realizando pruebas en equipos con características diferentes. Cabe mencionar que la programación y el manejo del puerto serie es uno de los más complicados pero uno de los más utilizados y seguros para garantizar datos de entrada y salida.

Por ultimo, este sistema de adquisición de datos es muy económico, con altas posibilidades de desarrollo a nivel de adquisición, así como de control y además este sistema puede ampliarse a través de la mejora continua del software y de la ampliación del hardware, así como la utilización de componentes electrónicos de mayor calidad, capacidad y velocidad.

BIBLIOGRAFÍA

1. "Aprenda Visual Basic 6.0 como si estuviera en primero". García de Jalón Javier Colección: "Aprenda..., como si estuviera en primero". Editorial Universidad de Navarra. España 1999.

2. "Circuitos electrónicos y sus aplicaciones". Grob. Ed. Mc Graw Hill, México, 1988.

3. "Diseño de circuitos y sistemas eléctricos". Dede Enrique J. Editores Boixareu. Barcelona – México.

4. "El Libro de las Comunicaciones del PC". José A. Carballar. Editorial RA-MA, Madrid, 1996.

5. "Fundamentos de programación". Peñalosa Romero, Ernesto. Edit. UNAM – ARAGÓN. México 1994.

6. Manual ECG Master Replacement Guide. Philip 19ª. Edición

7. "Visual Basic 4". Mark Steven Herman, Editorial Prentice Hall, México, 1996.

8. *http://electronica.eia.edu.co*

9. *http://paidoteca.dgsca.uman.mx*

10. *www.atmosfera.cl*

11. *www.beyondlogic.com*

12. *www.ecgproducts.com*

13. *www.steren.com.mx*