Escuela Politécnica Superior
Universidad de Huelva

Departamento de Ing. Electrónica,
Sistemas Informáticos y Automática

Memoria Externa y Puerto Serie

TERCER CURSO. INFORMÁTICA INDUSTRIAL II

Manuel Sánchez Raya
Versión 0.7
3 de Octubre de 2002

ÍNDICE

BIBLIOGRAFÍA:

Apuntes de Ingeniería Técnica Industrial. Universidad de Málaga.

Microcontroladores MCS-51 y MCS-251. José Matas Alcalá, Rafael Ramón Ramos Lara

1.- Conexión con Memoria Externa

1.1.- Introducción.

Muchas aplicaciones realizadas con microcontroladores de las familias MCS-5 1 y MCS-25 1 se puede resolver con versiones que disponen de memoria interna de programa, por lo que no es necesario, en estas versiones, el uso de circuito integrados de memoria adicionales. Las versiones con memoria interna EPROM son Útiles para elaborar prototipos, pero más aún son viables las versiones CON memoria EPROM del tipo OTP, One-Time Programmable, y las versiones con memoria EEPROM tipo flash de Atmel, ya que las primeras carecen de la ventana de cuarzo para borrar las memoria EPROM, lo que simplifica su encapsulado y reduce considerablemente su coste, y las segundas tienen la posibilidad de reprogramarlas un elevado numero de veces sin disponer de programador especial.

De toda maneras, según la aplicación, puede ser necesario ampliar la capacidad de memoria de lo microcontroladores, o emplear versiones que carezcan de memoria interna; en estos casos resulta imprescindible usar circuitos integrados de memoria externa para resolver la aplicación. Entonces, es importante conocer correctamente la forma de conectar los circuitos de memoria al microcontrolador: saber determinar si las memorias son compatibles a nivel temporal con el microcontrolador.

1.2.- Memorias semiconductoras.

En un sistema basado en microprocesador o microcontrolador es necesario conocer los distintos tipos de memoria que se pueden utilizar, así como la formar de realizar el acceso a éstas. Las memorias aquí más se emplean son memorias no volátiles de sólo lectura, ROM, destinadas a almacenar el código de programa generado en la aplicación. Estas memorias mantienen la información almacenada en ausencia de tensión de alimentación. Las memorias volátiles de lectura y escritura, RAM, se emplea para contener las variables temporales utilizadas en el mismo programa de la aplicación. Los tipos de memoria ROM más usuales se clasifican en función de la forma de grabar los datos:

- ROM, **Read Only Memory**: esta memoria se graba mediante la configuración interna de circuito integrado durante el proceso de fabricación. Luego, se debe proporcionar el programa de la aplicación al fabricante y resulta adecuada para largas series de fabricación.

- PROM, **ROM Programmable**: la memoria PROM incorpora fusibles en la constitución de cada celda interna de memoria, por lo que puede programarse por el usuario. La forma de grabar un dato en esta memoria suele consistir en aplicar una sobrecorriente al fusible de una celda de forma que cause su destrucción; en consecuencia, tan ~610 puede grabarse una vez.

- EPROM, **Erasable Programmable ROM** esta memoria emplea un transistor MOS de puerta flotante como elemento básico de cada una de sus celdas. Puede grabarse eléctricamente y borrarse mediante su exposición a una fuente de rayos ultravioleta, para lo cual dispone de un encapsulado especial cerámico con una ventana de cristal de cuarzo que encarece su coste. Esta memoria es adecuada para realizar prototipos y para series cortas de fabricación, aunque, en su versión OTP, **One Time Programmable**, se reduce considerablemente su costo debido a que se elimina la ventana de cuarzo del encapsulado, por lo que sólo puede programarse una vez y no se puede borrar.

- EEPROM o E2PROM, **Electrically Erasable and Programmable ROM** esta memoria utiliza el mismo transistor MOS de puerta flotante que la memoria EPROM, pero con una modificación tecnológica que permite grabar y borrar el transistor MOS eléctricamente mediante efecto túnel. La memoria EEPROM no tiene ventana de cuarzo, por lo que tiene un coste más reducido que la memoria EPROM. El tiempo de lectura de esta memoria es parecido al tiempo de lectura de las memorias RAM; no obstante, su tiempo de escritura, del orden de milisegundos, es demasiado elevado, lo que fuerza a establecer estados de espera en el proceso de escritura. Esta memoria es adecuada para aquellos sistemas que necesitan tener datos almacenados de forma estable, pero que pueden modificarse con cierta frecuencia.

- FLASH este tipo de memoria es de reciente aparición y son memorias EEPROM que tienen una configuración interna estructurada, a nivel de circuito, en bloques de bits, lo que permite reducir de manera considerable el área de silicio respecto a su homóloga EEPROM, lo que reduce, por tanto, su coste e incrementa su capacidad. El tiempo de escritura y de lectura de estas memorias es comparable con el requerido para las memorias RAM. Estas memorias se proporcionan con capacidades considerables, del orden del megabit, lo que hace que se apliquen para el almacenamiento de imágenes en cámaras digitales y en la sustitución de la cinta magnética de los contestadores telefónicos. El problema que presentan estas memorias es que pueden soportar un número bastante menor de ciclos de lectura y escritura que las memorias EEPROM.

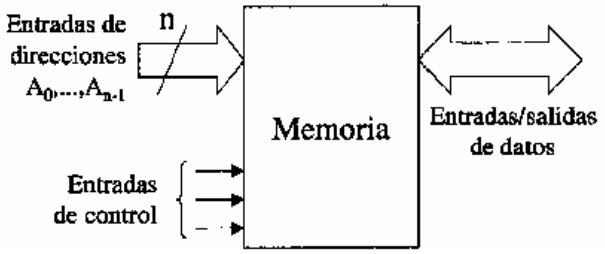
Las memorias del tipo RAM, **Ramdom Access Memory**, de lectura/escritura, se utilizan básicamente para grabar datos y resultados relacionados con el programa que ejecuta la aplicación, y son volátiles, es decir, pierden la información contenida cuando se interrumpe la tensión de alimentación del circuito integrado. Existen dos tipos de memorias RAM.

SRAM, **Static RAM**: esta memoria suele tener por base un biestable del tipo RS formado por cuatro transistores que almacenan el bit concreto, y un par de transistores adicionales que se usan para efectuar las operaciones de lectura/escritura.

DRAM, **Dynamic RAM**: existen muchos tipos de memorias DRAM en el mercado y se utilizan de forma masiva dentro del mercado de ordenadores destinados a la computación y el procesamiento de datos. La célula básica de esta memoria consiste en un único transistor y un condensador implementado en silicio de manera que su tamaño es sumamente reducido, por lo que suelen ser memorias de alta capacidad de almacenamiento. La información se almacena en el condensador de celda de la memoria, utilizando para ello la carga y descarga del condensador, factor que hace que sean memorias de bajo consumo. Tienen el inconveniente de que deben regrabarse cada pocos milisegundos, debido a que se producen pequeñas fugas en las cargas de los condensadores que causan la pérdida del estado lógico almacenado, lo que hace que tengan una circuitería interna compleja para el refresco de la memoria.

1.3.- Estructura externa de las memorias.

En la figura siguiente se presenta el esquema genérico de cualquier tipo de memoria y se indican cuáles son las entradas/salidas más comunes de este tipo de dispositivos.



Las memorias que se emplean para sistemas basados en un microcontrolador, suelen tener una estructura interna de celdas de tipo matricial, organizada en filas y columnas, donde cada celda contiene un bit de información, el número de columnas agrupa 8 bits (que forman un byte), y donde el número de filas determina la capacidad de la memoria. El tamaño de estas memorias es de 256x8bits o 256bytes, lkx8bits o 1 kbyte, 2kbytes, 4kbytes, 8kbytes y así sucesivamente en potencias de 2.

En estos sistemas también se utilizan memorias serie con un único bit de entrada/salida. El dato a leer o escribir en estas memorias puede tener varios tamaños: es común que sea de 8bits, de 16bits o de 32bits. Estas memorias suelen tener un encapsulado reducido y unas pocas líneas de control. Para acceder a cada posición de memoria se dispone de un número determinado de líneas de dirección, dependiendo del tamaño de la memoria que se desea emplear, de manera que se cumple la siguiente relación: $m = 2^n$, donde m es el tamaño de la memoria en bytes y n el número de líneas de dirección.

Las memorias también suelen tener 8 líneas de entrada/salida de datos, en las que se introducen los bytes que se quieren escribir o grabar. Se efectúa su escritura o lectura, según sea el caso, por medio de la activación de las líneas de control que tienen asociadas.

En líneas generales las memorias ROM, PROM y EPROM suelen tener una o más señales del tipo CS, **Chip Select**, o CE, **Chip Enable**, y OE, **Output Enable**, para seleccionar y realizar la lectura de la memoria, respectivamente.

En el caso de las memorias EPROM, éstas suelen tener dos señales de control: /CE, habilitación o selección del circuito integrado, y /OE, habilitación de la salida. La tabla siguiente resume los modos de funcionamiento de la memoria EPROM, considerando todos los posibles valores /CE y /OE.

/OE	/CE	Salidas	Modo
L	L	Dato	Lectura
H	L	Triestado	Salidas inhibidas
X	H	Triestado	Bajo consumo

En la tabla anterior se observa cómo las salidas presentan un estado triestado en el cual cada línea se pone en estado de alta impedancia, esta situación se da en la mayor parte de las memorias semiconductoras, cuando la señal CE es inactiva, ya que la memoria entra un estado de bajo consumo, con un consumo al menos un 25% menor de lo habitual.

En cuanto a las memorias del tipo RAM, suelen tener como señales de control /CS o /CE, que habilita el funcionamiento del circuito integrado, y RD/(/WR), Read/Write (lectura/escritura), que determina si se va a leer o escribir en la memoria; aunque, en lugar de esta señal combinada

lectura/escritura, puede tener una línea propia, /OE, para la lectura, y otra, /WE, **Write Enable**, para la escritura. La tabla siguiente resume el funcionamiento de esta memoria considerando las líneas /CS y RD/(/WR).

/CS	RD/WE	E/S de datos	Modo de funcionamiento
H	X	Triestado	Inhabilitada
L	H	Salida dato	Lectura
L	L	Entrada dato	Escritura

La tabla siguiente muestra el funcionamiento de la memoria RAM considerando las líneas /CE, /OE y /WE.

/CE	/OE	/WE	E/S de datos	Modo
H	X	X	Triestado	Standby
L	L	H	Salida dato	Lectura
L	H	L	Entrada dato	Escritura
L	L	L	Entrada dato	Escritura
L	H	H	Triestado	Inhabilitada

1.4.- Ciclos de fetch, de lectura y de escritura.

El acceso a la memoria por parte del microcontrolador se puede efectuar de tres formas distintas, denominadas ciclos, donde se lleva a cabo, en cada una de ellas, una secuencia determinada de tiempos. Estos ciclos son:

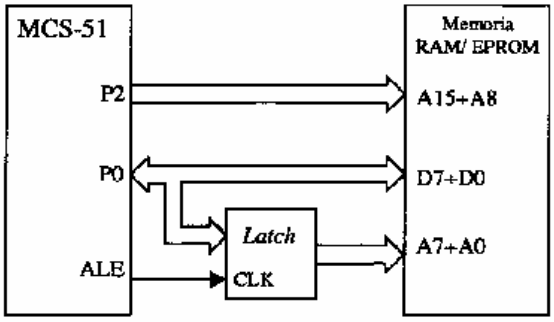
- **Ciclo de fetch o de lectura de código de operación**: el ciclo de fetch es una operación de lectura de la memoria externa, donde la información leída es el código de operación de las instrucciones que debe ejecutar el microcontrolador.
- **Ciclo de lectura**: el ciclo de lectura efectúa una lectura sobre la memoria externa para leer un operando de una instrucción, que puede ser un dato o una dirección.
- **Ciclo de escritura**: el ciclo de escritura escribe un dato sobre la memoria.

La duración de estos ciclos es diferente dependiendo del tipo de ciclo. La duración habitual de estos ciclos para la familia MCS-51 suele ser al menos de un ciclo máquina. Durante este tiempo se activan las señales involucradas en el acceso a la memoria externa de forma adecuada.

1.5.- Conexión entre la MCS-51 y la memoria externa.

En la conexión entre un microcontrolador de la MCS-51 y la memoria externa intervienen el bus de direcciones, el bus de datos y las líneas de control para gestionar la lectura/escritura en la memoria externa. El bus de direcciones para la MCS-5 1 es de 16 líneas y está soportado por los puertos P0 y P2 del microcontrolador. El bus de datos es de 8 bits y está ubicado en el puerto P0.

El puerto P0 entrega el byte bajo del bus de direcciones y el bus de datos, de forma simultánea, mediante una multiplexación temporal que se indica con la señal ALE. Esta multiplexación temporal se puede deshacer con un latch externo de 8 bits (figura) conectando el puerto P0 al latch y la señal ALE a la entrada de reloj de éste.



Además de la señal ALE, en la MCS-51 existen más líneas de control: /EA, PSEN, /RD y /WR. /EA, **Externa1 Access** inhibe, a 0 lógico, el acceso a la memoria interna de programas del microcontrolador. PSEN se activa cuando se accede a la memoria externa de programa. /RD se activa cuando el microcontrolador procede a la lectura de la memoria externa de datos (RAM). Y /WR se activa al escribir en la memoria externa de datos.

1.5.1.- Diagramas de tiempos para la MCS-51.

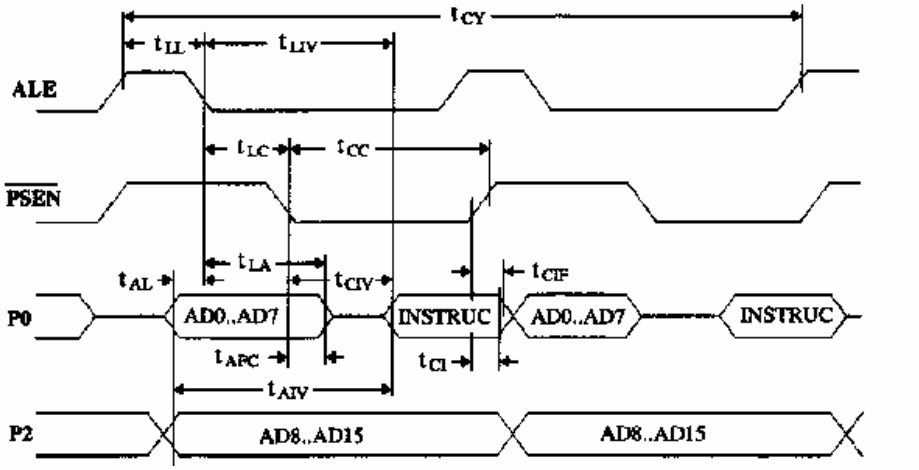
Las figuras siguientes muestran los diagramas de tiempos del ciclo de fetch, de lectura y de escritura para la MCS-51. En las tablas siguientes se indica el valor de los tiempos asociados para una frecuencia de reloj de 16MHz.

Valores mínimos:

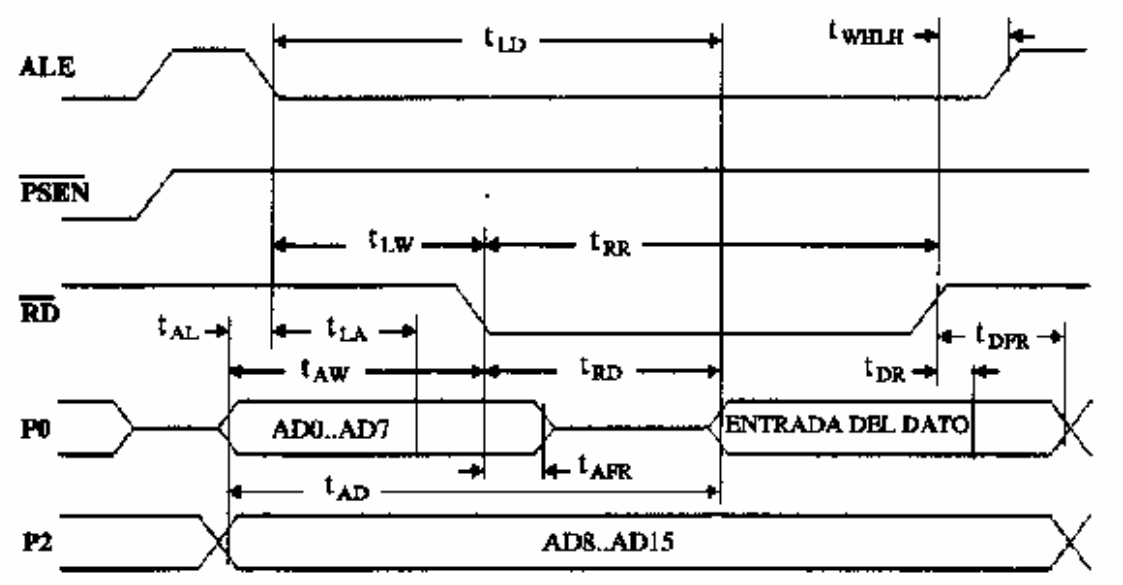
t_{LL}	t_{AL}	t_{LA}	t_{LC}	t_{CC}	t_{CI}	t_{RR}	t_{WW}	t_{DR}	t_{LW}	t_{AW}	t_{WHLH}	t_{DWX}	t_{DW}	t_{WD}
85	8	28	23	143	0	275	275	0	138	120	23	3	288	13

Valores máximos:

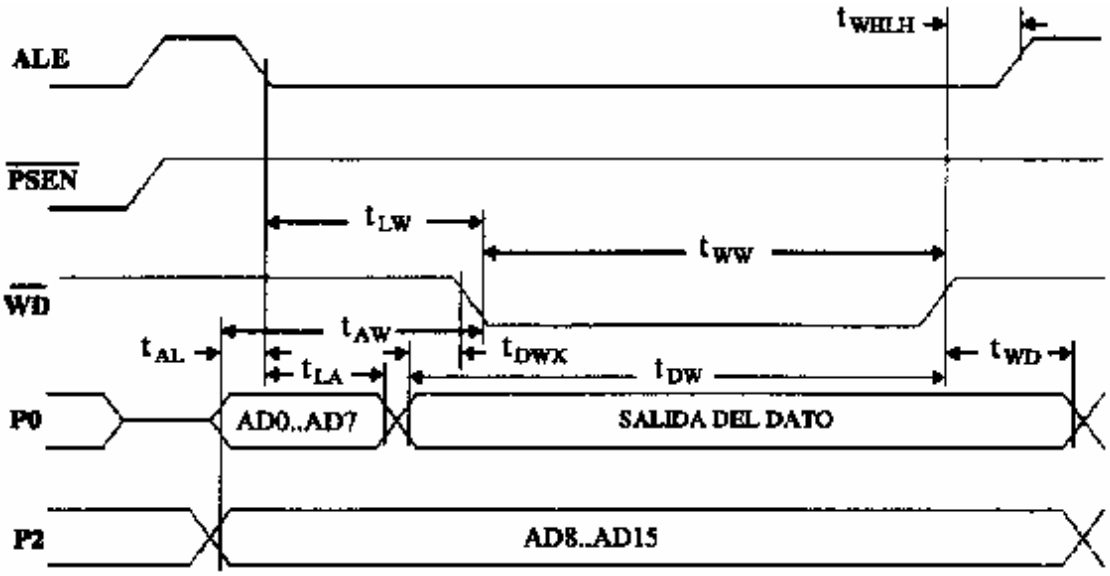
t_{LIV}	t_{CIV}	t_{CIF}	t_{AIV}	t_{CC}	t_{AFC}	t_{RD}	t_{DFR}	t_{LD}	t_{AD}	t_{LW}	t_{WHLH}	t_{AFR}
150	83	38	208	143	10	148	55	350	398	238	103	0



Ciclo FETCH en memoria externa.



Ciclo de lectura en memoria externa.



Ciclo de escritura en memoria externa.

En la ejecución del **ciclo de fetch** para la MCS-51 se realizan los siguientes pasos:

1. Colocar el byte alto y el byte bajo del bus de direcciones en el puerto P2, líneas AS, ..., A15, y P0, líneas A0,. .., A7, respectivamente.
2. Forzar a que la señal ALE tenga un flanco de bajada para que el contenido del puerto P0, líneas A0,. . ., A7, se almacenen en el latch.
3. Una vez que la dirección está presente a la entrada de la memoria, se activa la señal PSEN para realizar la lectura de la memoria.
4. La memoria coloca el código de operación en el bus de datos, puerto P0; el microcontrolador lee este dato coincidiendo con el flanco de subida de PSEN.

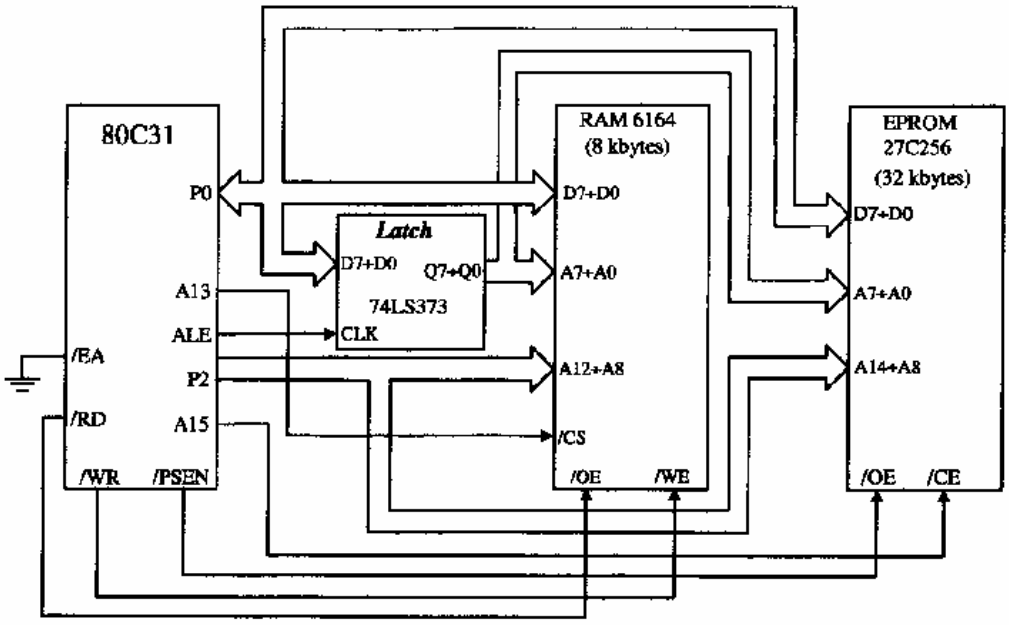
1.6.- Ejemplos de conexión para la MCS-51.

Ejemplo 1 Conexión de memorias EPROM y RAM.

La figura muestra un ejemplo de conexión de una memoria EPROM 27C256 de 32kbytes y una memoria RAM 6164 de 8kbytes a un microcontrolador 80C31. Según esta figura, la señal de habilitación de la memoria EPROM, /CE, está conectada directamente a línea A15 del bus de direcciones, de forma que cuando A15=0 la memoria está seleccionada, y cuando A15=1, no lo está. Este hecho implica que siempre que el código de las instrucciones del programa esté ubicado dentro de las primeras 32k direcciones, de las 64k direcciones posibles, la memoria EPROM quedará seleccionada, pues en el bus de direcciones aparece una dirección correspondiente a este rango. Estos rangos de direcciones donde se seleccionan las memorias EPROM, RAM o cualquier otro dispositivo que tenga que seleccionarse por medio de una dirección concreta, se denominan Mapa de memoria. El mapa de memoria donde se indican los rangos de direcciones de selección de las memorias EPROM y RAM de este ejemplo se muestran en la figura.

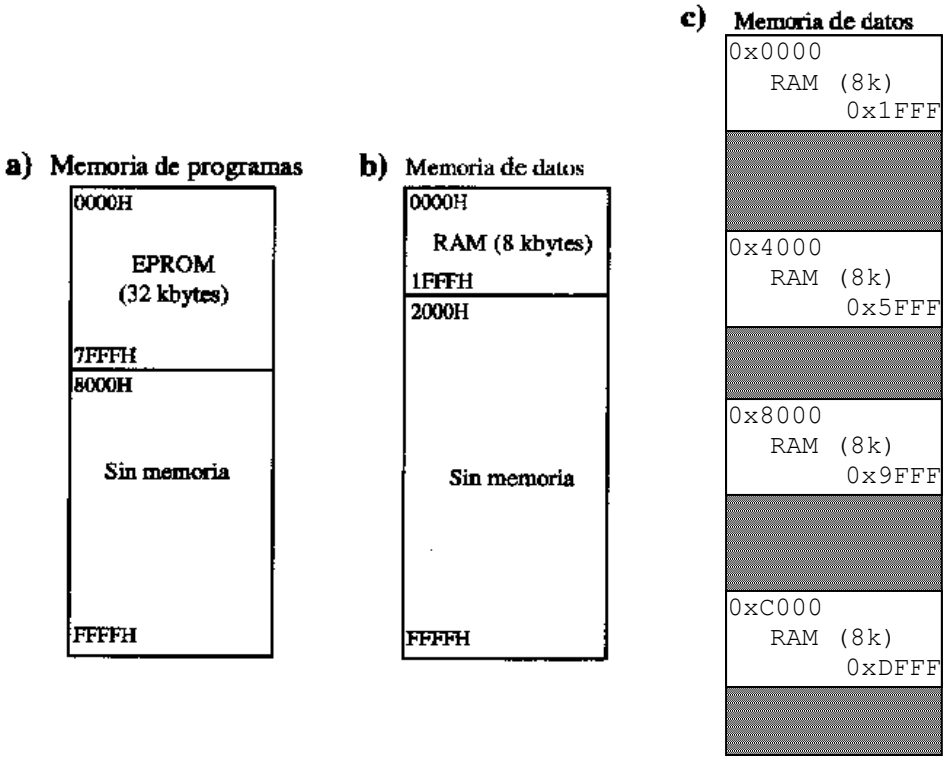
En el circuito de la figura se utiliza el circuito integrado 74LS373, constituido por 8 básculas D con salida triestado, para deshacer la multiplexación temporal en el puerto P0 entre las 8 líneas del bus de datos y las 8 líneas bajas del bus de direcciones. La señal de reloj de las básculas está conectada a la señal ALE, de manera que cuando ALE está a 1 lógico, el estado lógico presente a la entrada de las 8 básculas D pasa a la salida, y cuando la señal ALE pasa a 1 lógico, las básculas mantienen el último estado lógico presente en sus entradas.

La lectura de la memoria EPROM se realiza a través de la señal /PSEN (activa a 0 lógico y conectada a la entrada /OE de la memoria), para ello la memoria se debe seleccionar previamente con A15.

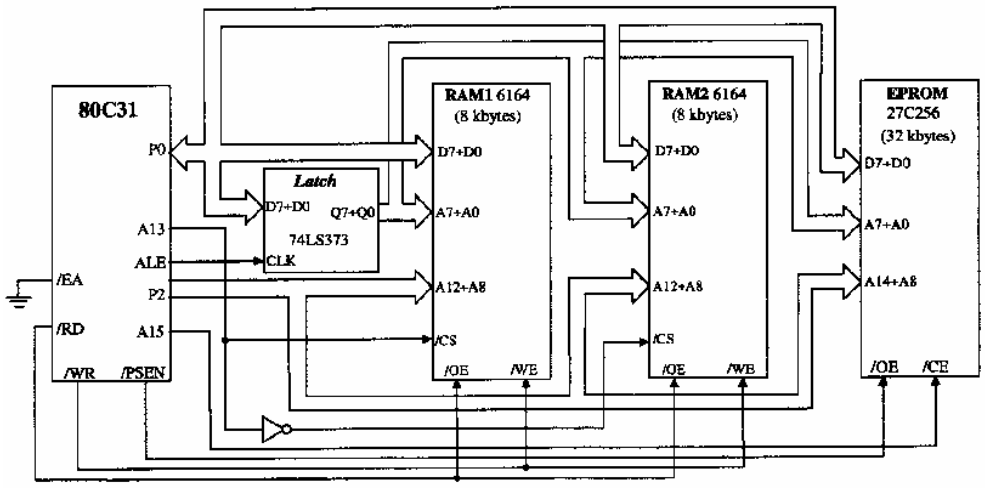


La selección de la memoria RAM se realiza por medio de la señal /CS, que selecciona la memoria cuando está a 0 lógico. La señal /CS se ha conectado directamente a la línea de dirección A13, de manera que la memoria se selecciona siempre que el microcontrolador realiza una lectura o escritura en los primeros 8kbytes de los 64kbytes posibles para la MCS-51. No obstante, la memoria RAM se selecciona cada vez que la señal /CS está a 0 lógico, lo que ocurre cuando las direcciones del bus están dentro de uno de los siguientes rangos de direcciones: 0x0000-0x1FFF, 0x4000-0x5FFF, 0x8000-0x9FFF y 0x0C000-0x0DFFF. Fuera de estas zonas

la memoria no está seleccionada, tal y como se aprecia en los mapas de memoria de la figura siguiente. El mapa de RAM verdadero es el (c), el (b) sería si hubiésemos empleado todos los bits de direcciones desde A15 hasta A13 para seleccionar la RAM. Esto último no se suele hacer porque resulta más costoso para sistemas pequeños, y lo hacemos como lo hemos hecho en el esquema anterior, quedando el mapa mostrado en (c).



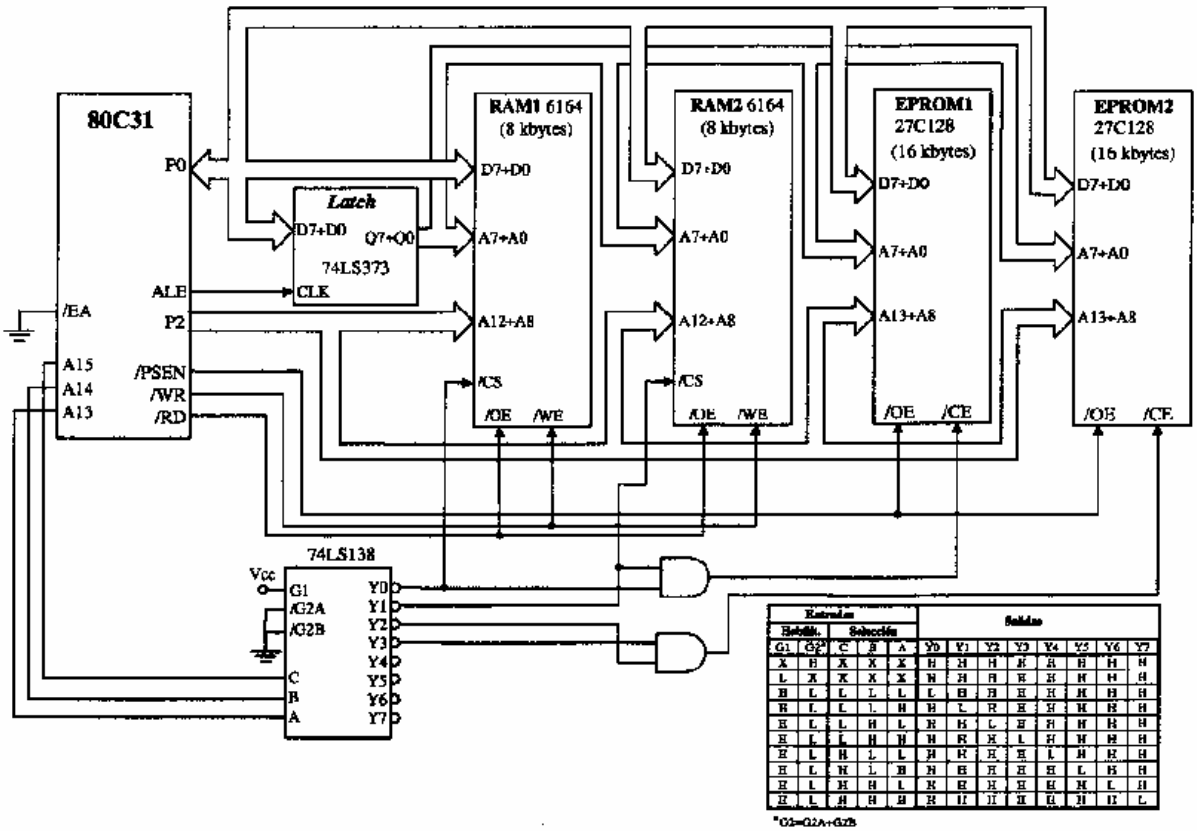
Según el mapa de memoria de la figura anterior, la capacidad de memoria RAM del sistema puede ampliarse en 8kbytes más conectando un circuito integrado de memoria RAM 6164 adicional, como se muestra en la figura siguiente. La memoria RAM2 se sitúa en las zonas del mapa de memoria de datos que no es imagen; para ello se conecta la línea A13 mediante una puerta lógica NOT a la señal ICS de la memoria. Con el circuito de la figura, la memoria RAM1 se selecciona cuando la línea A13 está a 0 lógico, mientras que la memoria RAM2 se selecciona cuando la línea A13 está a 1 lógico. La memoria RAM2 ocupa las zonas sombreadas que hay en el mapa de memoria de la figura anterior, y se selecciona cuando las direcciones del bus están dentro de uno de los siguientes rangos de direcciones: 0x2000-0x3FFF, 0x6000-0x7FFF, 0x0A000-0x0BFFF y 0x0E000-0x0FFFF.



Para utilizar más circuitos integrados de memoria se necesita emplear una lógica de selección de la memoria algo más compleja, que garantice la conexión de más memorias y de otros dispositivos al microcontrolador; para ello en el siguiente ejemplo se propone el uso de un decodificador en la selección de las memorias.

Ejemplo 2 Selección de memorias mediante el 74LS138.

La figura siguiente muestra el circuito de conexión de dos memorias EPROM 27C128 de l6 kbytes cada una, y de dos memorias RAM 6164 de 8kbytes cada una. La selección de las memorias se efectúa con el decodificador de 3 a 8 líneas 74LS138.



Las salidas Y0-Y7 del 74LS138 se activan a 0 lógico cuando en las entradas, C, B y A, aparece el código correspondiente de la salida, según la tabla de verdad del decodificador de la figura siguiente. Las entradas G1,/G2A y /G2B son las líneas de habilitación del decodificador y deben estar a 1, 0 y 0 lógicos, respectivamente, para que el decodificador funcione correctamente. El decodificador queda inhabilitado si la entrada G1 está a 0 lógico o cualquiera de las entradas /G2A o /G2B está a 1 lógico, es decir, si cualquiera de estas entradas está inhabilitada.

Conectando la línea A15 a la entrada C del 74LS138, la línea A14 a la entrada B y la línea A13 a la entrada A, la salida Y0 del decodificador se activa siempre y cuando las líneas A15, A14 y A13 estén a 0 lógico, lo que ocurre en el rango de direcciones comprendido entre 0000H y 1FFFH, es decir, las primeras 8k posiciones del mapa de memoria.

De la misma forma, cada una de las restantes entradas se activa con un rango de 8k posiciones del mapa de memoria, según la tabla siguiente.

Salida activa	Espacio de memoria	Memoria habilitada
Y0	0000H-1FFFH	RAM1 y EPROM1
Y1	2000H-3FFFH	RAM2 y EPROM1
Y2	4000H-5FFFH	EPROM2
Y3	6000H-7FFFH	EPROM2
Y4	8000H-9FFFH	-
Y5	A000H-BFFFH	-
Y6	C000H-DFFFH	-
Y7	E000H-FFFFH	-

Las líneas de selección de las memorias RAM1 y RAM2, al tener una capacidad de 8kbytes, se conectan directamente a las salidas Y0 y Y1 del decodificador, respectivamente. La memoria EPROM1 tiene una capacidad de l6 kbytes, por lo que su línea de selección, /CE, está conectada a las salidas Y0 e Y1 por medio de una puerta AND. Esta memoria se habilita siempre y cuando el microcontrolador efectúe una lectura en los primeros l6 kbytes del mapa de memoria. De la misma forma, la entrada /CE de la memoria EPROM2 se conecta a las salidas Y2 e Y3; se habilita cuando el microcontrolador realiza una lectura en el espacio comprendido entre 4000H y 7FFFH del mapa de memoria.

2.- Puerto de comunicación serie.

2.1.- Introducción.

Es frecuente que un sistema basado en microprocesador tenga que transmitir datos hacia otro sistema, o hacia un periférico, o un terminal del sistema. La información se puede enviar en serie a través de una simple línea, o canal de comunicación; o en paralelo, lo que requiere de un número considerable de líneas. La transmisión serie es adecuada para largas distancias por su simplicidad, y la transmisión paralelo es adecuada en distancias relativamente cortas, debido a que el número de líneas hace que sea una solución con un elevado costo.

Para transmitir datos por la línea telefónica a largas distancias es conveniente emplear un módem, que transforme las señales digitales a un formato analógico y que, así, haga ocupar un ancho de banda menor, debido a que la transmisión digital directa resulta inapropiada por el enorme ancho de banda que supone. No obstante, cuando las líneas de comunicación son propias del sistema, la transmisión se puede hacer íntegramente en formato digital, para lo cual existen varios estándares de comunicación, como el RS-232, el RS-422, el RS-485 y otros.

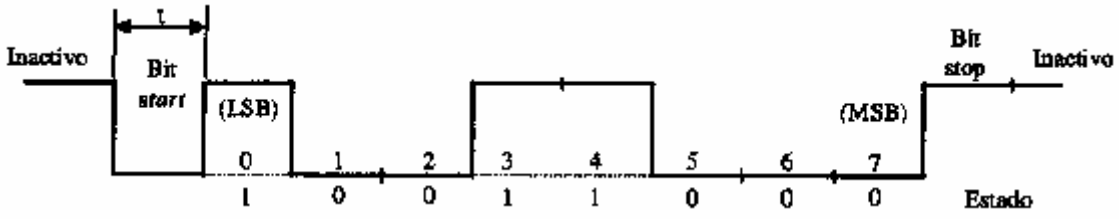
El formato de la información a transmitir en una comunicación serie puede ser de 7 ó 8 bits, coincidiendo con los empleados por la tabla de caracteres ASCII. La transmisión puede ser síncrona o asíncrona y puede efectuarse en varios sentidos: simplex, half-duplex y full-duplex como vimos en el tema anterior.

Cuando la comunicación serie es síncrona se envía una señal de reloj en la transmisión que marca las pautas de la comunicación. Esta señal se puede proporcionar a través de una línea adicional entre los sistemas que se quiere comunicar, o bien, se puede extraer de alguna forma específica de codificación: en ese caso el receptor obtiene la señal de reloj de la propia señal recibida.

En el caso de que la comunicación sea asíncrona tanto el transmisor como el receptor deben secuencia la comunicación de acuerdo con una base de tiempos, pues la transmisión o la recepción pueden ocurrir en cualquier momento. En este tipo de comunicación la inactividad en la transmisión se indica mediante un nivel lógico alto.

El inicio de la transmisión de un carácter, en una comunicación serie asíncrona, se indica mediante una transición brusca entre un nivel lógico alto y un nivel lógico bajo, a partir del cual se extrae cada uno de los bits hacia la línea, con intervalos fijos de tiempo t (figura 9.1): los intervalos de tiempo en la comunicación determinan la velocidad de transmisión en bits por segundo, bits/seg, o baudios. El primer bit de la transmisión consiste en un bit de inicio (start), que es un cero lógico. Luego le siguen los bits del dato, comenzando por el bit menos significativo y finalizando con un bit, o dos, de parada (stop).

Al principio de establecer las bases de la comunicación serie, se utilizaban dos bits de stop por cuestiones tecnológicas del momento. No obstante, en la actualidad es suficiente con emplear un solo bit de stop. El bit de stop (figura siguiente) se indica manteniendo un nivel lógico alto durante un intervalo t, tras el cual, puede procederse a transmitir el siguiente dato. Así, la inclusión del bit de start y el de stop, en la comunicación serie asíncrona, resulta imprescindible para separar y distinguir los datos transmitidos.



Generalmente, el dato en una comunicación serie es de 8 bits, a los que se puede añadir un noveno bit de paridad, con el fin de detectar errores en la transmisión.

El sentido de la transmisión es simplex cuando se realiza en una Única dirección, del emisor al receptor, es haf-duplex cuando se puede realizar en ambas direcciones, emisor-receptor y receptor-emisor (pero no en ambas direcciones simultáneamente), y full-duplex cuando se pueden enviar datos en ambas direcciones de forma simultánea.

En la transmisión asíncrona se puede dar el caso de que un sistema sea lento a la hora de procesar los datos que recibe, por lo que éste debe ser capaz de indicarle al transmisor su incapacidad de procesar más datos y que, por tanto, se espere hasta una nueva transmisión. Este problema se suele solventar mediante el protocolo XONKOFF, en el cual se dispone de dos caracteres ASCII de control, DC1 (XON) y DC3 (XOFF), para detener y para reanudar la comunicación, respectivamente. Cuando el receptor no puede procesar más datos envía un carácter XOFF al emisor, tras el cual el emisor deja de transmitir hasta que recibe un carácter XON.

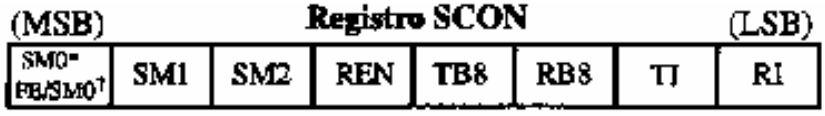
2.2.- La comunicación serie en la MCS-51.

El puerto de comunicación serie de la MCS-51 permite realizar transmisiones de datos en serie en los modos síncrono y asíncrono. Para la transmisión síncrona dispone de un único modo de funcionamiento, modo 0, y para la transmisión asíncrona full-duplex con velocidad de transmisión programable dispone de los modos 1, 2 y 3 de funcionamiento. La generación de la velocidad de transmisión del puerto serie se puede basar tanto en el Timer 2 como en el Timer 2, y utiliza sus recursos para determinar la base de tiempos de la transmisión.

El puerto serie, además, tiene un sistema automático que es capaz de detectar un error en el bit de stop. Además, el bit de paridad se puede emplear en la transmisión para poder detectar un error de 1 bit en la comunicación; para ello el bit de paridad se transmite con el dato como noveno bit de transmisión. El puerto también tiene un sistema de reconocimiento de direcciones, el cual permite la conexión de varios microcontroladores entre sí.

La determinación de la velocidad de transmisión depende del modo en que se esté operando. Por ejemplo, para el Timer 2, en el modo 0 tan sólo se permite una velocidad de transmisión, mientras que en el modo 2 pueden programarse dos velocidades de transmisión, y en los modos 1 y 3 se pueden programar distintas velocidades de transmisión.

Los terminales asociados al puerto serie son TXD, terminal P3.1, y RXD, terminal P3.0. En el modo 0, el terminal TXD soporta la señal de reloj de la transmisión síncrona.



Bit	Comentario
FE/SM0	Frame Error o bit 0 de selección de modo del puerto serie. Esta opción solo es válida para las versiones 8XC51Fx de MCS-51. Este bit actua como FE o SM0 según sea el estado del registro SMOD0 (bit 6 registro PCON), si SMOD0=1 actua como FE y si SMOD0=0 actúa como SM0. El bit FE se activa a 1 lógico cuando se detecta un bit de stop inválido en la recepción de un dato. Una vez activo, este bit no se borra por otros datos recibidos y se debe borrar por software. SM0 es el bit de selección del puerto serie, su modo de funcionamiento se selecciona junto a SM1.
SM0	Bit 0 de selección de modo del puerto serie. Aparece con este formato en la MCS-51, excepto para las versiones 8X51Fx. SM0 junto con SM1, determinan el modo de funcionamiento del puerto serie.
SM1	Bit 1 de selección de modo del puerto serie. SM1 SM0 Modo 0 0 0 0 1 1 1 0 2 1 1 3
SM2	Bit 2 de modo del puerto serie. Permite habilitar o no, la comunicación serie, en entornos multiprocesador.
REN	Bit de habilitación de la recepción. REN=1 habilita la recepción de un dato y REN=0 la inhabilita
TB8	Bit noveno de transmisión. Es el décimo bit en la transmisión de un dato para los modos 2 y 3
RB8	Bit noveno de recepción. Obtiene el valor del décimo bit del dato transmitido en los modos 2 y 3.
TI	Bit de interrupción en transmisión. Se pone a 1 cuando finaliza la transmisión del dato. Debe borrarse por software.
RI	Bit de interrupción en recepción. Se pone a 1 cuando se recibe un dato. Debe borrarse por software.

Los registros asociados al puerto de comunicación serie son SBUF, SCON en toda la familia MCS-51, y los registros SADDR y SADEN para las versiones 8XC52/54/58,8XC51FX y 87C51GB. El registro SBUF, Serial Buffer, es el buffer del puerto serie, que almacena el dato recibido o enviado por el puerto. El registro SCON, Serial Control (tabla 9.1), es el registro de control que permite programar las características de funcionamiento del puerto. El registro SADDR se utiliza para definir la dirección del microcontrolador que hace la función de esclavo, cuando éste se encuentra en un entorno de comunicación multiprocesador. Finalmente, el registro SADEN especifica el byte de máscara para el microcontrolador esclavo, en la comunicación multiprocesador.

En el puerto de comunicación serie el registro SBUF está duplicado, de manera que uno de los registros está conectado al terminal TXD y el otro registro está conectado al terminal RXD. Esta duplicidad del registro SBUF permite poder transmitir y recibir datos simultáneamente (comunicación full-duplex). Sin embargo, con el registro SBUF se trabaja de una manera sencilla, pues se trata a éste como a un único registro: se transmite un dato al cargarlo en SBUF, y se lee un dato de SBUF cuando éste se ha recibido.

2.3.- Modos de funcionamiento del puerto serie.

El puerto de comunicación serie de la MCS-51 puede funcionar con cuatro modos distintos de funcionamiento: modos 0, 1, 2 y 3. El modo 0 se emplea para la comunicación síncrona y los modos 1,2 y 3 en la comunicación asíncrona.

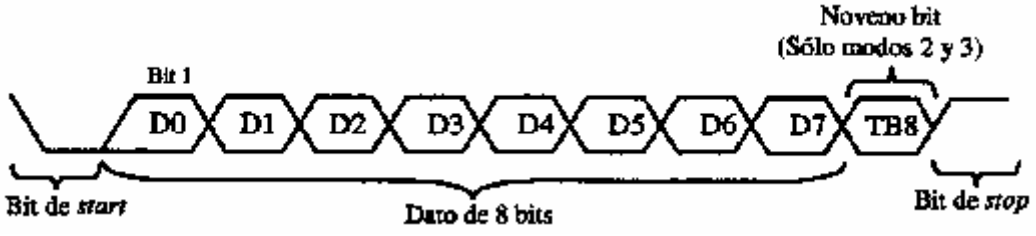
En cualquiera de los modos de funcionamiento del puerto serie, la transmisión se efectúa en el momento que se carga el registro SBUF con un dato mediante una instrucción de escritura. La recepción de un dato debe habilitarse de antemano, poniendo a 1 lógico el bit REN del registro SCON y poniendo a cero el bit RI. Cuando se recibe un dato el bit RI se pone a 1 lógico y activa el proceso de interrupción del puerto, en el cual debe realizarse la lectura del registro SBUF por medio de la rutina de RSI.

2.3.1.- Modo 0. Modo síncrono.

Este modo se determina mediante la puesta a cero lógico de los bits SM0 y SM1 del registro SCON. La velocidad de transmisión de datos es de Fosc/12, donde Fosc es la frecuencia de reloj del microcontrolador. El terminal TXD genera la señal de reloj que sincroniza la comunicación, y la transmisión y recepción de datos se lleva a cabo a través del terminal RXD.

2.3.2.- Modos 1,2 y 3. Modos asíncronos.

En los modos 1,2 y 3 se pueden enviar datos a través de la línea TXD y recibirlos a través de la línea RXD de forma simultánea (Full-duplex). En la figura siguiente se muestra la trama del dato en la transmisión de estos modos.



a) Modo 1

En el modo 1, la trama de datos está formada por 10 bits. El primer bit de la trama es el bit de start, a 0 lógico, le siguen los 8 bits del dato, y finaliza con un décimo bit que consiste en el bit de stop, a 1 lógico.

La base de tiempos se puede generar, en este modo, por medio del Timer 1, del Timer 2 (en aquellas versiones que disponen de tres Timers) o de ambos Timers. La velocidad de transmisión del puerto serie, en baudios, depende del tiempo de desbordamiento de los Timers. Por defecto, la velocidad de transmisión la establece el Timer 1, y viene dada en baudios por la fórmula siguiente:

$$\text{Frec. transmisión} = 2^{X} \cdot \frac{\text{Frec. rebasamiento del } \textit{Timer 1}}{32}$$

, donde X puede valer 0 ó 1 lógico. Para la MCS-51, excepto las versiones 8XC51Fx, el bit X se corresponde con el bit SMOD del registro PCON. Para las versiones 8XC51Fx el bit X se corresponde con el bit SMOD1 del registro PCON. En realidad el bit SMOD y SMODl son el mismo bit con distintos nombres, y corresponden al bit 7 del registro PCON.

En la comunicación serie se debe evitar que el Timer provoque una interrupción cada vez que llega a desbordamiento; para ello debe de inhibirse la interrupción del Timer mediante el bit ET del registro IE. El Timer 1 se suele configurar en el modo 2, como temporizador de 8 bits con autorrecarga (modo 2), aunque puede también configurarse en los modos 0, 2 y 3. Para este caso, la velocidad de transmisión dependerá del valor cargado el registro TH1 del Timer 1:

$$\text{Frec. transmisión} = 2^{X} \cdot \frac{F_{osc}}{32 \cdot 12 \cdot [256 - TH1]}$$

, donde F_{OSC} es la frecuencia de reloj del microcontrolador. El bit X, al igual que para la ecuación anterior, es el séptimo bit del registro PCON. La tabla siguiente muestra distintas frecuencias de transmisión que se pueden obtener con el Timer 1 en el modo 2 de funcionamiento.

Frecuencia (Baudios)	Frecuencia Oscilador	SMOD	Timer 1	
			C/T	Valor recarga TH1
62.5 Kbaud(Max)	12.0 MHz	1	0	FFH
19.2 Kbaud	11.0592 MHz	1	0	FDH
9.6 Kbaud	11.0592 MHz	0	0	FDH
4.8 Kbaud	11.0592 MHz	0	0	FAH
2.4 Kbaud	11.0592 MHz	0	0	F4H
1.2 Kbaud	11.0592 MHz	0	0	E8H
137.5 baud	11.0592 MHz	0	0	1DH
110.0 Kbaud	6.0 MHz	0	0	72H

b) Modo 2

En el modo 2, la trama de bits del puerto serie está compuesta de 11 bits. El primer bit es el bit de start, le siguen los 8 bits del dato más un noveno bit definible por el usuario, y termina con un bit de stop. El noveno bit del dato transmitido se corresponde con el estado del bit TB8 del registro SCON. En la recepción, este noveno bit se ubica en el bit RB8 del registro SCON.

Para transmitir en este modo se debe poner el noveno bit a transmitir en el bit TR8, antes de realizar la escritura del dato en el registro SBUF. La recepción se debe habilitar antes que se produzca, poniendo para ello el bit REN a 1 lógico y el bit RI a 0 lógico.

En el modo 2 sólo se permiten dos velocidades de comunicación para el puerto serie, dependiendo del valor del bit X

$$\text{Frec. transmisión} = 2^{x} \cdot \frac{F_{osc}}{64}$$

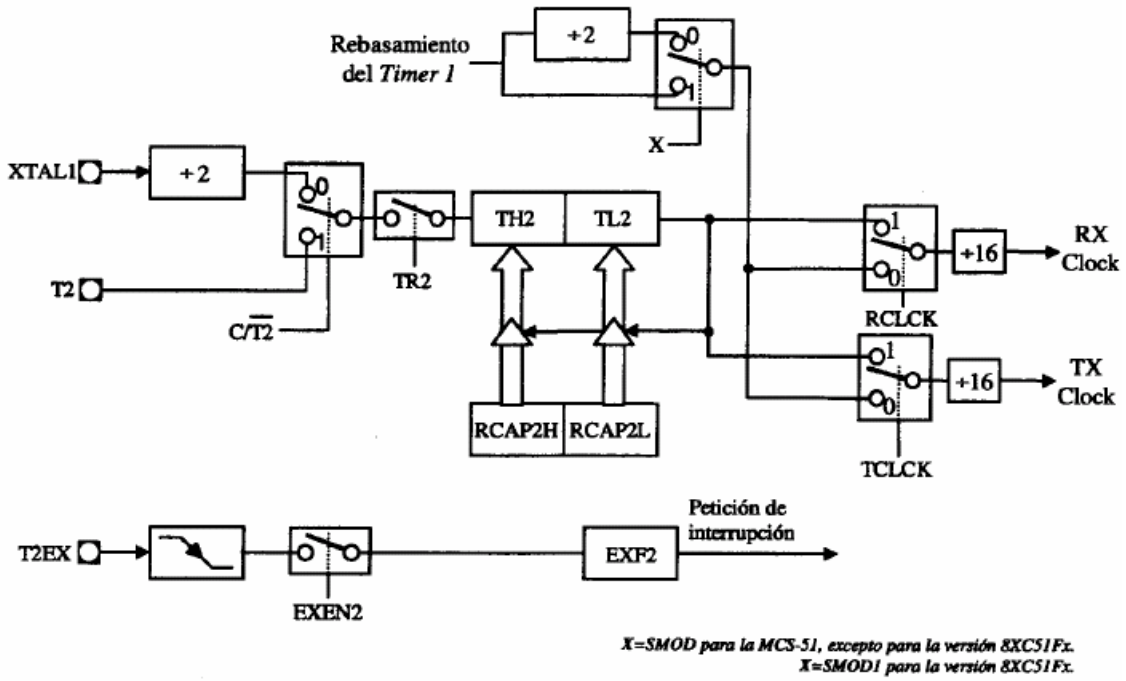
El bit X tiene la misma correspondencia que la definida para el resto de modos de funcionamiento.

c) Modo 3

El modo 3 del puerto serie es similar al modo 2, con la Única diferencia que la velocidad de la comunicación del puerto serie se obtiene de la misma manera que en el modo 1.

2.3.3 El Timer 2 como base para el puerto serie.

El Timer 2 tiene un modo de funcionamiento específico utilizado para fijar la velocidad de comunicación de datos por el puerto serie, Baud Rate Generator Mode (ver figura). Para seleccionar este modo se deben programar adecuadamente los bits RCLCK y TCLCK del registro T2CON, como se muestra en la tabla.



RCLCK	TCLCK	Generador de baudios en recepción	Generador de baudios en transmisión
0	0	Timer 1	Timer 1
0	1	Timer 1	Timer 2
1	0	Timer 2	Timer 1
1	1	Timer 2	Timer 2

Cuando el Timer 2 se selecciona como generador de baudios funciona de forma similar al modo autorrecarga. Los registros TH2 y TL2 se recargan con el valor de los registros RCAF'2H y RCAP2L, en cada desbordamiento. La velocidad de comunicación depende del valor puesto en estos registros:

$$\text{Frec. transmisión} = \frac{F_{OSC}}{32 \cdot [65536 - (RCAP2H, RCAP2L)]}$$

En la figura anterior se muestra el diagrama de bloques del Timer 2 configurado en modo generador de baudios. En la tabla siguiente se indican las distintas velocidades de comunicación y el valor a cargar en los registros RCAP2H y RCAP2L para conseguir distintas velocidades de transmisión.

Baudios	Frecuencia oscilador	RCAP2H	RCAP2L
375.0 Kbaud	12.0 MHz	FFH	FFH
9.6 Kbaud	12.0 MHz	FFH	D9H
4.8 Kbaud	12.0 MHz	FFH	B2H
2.4 Kbaud	12.0 MHz	FFH	64H
1.2 Kbaud	12.0 MHz	FEH	C8H
300.0 baud	12.0 MHz	FBH	1EH
110.0 baud	12.0 MHz	F2H	AFH
300.0 baud	6.0 MHz	FDH	8FH
110.0 baud	6.0 MHz	F9H	57H

2.4 Detección de errores.

En las versiones 8XC51Fx es posible detectar errores en la recepción del bit de stop para los modos de trabajo 1,2 y 3, para lo cual se debe poner a 1 lógico el bit SMOD0 del registro PCON. Con esta opción habilitada, el puerto serie comprueba el bit de stop de cada dato recibido y si es erróneo se pone a 1, de manera automática, el bit FE del registro SCON. En consecuencia, para emplear este recurso, se debe examinar el estado del bit FE cada vez que se recibe un dato. El bit FE no es modificado por datos posteriores recibidos en el puerto serie, y se debe borrar por software, por lo que se deberá incluir una instrucción que lo borre cada vez que se active.

Ejemplo 1 Control de una impresora de 40 columnas

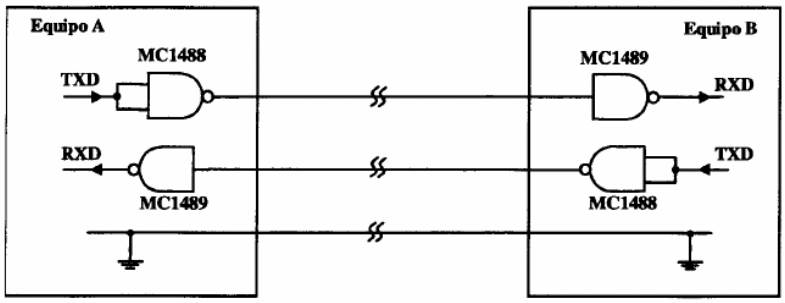
En los equipos de tipo comercial, como una balanza electrónica o un terminal punto de venta, se utilizan impresoras de 40 columnas para la impresión del ticket de la compra del usuario. Estas impresoras se conectan al sistema basado en microcontrolador mediante el bus RS-232C, que es un bus de comunicación para distancias cortas y velocidades de comunicación moderadas.

Con el bus RS-232C sólo se pueden conectar un emisor y un receptor, y las velocidades de comunicación más comunes son: 300, 1.200, 2.400, 4.800, 9.600 y 19.200. Los niveles de tensión del bus son bipolares, el transmisor debe generar un nivel de tensión entre -5 y -15V para un 1 lógico, y entre +5 y +15V para un 0 lógico. El receptor responde con una tensión más negativa que -3V para un 1 lógico, y con una tensión más positiva que +3V para un 0 lógico.

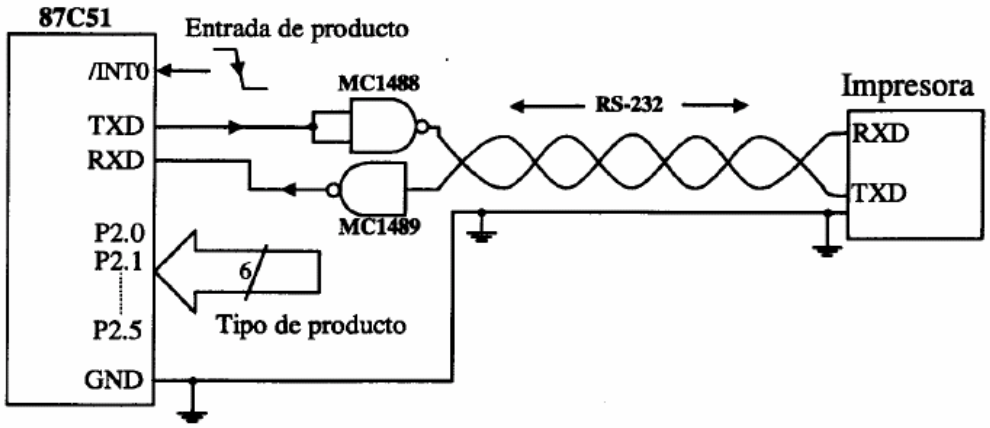
Para realizar la interfaz con el bus existen distintos circuitos integrados de interfaz entre niveles TTL y RS-232, y viceversa. Los circuitos integrados más comunes son el MC1488, que tiene cuatro transmisores, y el MC1489A, que tiene cuatro receptores. Estos circuitos integrados los realizan distintos fabricantes y son de bajo coste. El MC1488 necesita una tensión de alimentación entre ±9V y ±15V. Para niveles CMOS se pueden usar el DS14C88 y el S14C89A de National Semiconductor, para la interfaz entre CMOS y RS-232, y viceversa. También se

pueden usar unos integrados de MAXIM, los MAX232 que incluyen un elevador de tensión para obtener a partir de 5V los niveles compatibles RS-232.

El número mínimo de señales con el que se puede realizar la comunicación bidireccional entre dos equipos es: TXD para la transmisión, RXD para la recepción y una línea de masa (figura siguiente).



En este ejemplo se desea conectar una impresora de 40 columnas a un 87C51 mediante el bus RS-232 (figura siguiente). El microcontrolador se utiliza en una aplicación de etiquetaje, donde se imprime un número asignado al producto y un texto determinado según el tipo de producto. Las etiquetas son autoadhesivas y se pegan al producto mediante un aplicador automático.



El control de la impresora se realiza mediante caracteres ASCII reservados para ello. El microcontrolador iniciará la comunicación con la impresora enviándole el carácter ASCII STX Start of text, que se corresponde con el número hexadecimal 02H. Tras este carácter se le puede enviar un carácter de comando como LF, Line feed, (0AH); CAN, Cancel, (18H); ESC, Escape, (1BH); para cambiar el carácter de impresión en negrita, cursiva, etc. Tras este comando, se le envía el texto que se quiere imprimir en ASCII, y por último el comando de impresión CR, Carriage return, (0DH), por lo que se imprimirá el texto transmitido.

En este ejemplo se trata tan sólo de imprimir el texto "PRODUCTO Nº:" seguido de cuatro cifras que identifican el tipo de producto. La aplicación tiene un interruptor de final de carrera como sensor de la llegada del producto conectado a la entrada de interrupción /INT0. El interruptor proporciona un 0 lógico cuando hay un producto a etiquetar y un 1 lógico cuando no hay producto. La empresa tiene 64 tipos de productos diferentes, codificados con cuatro cifras cada uno. Los códigos de estos productos se han almacenado en la memoria de programas. El microcontrolador recibe en el puerto P2 seis líneas de entrada que identifican el tipo de producto, al mismo tiempo que se activa el interruptor final de carrera conectado a /INT0.

El programa de este ejemplo se muestra a continuación:

```
#include <reg51.h>
#include <stdio.h>

#define uchar unsigned char
bit Start_TX; /* Bit que inicia la transmisión */

/*****************************************************************/
/* Programa comunicación y control de una impresora de 40 columnas  */
/*****************************************************************/

/* Rutinas de servicio de interrupción */
/* vector generado como (8*n+3) */

/*****************************************************************/
/*                Rutina de servicio de INT0                     */
/*****************************************************************/
void int0_int (void) interrupt 0 using 1
{
      Start_TX=1; /* para indicar inicio de transmisión */
}

/*****************************************************************/
/*                       Subrutina de impresión                  */
/*****************************************************************/

uchar code cabecera[14]="PRODUCTO N°: ";
uchar data temporal[18];
uchar code tabla[10]={0x10,0x15,          /* Cód producto no 1 */
                      0x25,0x46,          /* Cód producto no 2 */
                      0x1A,0x1F,          /* Cód producto no 3 */
                      0x78,0x22,          /* Cód producto no 4 */
                      0x99,0x56};         /* Cód producto no 5 */

void copia_de_ROM(uchar data *temporal,uchar code* tmp2)
{
uchar data *punt1=temporal;
uchar code *punt2=tmp2;

      while (*punt2) {
            *punt1=*punt2;
            punt1++;
            punt2++;
            }
}

uchar bin_ASCII(uchar valor)
{
return (valor+0x30);
}

void Imprime_cod(uchar valor)
{
      copia_de_ROM(temporal,cabecera);/* copia caracteres ROM a RAM */
            /* Extrae primer carácter primer código */
      temporal[12]=bin_ASCII(tabla[valor]>>4);
            /* Extrae segundo carácter primer código */
      temporal[13]=bin_ASCII(tabla[valor]&0x0F);
            /* Extrae primer carácter segundo código */
      temporal[14]=bin_ASCII(tabla[valor+1]>>4);
```

```
            /* Extrae segundo carácter segundo código */
      temporal[15]=bin_ASCII(tabla[valor+1]&0x0F);
      temporal[16]='\n';            /* Salto de línea */
      temporal[17]=0x0;             /* Coloco fin de cadena */
      printf(temporal);
}

/*****************************************************************/
/* Rutina de inicio                                              */
/*****************************************************************/

void main(void) {
uchar producto; /* variable para almacenar el código de producto */

      IT0 = 1;     /* interrupción INT0 por flanco descendente */
      EX0 = 1;
      EA  =1;
      SCON = 0x52; /* Configura comunicación serie en modo 1 */
      TCON = 0x40; /* TCON */
      TMOD = 0x20; /* Timer 1 en Modo 2, GATE=0 y C/T=0 */
      TH1 = 0x0FD; /* Valor para 9600 baudios */

      Start_TX = 0;     /* para indicar que se puede transmitir */

      while (1) {
            while (!Start_TX);  /* Bucle espera a que Start_TX sea 1 */
            Start_TX=0;             /* Pone a cero Start_TX */
            producto=P2;            /* Lee P2 tipo de producto */
            producto=producto & 0x3F; /* Enmascara bits no usados */
            Imprime_cod(producto);  /* llamada función de impresión */
      }
}
```

En este programa se utiliza la variable Start_TX como indicativo de que ha entrado un producto. La transmisión consiste en ir colocando los datos contenidos en la cadena de texto en la memoria interna que se utiliza como buffer de datos para la transmisión, esta es la variable de tipo array de unsigned char “temporal”. En total se transmiten cada vez 18 caracteres: 14 caracteres correspondientes al texto y 4 caracteres correspondientes al código de cuatro cifras del producto.

El código del producto se ha situado en el array “tabla” y está formado por dos bytes. En el programa primero se lee el byte alto del código en la tabla, que se desglosa en dos bytes en formato ASCII: el primero corresponde a los 4 bits altos y el otro a los 4 bits bajos, esto es, se encuentra codificado en BCD EMPAQUETADO. Luego se lee el byte bajo del código, que se convierte a ASCII de la misma manera que el primer byte.

A la impresora se transmite primero el carácter STX (0x02), seguido de 18 bytes, almacenados en el buffer de memoria y que corresponden al texto y al código de producto, y por último el carácter CR (0x0D), que hace que se impriman los datos transmitidos.

Para realizar la comunicación se ha seleccionado una velocidad de 2.400 baudios; para ello se configura el puerto serie en el modo 1, en el que se transmiten 10 bits: 1 bit de start, 8 bits de dato y 1 bit de stop. El Timer 1 del microcontrolador se utiliza como base de tiempos para la transmisión, por lo que debe estar configurado en el modo 2 de 8 bits con autorrecarga, con el bit C/T a 0 lógico, con el bit SMOD a 0 lógico, con el valor FDH de recarga en el registro THl , y con una frecuencia de reloj de 12.000MHz.

Ejemplo 2 Comunicación multipunto mediante RS-485.

En la industria de la alimentación es frecuente tener múltiples cámaras frigoríficas en distintos puntos de una nave industrial, o bien dentro de una misma área comercial. La regulación y la monitorización de la temperatura en cada una de las cámaras frigoríficas es de suma importancia, especialmente en el sector de los congelados, donde es imprescindible controlar en todo momento el estado de conservación y la calidad de un producto.

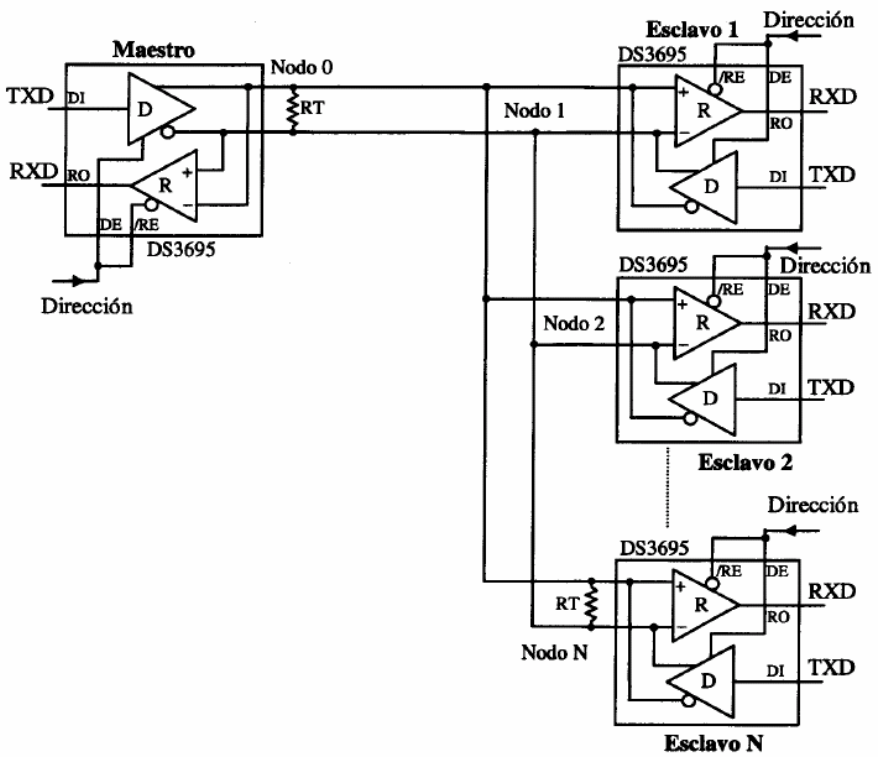
En la figura siguiente se propone una red de comunicación basada en la norma RS-485, para realizar el control y la monitorización entre un ordenador central que hace la función de "maestro" y las cámaras frigoríficas que hacen la función de "esclavos". En esta aplicación se ha escogido la norma RS-485 porque es un sistema de transmisión diferencial que permite la comunicación bidireccional de hasta 32 equipos en un bus de datos común, permite la transmisión a gran distancia y a gran velocidad, y los emisores deben incorporar autoprotección contra la sobrecarga de acceso (es decir, la situación de tener múltiples emisores intentando acceder al mismo tiempo a la línea de transmisión).

Recordamos que las principales características de la RS-485 son:

- La máxima longitud de cable es de 1.200 metros.
- La velocidad máxima de transmisión es de 10Mbits/seg.
- Impedancia de entrada del receptor de 12 kohm.
- Margen de modo común de la entrada del receptor de -7V a +12V.
- Sensibilidad de la entrada diferencial de ±200mV en un margen de modo común de -7V a +12V.

La interfaz con el bus RS-485 se implementa con el circuito integrado DS3695 o el DS75176 de National Semiconductor, que tiene un transmisor y un receptor diferencial, cuyo estado se controla con las entradas DE y /RE, respectivamente.

Cada una de las cámaras frigoríficas tiene un circuito electrónico formado por un microcontrolador 87C51, cuatro dígitos de siete segmentos donde se visualiza la temperatura en grados centígrados de la cámara, un DS1620 de Dallas Semiconductor que hace de sensor de temperatura y de acondicionador de señal, y un DS3695 para la interfaz RS-485 (figura siguiente).

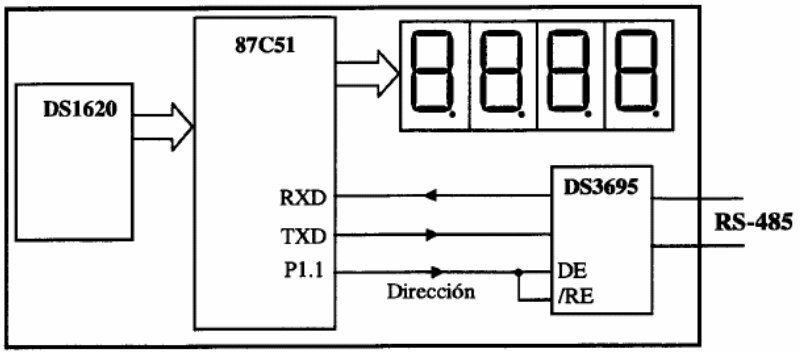


Para realizar la comunicación en la red se debe establecer un protocolo, en el que se definen una serie de reglas con las que se gestionará la comunicación entre las cámaras frigoríficas y el equipo maestro. Cada una de las conexiones en la red RS-485 se denomina nodo. Cada nodo puede, mediante un protocolo, enviar o recibir mensajes o bloques de datos de la red.

Las funciones que deberá realizar el circuito electrónico de cada una de las cámaras son monitorizar la temperatura de la cámara en el visualizador de cuatro dígitos de siete segmentos y transmitir la temperatura de la cámara al equipo maestro cada vez que éste se lo pida.

Para comprobar el estado de la red y de la interfaz con cada nodo, el equipo maestro enviará un comando y un dato a cada uno de los esclavos, lo que forzará al esclavo a transmitir el dato recibido al maestro. De esta forma el equipo maestro puede comprobar si hay alguna anomalía en la red o en la interfaz de un nodo determinado.

La red estará formada por un equipo central que hará de maestro y por veinticinco esclavos. El protocolo estará formado por tres bytes. El primer byte se corresponderá con el número del nodo destino de la transmisión. A cada nodo se le asignará un número: el nodo 0 se corresponderá con el nodo maestro y los nodos esclavos se numerarán consecutivamente como nodo 1, nodo 2, ..., nodo 25. El segundo byte del protocolo será el número del nodo que origina el mensaje. Y, por último, el tercer byte se corresponderá con un comando ejecutable por el nodo esclavo, que podrá ser el carácter ASCII DC1 (11H) o DC2 (12H). El comando DC1 indicará al nodo esclavo que debe transmitir los datos correspondientes a la temperatura al nodo maestro, y el comando DC2 indicará que se va a hacer un chequeo de la línea por parte del nodo maestro.



La lectura de la temperatura en cada microcontrolador estará almacenada en las posiciones 20H y 21H de la memoria interna. La posición 20H contendrá el byte de menor peso y la posición 21H el byte de mayor peso. En consecuencia, cuando un nodo esclavo reciba el comando DC1, le transmitirá al nodo maestro estos dos bytes correspondientes a la temperatura de la cámara frigorífica. El nodo esclavo, en este caso, transmitirá un primer byte correspondiente al número del nodo maestro (00H), un segundo byte que corresponde con el número del nodo esclavo y los dos bytes correspondientes al valor de la temperatura de la cámara.

En el caso que un nodo esclavo reciba el comando DC2 de chequeo de la red, éste pasará a la espera de un dato concreto que le debe transmitir el nodo maestro. Para ello, el nodo maestro transmite un primer byte, correspondiente al número del esclavo y un segundo byte que consiste en la constante AAH.

A continuación se muestra el programa que realiza la comunicación de un microcontrolador 87C51 como el nodo esclavo número 2 de la red. En este programa no se considera el control del DS1620 ni la visualización en los cuatro dígitos de siete segmentos de la figura anterior. La velocidad de comunicación se ha configurado para 19.200 baudios, y se ha programado el puerto serie en el modo asíncrono 1, al igual que en el ejemplo anterior. El Timer 1 está configurado en el modo 2 de 8 bits con autorrecarga, con el bit C/T a 0 lógico, con el bit SMOD a 1 lógico, con el valor FDH de recarga en el registro TH1 y con 11.059MHz de frecuencia de reloj.

```
/*************************************************************/
/* Programa comunicación bus RS-485 del segundo módulo esclavo */
/*************************************************************/
/* Declaración de constantes y de variables                  */
/*************************************************************/
#include <reg51.h>
#include <stdio.h>
#define uchar unsigned char

#define ID_MOD    0x02  /* Número de identificación de este módulo */
#define ID_Master 0x00  /* Número de identificación del maestro */
#define DC1       0x11  /* Valor del Comando DC1 */
#define DC2       0x12  /* Valor del comando DC2 */

uchar TX_data;
bit TX_en;
uchar data Orden_in;    /* Variable indica el nº de byte recibido */
uchar data Orden_tx;    /* Variable indica el tipo de transmisión */
uchar data Dato_Eco;    /* Contendrá dato enviado por maestro */
uchar Temp_bajo;  /* Contiene el byte bajo de la temperatura */
uchar Temp_alto;  /* Contiene el byte bajo de la temperatura */
```

```
/*****************************************************************/
/* Rutina de servicio del puerto serie                           */
/*****************************************************************/
serial_int () interrupt 4  using 1 {
uchar tmp;

if (RI) {               /* Si ha sido recepción */
      tmp=SBUF;
      switch (Orden_in) {
            case 0: {if (tmp==ID_MOD) Orden_in++;
                     break;}
            case 1: {if (tmp==ID_Master) Orden_in++;
                     else Orden_in=0;
                     break;}
            case 2: {
                  if (tmp==DC1) {
                        Orden_in=0;  /* Inicializa contador */
                        Orden_tx=1;  /* transmitir temperatura */
                        }
                  else if (tmp==DC2) Orden_in++;
                  else Orden_in=0;
                  }
                  break;
            case 3: {   Dato_Eco=tmp;
                  Orden_tx=0x02;
                  Orden_in=0;}
                  break;
            default: Orden_in=0;
            }
   RI=0;
  }
}

sbit SENTIDO=P1^1;  /* control del dispositivo de transmisión */

/*****************************************************************/
/* Subrutina 'Trans_temp" transmisión temperatura                */
/*****************************************************************/
void trans_temp(void) {
      SENTIDO=1;              /* Establece sentido transmisión */
      putchar(ID_Master);     /* Carga nº identificación del maestro */
      putchar(ID_MOD);        /* Carga identificación de módulo */
      putchar(Temp_alto);     /* Carga byte alto de temperatura */
      putchar(Temp_bajo);     /* Carga byte bajo de temperatura */
      SENTIDO=0;              /* Establece sentido recepción */
      }
/*****************************************************************/
/* Subrutina 'Trans_Eco" para Eco con el dato enviado por el 1C  */
/*****************************************************************/
void trans_eco(void) {
      SENTIDO=1;              /* Establece sentido transmisión */
      putchar(ID_Master);     /* Carga identificación del maestro */
      putchar(ID_MOD);        /* Carga identificación de módulo */
      putchar(Dato_Eco);      /* Carga el dato a enviar de Eco */
      SENTIDO=0;              /* Establece sentido recepción */
      }
```

```
/*****************************************************************/
/* Rutina Inicio (configuración de la comunicación a 19200 baudios) */
/*****************************************************************/

void main (void) {

      Orden_in=0;       /* pone a cero algunas variables */
      Orden_tx=0;
      SENTIDO=0;        /* Establece el sentido RS-485 en recepción*/

      SCON = 0x60;      /* Configura comunicación serie en modo 1 */
      TCON = 0x40;      /* TCON */
      TMOD = 0x20;      /* Timer 1 en Modo 2, GATE=0 y C/T=0 */
      TH1 = 0x0FD;      /* Valor para 9600 baudios */
      IE = 0x90;
      REN = 1;
      TI=1;

while (1) {
      if (Orden_tx==0x1) {
/* si se trata de enviar la temperatura lo hace */
            Orden_tx=0;
            trans_temp();
      } else if (Orden_tx==0x2) {
/* si se trata de enviar un eco el proceso es diferente */
            Orden_tx=0;
            trans_eco();
            }
      }
}
```

La rutina principal está continuamente pendiente de la operación con el nodo maestro, que viene indicada por la variable Orden-a. Esta variable puede valer 01H cuando se debe transmitir la temperatura leída por el microcontrolador, o bien 02H cuando debe transmitir el dato recibido por parte del nodo maestro.

Las entradas DE y /RE del DS3695 determinan el sentido de la comunicación con la red, y ambas están conectadas a la patilla P1.l (figura anterior), de manera que cuando P1.l está a 0 lógico el DS3596 está configurado para recibir datos y a 1 lógico para transmitir datos. El terminal P1.1 establece el sentido de la comunicación.

En la rutina de RSI del puerto serie se utiliza la variable Orden_in como contador de 0 a 3 que indica el orden del byte recibido por parte del nodo maestro. El protocolo establece que, para dirigirse al microcontrolador del ejemplo (nodo 2), el nodo maestro debe transmitir el dato 02H (destino del mensaje), el dato 00H (origen del mensaje) y un comando de orden que puede ser DCl o DC2. En el caso de recibir el carácter DC2, la rutina de RSI inicializa la variable Orden-in y pasa a la espera del dato que se debe transmitir como eco. En el caso de que el destino de la comunicación no sea este nodo, o bien que el origen de la comunicación no sea el nodo maestro, el proceso obviará los datos recibidos, y pasará a la espera de un mensaje del nodo maestro.

Ejemplo 3 Comunicación a través de interrupciones. Gestión de buffer de transmisión.

En este ejemplo, en lugar de usar las rutinas de transmisión de la librería de C estándar empleamos nuestros propios métodos de gestión del puerto serie añadiendo una zona de memoria (buffer) que almacena siguiendo un esquema fifo hasta 32 caracteres recibidos o listos para enviar. El programa principal que se ejecuta puede de esta forma independizarse del proceso de recepción/transmisión y puede "dejar" o "coger" cadenas de caracteres de los buffers mientras que la transferencia de bytes se lleva a cabo realmente por la interrupción.

La rutina loadmsg carga el array buffer y señala el comienzo de transmisión. Los buffers están gestionados mediante dos índices (in y out) y algunas banderas. Haciendo los buffers de 32 bytes de longitud, los índices se pueden manejar mediante simples operaciones lógicas AND, que resultan más rápidas que la operación de módulo (%). Cuando rbin=rbout, rbuf está lleno, y no se insertarán más caracteres. Cuando tbin=tbout, tbuf está vacío, y la interrupción de transmisión se detiene hasta que se soliciten más por parte de la UART.

Se puede realizar este sistema de buffers fifo de otras maneras, aunque aquí nos centramos en la interrupción del puerto serie.

```
#include <reg51.h>
#define uchar unsigned char
/* buffers que almacenan datos a transmitir
y datos recibidos en memoria externa de datos */
uchar xdata rbuf[32];
uchar xdata tbuf[32];

/* variables de manejo de fifos */
uchar rbin,rbout,tbin,tbout;
bit rfull,tempty,tdone;

/* cadena de texto en zona de codigo */
uchar code m1[]={"Esto es una prueba\r\n"};

/* Interrupción de puerto serie */
void serint(void) interrupt 4 using 1{
  /* si recibimos byte y no esta lleno el buffer
  lo introducimos en el buffer */
      if (RI && !rfull) {
            rbuf[rbin]=SBUF;
            RI=0;
            rbin=++rbin & 0x1F;
            /* si llegamos al final marcamos su llenado */
            /* otra funcion se encargara de procesar estos datos */
            if (rbin==rbout) rfull=1;
         }
      /* Si hemos terminado de transmitir byte
      transmitimos otro hasta tempty */
      else if (TI && !tempty) {
            SBUF=tbuf[tbout];
            TI=0;
            tbout=++tbout & 0x1F;
            /* Si ya no hay mas señalamos fin */
            /* otra funcion se encarga de colocar los
            nuevos datos a transmitir */
            if (tbout==tbin) tempty=1;
         }
      /* caso contrario ponemos a cero TI */
    else if (TI) {
```

```
            TI=0;
            tdone=1;
         }
}

/* Carga el buffer con un texto a transmitir
este esta situado en memoria de codigo */
void loadmsg (uchar code *msgchar) {
      while ((*msgchar != 0) &&
                  ((((tbin+1)^tbout) & 0x1F) != 0)) {
            /* mientras haya caracteres y el buffer no se llene */
                        tbuf[tbin]=*msgchar;
                        msgchar++;
                        tbin=++tbin & 0x1F;
                        if (tdone) {
                              /* comenzar transmisión si todo acabo */
                              tempty=0;
                              tdone=0;
                              TI=1;
                              }
                     }
        }

/* procesa caracteres recibidos de uno en uno
por ahora no hace nada */
void process (uchar ch) {return;}

/* procesa caracteres recibidos */
void processmsg(void) {
      while (((rbout+1)^rbin) != 0) {
            process(rbuf[rbout]);
            rbout=++rbout & 0x1F;
            }
}

/* programa principal, por ahora solo carga el buffer de salida
y procesa caracteres que llegan */
void main(void) {

      SCON = 0x60;      /* Configura comunicación serie en modo 1 */
      TCON = 0x40;      /* TCON */
      TMOD = 0x20;      /* Timer 1 en Modo 2, GATE=0 y C/T=0 */
      TH1 = 0x0FD;      /* Valor para 9600 baudios */
      IE = 0x90;
      REN = 1;

      tempty=tdone=1;
      rfull=0;
      rbout=tbin=tbout=0;
      rbin=1;
      /* bucle sin fin */
      for (;;) {
            loadmsg(&m1);
            processmsg();
    }
}
```