UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN

"DISEÑO E IMPLEMENTACIÓN DE
UN SISTEMA DE CONTROL PARA EQUIPOS DE ILUMINACIÓN
Y ACCESORIOS CON PROTOCOLO DMX512"

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A N:

LOPEZ DE LA ROSA ALFREDO

PRADO VARGAS RENE

ASESOR: ING. JORGE BUENDIA GOMEZ

CUAUTITLAN IZCALLI, EDO. DE MÉX. 2009

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Siendo el hombre un ser creativo por excelencia, la naturaleza para él no es solo un objeto de admiración, sino un reto de superación. A partir de la invención de la rueda, su progreso va en continuo avance y no contento con recorrer la tierra, ya pisó la luna y su inquietud palpa ya otros planetas.

"Lo que con mucho trabajo se adquiere, mas se ama"

Aristóteles

A MIS PADRES

A quienes sin escatimar esfuerzo alguno han sacrificado gran parte de su vida por formarme y educarme. A quienes la ilusión de su vida ha sido convertirme en una persona de provecho. Por esto y más…. Gracias

Gudelia y Amador.

A MI ESPOSA

Por el apoyo emocional brindado y las dosis de aliento para no sucumbir en los momentos de dificultad, en el tiempo que tarde en lograr este objetivo.

Fabiola.

A MIS HERMANOS

Ángel, Irma, Arturo, Jorge, Juan Carlos y Charly. A su sonido Liverpool y empresa ESDE audio por inspirarme a desarrollar el tema de tesis y por mostrarme que puedo ir mas haya de lo que me enseñaron.

A MI AMIGO

A mi compañero de tesis, que fue parte esencial y pilar en los ánimos y desarrollo de esto, al ser el último escalón para poder alcanzar este sueño, este MI SUEÑO, que ahora es una realidad.

Rene.

Alfredo López de la Rosa

A Dios. *Por estar conmigo y permitirme concluir una etapa mas en la vida.*

A mis Padres. *A quien la ilusión de su vida ha sido convertirme en persona de provecho. Gracias por todo su esfuerzo y sacrificio, pero sobre todo por su amor. Todo mi corazón para ustedes.*

Alberto Prado y Francisca Vargas

A mis Hermanos. *Con los que siempre podré contar en los buenos y malos momentos. Gracias por todo su apoyo.*

Ignacio Prado, Jorge Prado y Hugo Prado

A mis sobrinas. *Que llenan de luz y alegría mi vida*

Karen y Magdiel

A la UNAM. *Por permitirme formar parte de una gran institución. Es un enorme orgullo.*

A la FES.

A mis profesores

A mis compañeros. *Generación (1998-2002).*

DEDICATORIA

A toda mi familia

En especial a:
Ignacio prado +.
Te quiero y extraño

Rene Prado Vargas

INDICE

CAPITULO 1. CONTROL DE ILUMINACION DMX512

CAPITULO 2. PROTOCOLO DMX512

CAPITULO 3. MICROCONTROLADOR DE LA FAMILIA ATMEL

CAPITULO 4. COMUNICACIÓN SERIAL

CAPITULO 5. HERRAMIENTAS DE DESARROLLO

CAPITULO 6. DISEÑO E IMPLEMENTACION DEL SISTEMA DE CONTROL

CAPITULO 7. INTERFAZ GRAFICA DE USUARIO

CAPITULO 8. MANUAL DE OPERACIÓN

CAPITULO 1. CONTROL DE ILUMINACION DMX512

CAPITULO 2. PROTOCOLO DMX512

CAPITULO 3. MICROCONTROLADOR DE LA FAMILIA ATMEL

CAPITULO 4. COMUNICACIÓN SERIAL

CAPITULO 5. HERRAMIENTAS DE DESARROLLO

CAPITULO 6. DISEÑO E IMPLEMENTACION DEL SISTEMA DE CONTROL

CAPITULO 7. INTERFAZ GRAFICA DE USUARIO

CAPITULO 8. MANUAL DE OPERACIÓN

INTRODUCCION

En la industria escenografica han existido diferentes empresas fabricantes de consolas y dimmers para usos escenograficos, lo que genero sistemas de comunicación y transmisión de señales diferentes, esto ocasiono una gran incompatibilidad entre marcas. Además, cada fabricante, no solo utilizaba conectores propios; sino mecanismos opuestos, con la idea desacertada de eliminar a la competencia y lograr atrapar al consumidor. Con la llegada de la digitalización se acentúo más el problema y surge la necesidad de estandarizar los sistemas.

En respuesta a la necesidad de estandarizar un sistema de comunicación único para la industria escenografica. En 1986 se desarrollo el standard AMX192 y con una posterior revisión en 1990 surge el standard DMX512 y para el 2004 es aprobado por la ANSI como “E1.11, USITT DMX512–A", o solo "DMX512-A", y es mantenido por la ESTA (Entertainment Services & Technology Association), para convertir el sistema de comunicación entre consolas y dimmers en un estándar capaz de satisfacer las necesidades de los sistemas de iluminación requeridos en los campos escenográficos, tales como: auditorios, foros, escenarios móviles de grupos, djs, etc; de una forma eficiente y confiable.

El campo de la iluminación escenografica se hace tan extenso, que los equipos diseñados para cumplir con necesidades especificas de trabajo ya no satisfacen los requerimientos de uso para otra aplicación parecida, por lo que es frecuente que nos veamos en la necesidad de implementar y/o adaptar circuitos electrónicos que realicen las funciones o tareas específicas que cumplan los requerimientos adicionales sin interferir en las funciones para las cuales fue diseñado el equipo.

En la actualidad a pesar de contar con equipos de control dedicados de manera especifica a realizar numerosas tareas; también se ha requerido llevar acabo el control de estas a través de un PC de escritorio o portátil, ya sea por medio del mismo equipo que incorpore un sistema de comunicación para enlazarse con una PC o que se disponga de otro dispositivo adicional.

El presente trabajo pretende realizar el diseño e implementación de un sistema DMX512 que incorpore una alternativa adicional de control. Contara con entrada/salida de información, almacenamiento de datos y enlaces de comunicación. Además de una aplicación de software en PC con la que el usuario pueda interactuar a través de una interfaz con el control DMX512.

Descripción de los capítulos.

El trabajo de tesis esta dividido en 8 capítulos:

En el cap. 1 Se describirá el uso del Standard dmx512 en la iluminación escenografica y se mostraran los términos empleados, dispositivos, tipo de cables y elementos necesarios para realizar las conexiones adecuadas que cumplan los requerimientos exigidos por el Standard.

En el cap. 2 se desglosaran los elementos que componen el formato del protocolo DMX512, por el cual se transmitirán los datos a los dispositivos. Es una parte indispensable para poder llevar acabo la implementación de este proyecto.

En el cap. 3 A fin de poder llevar acabo el desarrollo de diseño e implementación es necesario retomar algunos conceptos teóricos en microcontroladores y comunicación serial. En este capitulo se describirá los aspectos fundamentales de los microcontroladores y su importancia en los sistemas digitales de control. Este punto se referirá en particular al microcontrolador AT89C52 de la familia ATMEL.

En el cap. 4 Se mencionará la importancia de la comunicación serial en los sistemas digitales y los elementos que intervienen, así como los estándares mas empleados en la industria. Se emplearan los estándares rs-232 y rs-485 en el diseño, para la comunicación con la PC y con los accesorios de iluminación respectivamente.

En el cap. 5 Explicaremos las herramientas de desarrollo empleadas para conseguir nuestro objetivo. Se describirá el uso de KEIL para la edición, depuración y simulación de código ensamblador para la programación del AT89C52, así como la aplicación Ez-downloader para realizar la descarga directa al microcontrolador del código maquina obtenido con KEIL. Este capitulo también abarca la descripción y uso de visual Basic para poder realizar la aplicación en PC

En el cap. 6 Se definirán la forma de conexión de las líneas de control, datos y alimentación de cada uno de los elementos como: teclado, display, memorias y C.I. empleados en el diseño e implementación del sistema.

En el cap. 7 Se desarrollara la interfaz grafica con la cual el usuario pueda interactuar para poder realizar el enlace con el sistema de control, por medio de un PC.

En el cap. 8 se pretende mostrar el sistema final completo por medio de una manual de usuario, describiendo: sus funciones, las conexiones entre los elementos y la forma operativa.

CAPITULO 1

1 CONTROL DE ILUMINACIÓN DMX512

En este capitulo explicaremos los conceptos relacionados con el protocolo DMX512, así como: dispositivos, cables y elementos necesarios para realizar las conexiones adecuadas que cumplan los requerimientos exigidos.

1.1 Descripción

El protocolo **DMX512** se ha establecido ahora como el Sistema de Gestión Escenografica numero uno para aplicaciones dinámicas de iluminación. DMX512 **(DIGITAL MULTIPLEX 512)** fue originalmente pensado para su uso en controladores a nivel de iluminación (dimmers), pero pronto se convirtió en el protocolo preferido no sólo para control de dimmers, sino también para controlar equipos de iluminación como scanners, cabezas móviles, y dispositivos de efectos especiales. Como los que se ilustran en la Figura. 1.1

Figura. 1.1. Accesorios de iluminación

1.1.1 Universo DMX

El protocolo DMX512 se basa en la utilización de canales para transmitir órdenes de control a los aparatos que lo soporten. El número máximo de canales que soporta el protocolo es de 512 canales, el cual forma un universo DMX512 o también conocidos como escena. Las mesas profesionales más comunes que usan DMX pueden soportar hasta 8 universos DMX.

1.1.2 Canales DMX

Generalmente cada canal DMX controla un efecto específico del aparato a controlar. De esta manera para una lámpara del tipo par64, por ejemplo, el canal DMX 1 servirá para controlar el nivel de intensidad luminosa, el canal DMX 2 para controlar el efecto estrobo de la misma y el canal DMX 3 para el cambio de colores.

Dispositivos más complejos, tales como cabezas móviles, scanners, o máquinas de humo requieren de mayor cantidad de canales DMX al tener más funciones las cuales pueden ser controladas independientemente, que pueden variar desde 4 canales hasta mas de 16 canales.

1.1.3 Valores DMX

El valor dentro de un canal DMX varía en el rango contenido por 8 bits el cual representa un cambio a razón de 000 hasta 255 valores.

Un foco de luz convencional controlado a través de un dimmer o regulador con soporte para DMX utiliza generalmente un canal DMX ya que sobre lo único que tendríamos control es la intensidad luminosa. Así pues, el valor DMX 000 generalmente significará que la intensidad del foco estará en su más bajo nivel: apagado o al 0%, y el valor DMX 255 de el mismo esté en su máximo nivel: encendido o al 100%. Las reacciones al comando DMX varían considerablemente de acuerdo con el aparato en operación y sus características.

1.1.4 Tipo de Cable

El DMX512 utiliza un cable con dos conductores par trenzado, llamado en inglés Twisted pair. Si las señales transmitidas son diferenciales (de polaridad opuesta), tal pareja aumenta notablemente la inmunidad a los disturbios. La elección del cable no debe ser descuidada. Existen en el mercado varios tipos diferentes en dimensiones generales, en secciones, con aislamiento y con revestimiento externo. Para las aplicaciones live (en vivo) se aconseja utilizar un cable con un revestimiento robusto pero bastante blando. En cambio para las instalaciones fijas es posible utilizar un cable un poco rígido; lo importante es que corresponda a las características indicadas

por el estándar EIA RS-485, como se aprecia en la Tabla 1.1. El cable de un par debe tener una baja capacidad por metro, una impedancia entre 100 y 150 Ohm, una defensa externa con una funda metálica integral, una defensa interna con hoja de mylar (poliéster transparente) y una sección mínima de 24AWG (0,5mm). Normalmente este grupo de línea es lo suficientemente inmune a los disturbios. La línea DMX puede funcionar bien hasta con un par trenzado para micrófono o telefónico, pero esto no garantiza el que no pueda dejar de funcionar en cualquier momento. Las instalaciones y situaciones en las que cables no idóneos continúan funcionando por años pueden y seguramente darán problemas cuando uno menos se los espera, entonces nos daremos cuenta de que el cable convalidado cubre las posibles causas de interferencia que pueden verificarse hasta ocasionalmente y no sólo en determinadas circunstancias, mientras un cable idóneo no lo hace.

El cableado y los conectores necesarios deben tener las siguientes características:

El Cable debe ser el conocido como PAR TRENZADO (Twisted pair) de un mínimo de impedancia de 100 / 150 Ohm, una sección mínima de 0,5 mm, una defensa interna con hoja de mylar y una funda metálica integral. En cuanto al revestimiento externo es importante contar con un material lo más flexible posible, ya que esto impedirá que nuestro cable de señal oponga resistencia al movimiento, traslado, instalación, etc., especialmente en equipamientos de renta o servicio de iluminación. Figura 1.2 y Tabla.1.1

Figura. 1.2. Cable par trenzado, empleado en RS-485

SISTEMA	CARACTERÍSTICAS
Norma	**RS-485**
Tensión Máxima	**30 – 300volt**
Sección/Calibre	**24 AWG**
Impedancia Ohm	**120**
Capacitancia pF/m	**47.8**
Resistencia Eléctrica	**88**

Tabla 1.1. Características del conductor.

1.1.5 Configuración del Cable

Para realizar las conexiones necesarias entre la consola DMX, las luminarias y accesorios; se requiere usar cable par trenzado, descrito anteriormente, y conectores XLR tipo cannon de 3 pines usados comúnmente. A continuación se muestra en la Figura 1.3, la configuración del cable con respecto a los pines del conector y tipos de conectores en un cable armado.

Figura 1.3. Conectores XLR tipo CANNON

1.1.6 Configuración de Conexiones DMX

Los aparatos DMX (scanners, cabezas móviles, etc.) deben estar conectados antes de la puesta en marcha de la consola DMX para que las unidades de recepción y de decodificación de

los aparatos DMX reconozcan el protocolo. Los aparatos están conectados vía un cable con conector XLR 3 polos o 5 polos dependiendo el estándar americano o europeo respectivamente (En este trabajo se hará referencia únicamente al estándar Americano). La conexión inicia en la consola con un conector XLR hembra el cual se conecta por medio de un cable par trenzado al conector XLR macho del dispositivo DMX a controlar.

La salida DMX de un aparato puede respectivamente estar conectada de nuevo con la entrada DMX de un segundo aparato. Figura 1.4

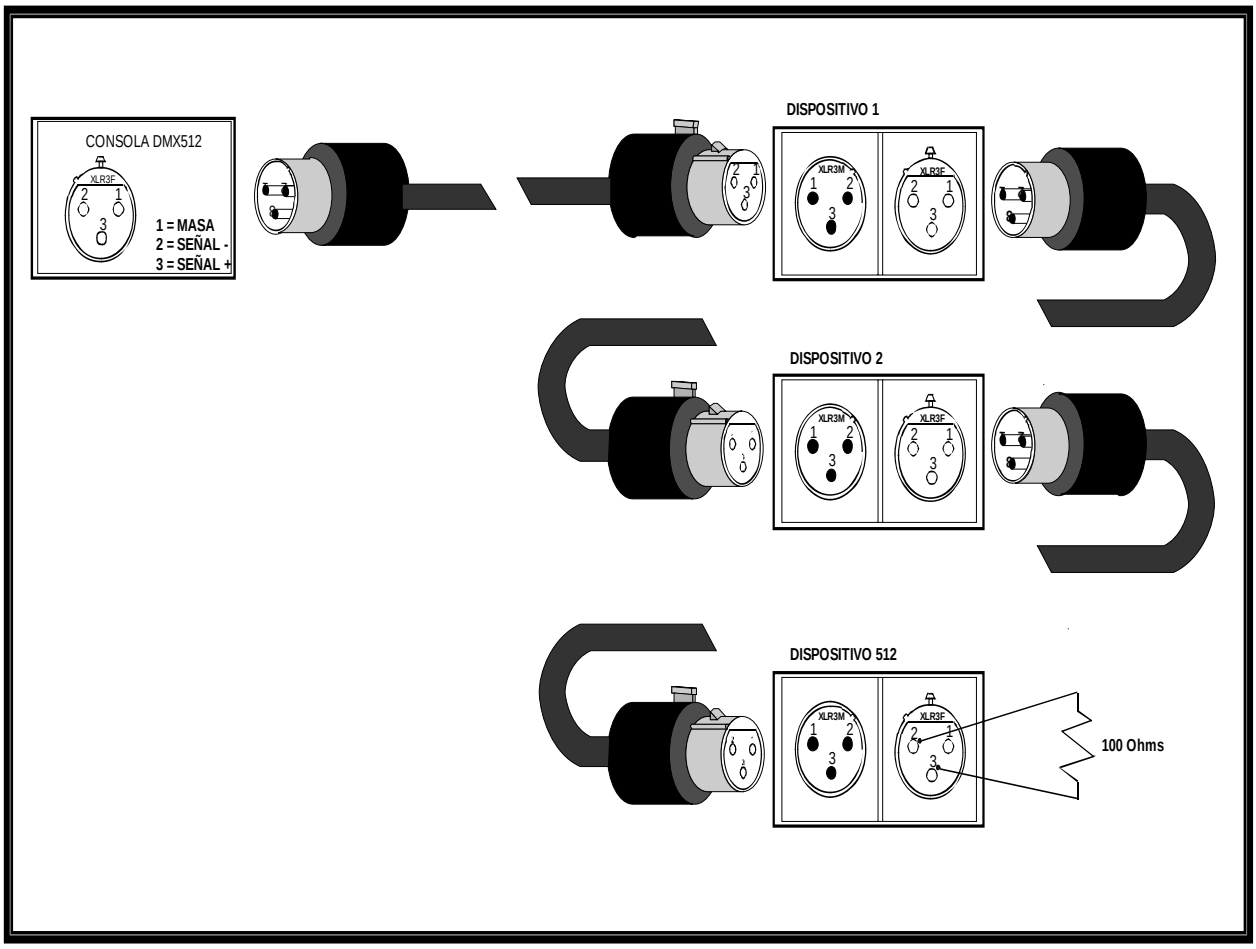


Figura 1.4. Conexión entre consola y dispositivos DMX512

1.1.6.1 Splitter y Buffer

Cuando se requiera realizar una distribución de señales de control, se emplean los splitters. En la distribución de los cables DMX las ramificaciones tipo **Y** están prohibidas y son extremadamente peligrosas porque degradan notablemente la calidad de la señal y vuelven poco estable la transmisión. Para efectuar una ramificación de tipo Y es necesario utilizar un splitter.

Los splitters son amplificadores de señal dmx512 múltiples que permiten efectuar una ramificación de tipo Y, además de ramificaciones con más salidas. Por otro lado, reacondicionan la señal permitiendo prolongar la distancia de utilización. Figura 1.5

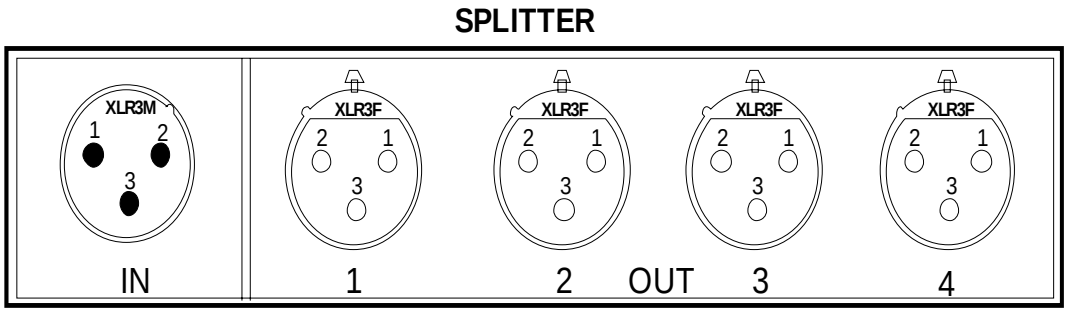


Figura 1.5. Splitter 1x4

Los buffers tienen una entrada y una salida y su función es amplificar y reacondicionar la señal para permitir un prolongamiento de la distancia de utilización sin la posibilidad de conexiones de tipo Y. Los splitters y los buffers pueden ser opto aislados o no. Los opto aislados son mejores porque además de sus características ya descritas permiten resolver el problema relativo a eventuales malos funcionamientos causados por los indeseables anillos a tierra. Por esto son muy utilizados para aislar entre ellas dos o más líneas. Figura 1.6

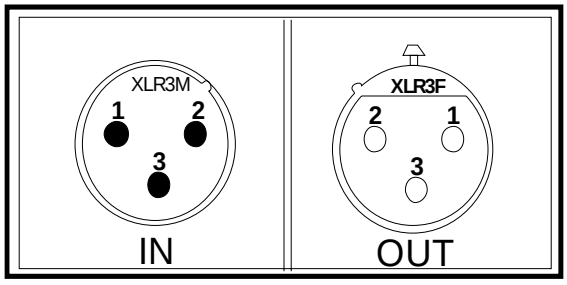


Figura 1.6. Buffer

En líneas muy largas se utilizan los buffers para amplificar y reacondicionar la señal. En algunas instalaciones es valido el uso de splitter y buffer por la distribución y separación entre dispositivos como se observa en Figura 1.7

Es fundamental colocar al final de cada línea una resistencia de terminación de red con valor entre 80 a 120 Ohms.

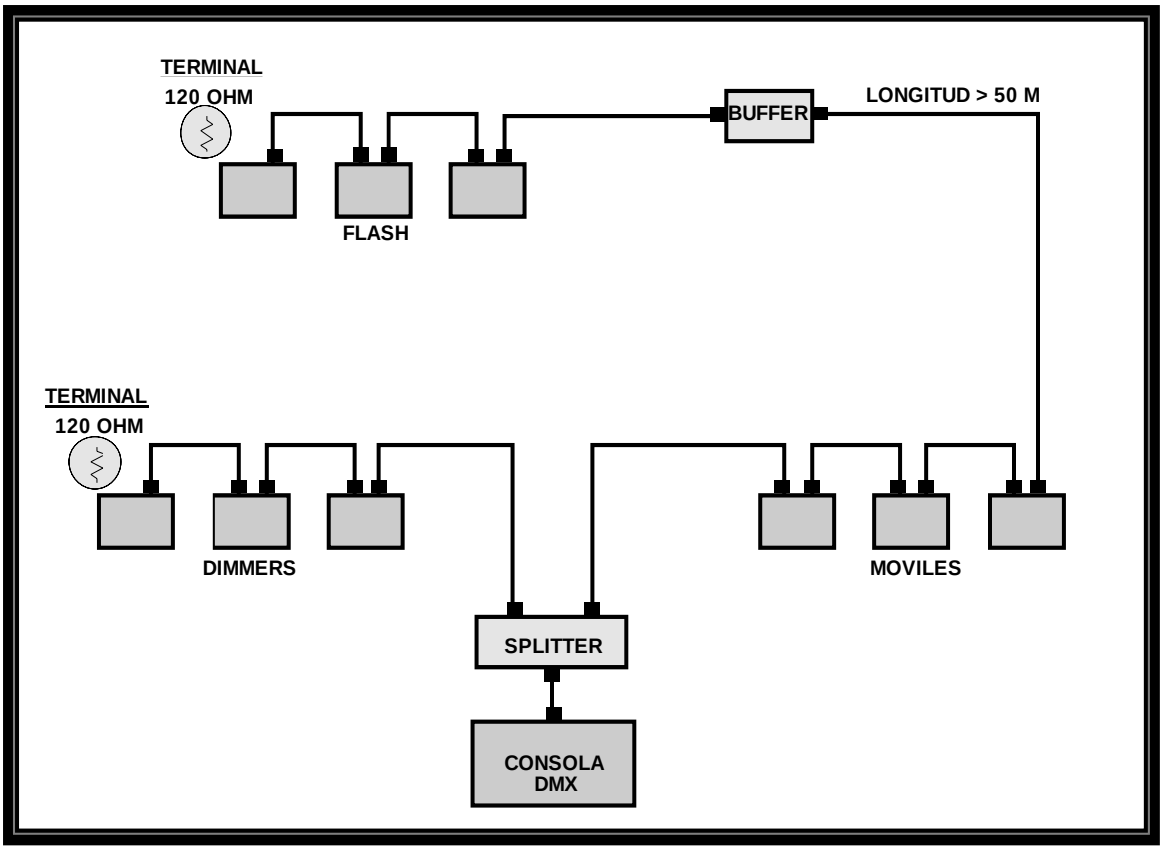


Figura 1.7. Sistema escenografico DMX512

1.1.7 Direccionamiento DMX

Podemos encontrar 2 formas de asignar las direcciones DMX a nuestros equipos dependiendo del dispositivo que sea.

1) En la gran mayoría de los dispositivos económicos, cuentan con un direccionamiento a base de un DIP-Switch. Este se debe encontrar en una parte visible y externa del dispositivo, para que el usuario pueda modificar su selección. Figura 1.8

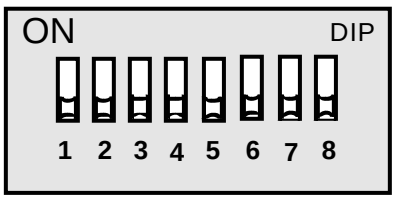


Figura 1.8. Dip 8 switch

2) Los más modernos, cuentan con display digitales. Esto con el fin de facilitar a los usuarios el direccionamiento de cada luminaria. Figura 1.9

Figura 1.9. Dispositivo con direccionamiento digital

En regla general, las direcciones de los aparatos están reguladas en los aparatos a través de interruptores DIP. La disposición de la dirección se efectúa en la mayoría de los casos por la ponderación de valores en formato binario. Ejemplo: un escáner tiene 4 funciones (movimiento X/Y, elección de los colores, de los gobos o figuras proyectables) y su dirección de arranque debe estar en el canal 13, Figura 1.10. El interruptor DIP del scanner DMX debe tener la siguientes posiciones: interruptor 1, 3 y 4 en "ON", todos los demás interruptores quedan en "OFF". Para este scanner, las direcciones 13-16 quedan asignadas a sus funciones.

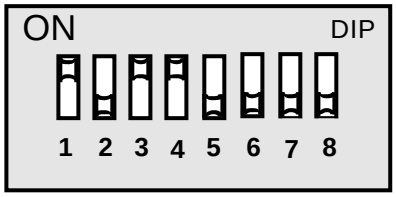


Figura 1.10. Direccionamiento del dip-switch canal 13

Los dispositivos DMX conectados pueden compartir las mismas direcciones, sin que se vean afectados; esto se debe a que el equipo conectado simplemente lee el valor contenido en el canal y dependiendo si los equipos son diferentes interpretaran el valor de forma distinta, a aquellos que son iguales.

CAPITULO 2

2. PROTOCOLO DMX512

En este capitulo se desglosaran las características y los elementos que componen el formato del protocolo DMX512, es decir cada una de las partes que lo integran. Es importante entenderlo para poder llevar a cabo la implementación en nuestro sistema de control y establecer la comunicación adecuada con aquellos dispositivos que ya cuentan con el protocolo. El protocolo se representa como se muestra en la Figura 2.1 y Tabla 2.1

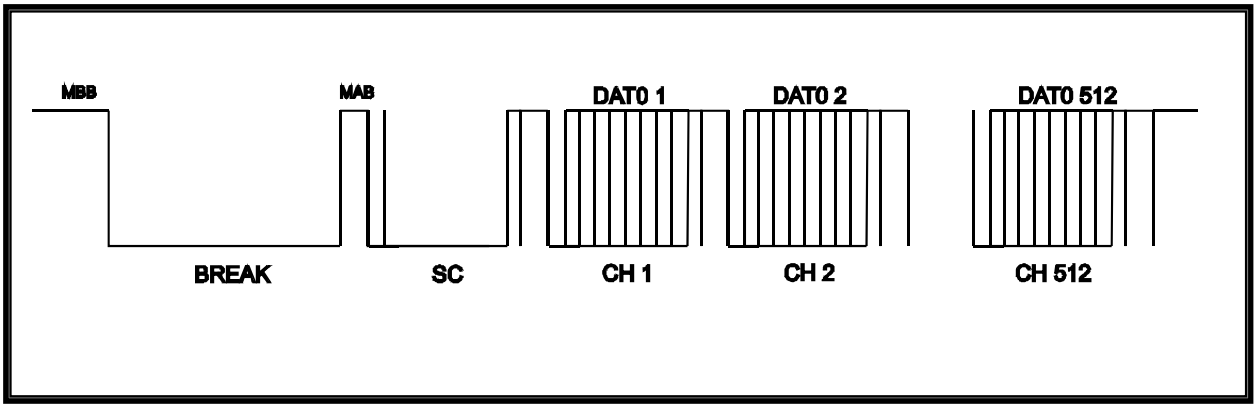


Figura 2.1. Formato de Protocolo DMX512

DESIGNACION	DESCRIPCION	MIN	TIPICO	MAX	UNIDAD
-	Velocidad de Transmisión	245	250	255	Kbit/s
-	Tiempo de BIT	3.92	4	4.08	μs
-	Tiempo mínimo de actualización para 512 canales	-	22.7	-	ms.
-	Rango mínimo de actualización para 512 canales	-	44	-	μs
1	Duración del Break	88	-	-	μs
2	Mark After Break (MAB)	8	-	<1.00	μs s
9	Duración entre Canales	0	-	<1.00	S
10	Mark Before Break (MBB)	0	-	<1.00	S
11	Tiempo entre BREAK-BREAK	1196	-	- <1.00	μs
13	Paquete DMX512	1196	-	- <1.00	μs

Tabla 2.1. Tiempos establecidos por el Protocolo DMX512

2.1 Formato

Los datos transmitidos serán en formato serial asíncrono. Los canales DMX512 serán transmitidos secuencialmente, empezando por el canal 1 y terminando con el último canal implementado, hasta un máximo de 512. Antes de transmitir el primer canal de datos, se transmitirá una secuencia de reset. Esta secuencia de reset se forma por un BREAK, seguido por un MAB y un SC, en apartados siguientes se describe cada uno de ellos. Los valores validos contenidos en los canales DMX512, serán de 0 a 255 decimal.

2.2 Formato de Canal

El formato de la transmisión de datos para cada valor del canal transmitido será el siguiente: Figura 2.2 y Tabla 2.2

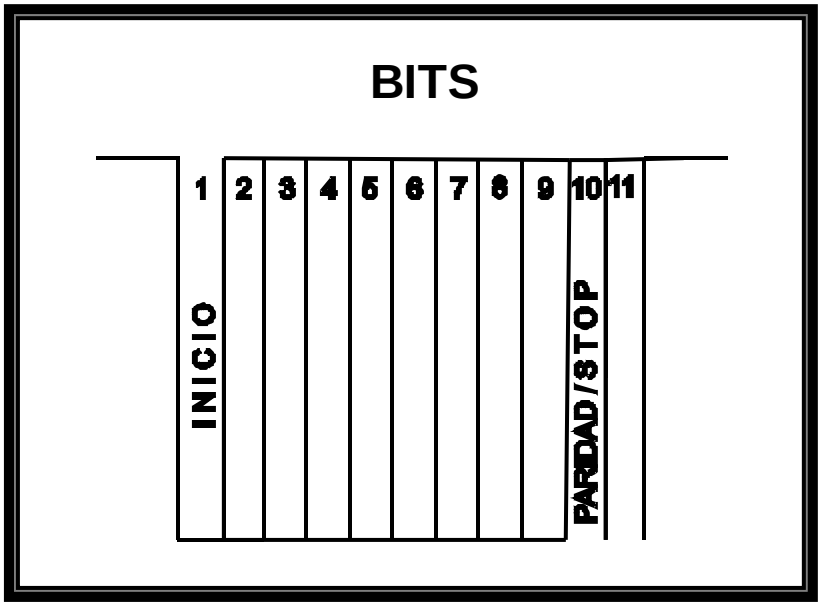


Figura 2.2. Formato de envío de dato

POSICION DE BIT	DESCRIPCION
1	Bit de inicio, Nivel bajo
2 a 9	Valores de bits de datos. Lógica positiva, de los bits menos significativos a los más significativos.
10 , 11	Bit de parada. Nivel alto
Paridad	No transmitido

Tabla 2.2. Descripción de formato en bits para el envío de dato

2.3 Breaks (BREAK)

El BREAK indica el comienzo de un nuevo paquete. Se compone de la transición de un nivel alto a bajo, permaneciendo en un nivel bajo durante 88 microsegundos mínimo o de mas duración, sin exceder 1s, seguido por una transición de nivel bajo a alto. Figura 2.3

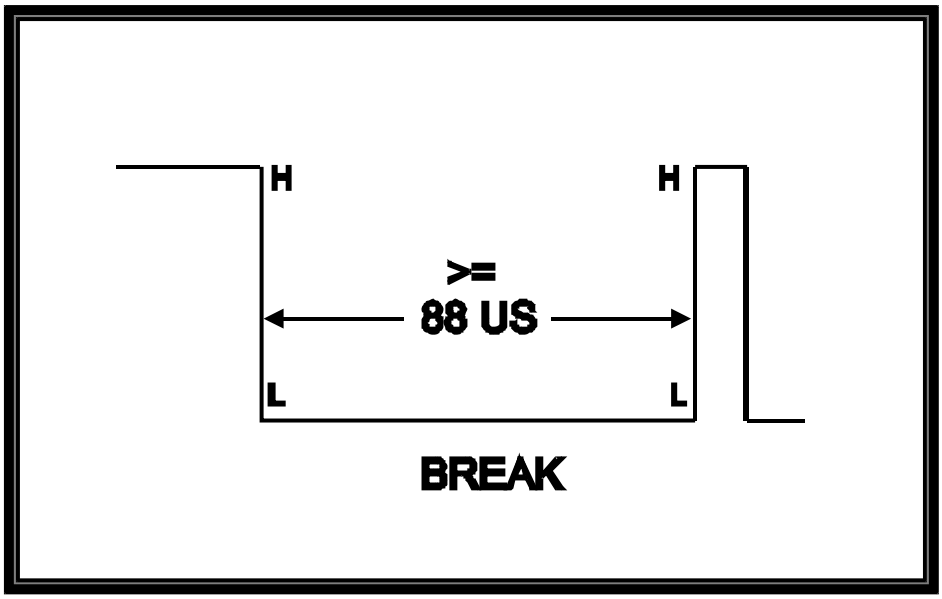


Figura 2.3. Duración del Break

2.4 Mark After Breaks (MAB)

Indica el envío del Start Code (SC). Es la marca que separa el fin del BREAK y el inicio del SC, la duración no será inferior a 8 microsegundos ni superior a 1 segundo. Todos los transmisores DMX512, deberán presentar un MAB de no menos de 8 microsegundos y los receptores deberán reconocer un MAB igual a 8 microsegundos. Figura 2.4

Nota: La versión de 1986 de esta norma especifica un MAB de 4 microsegundos. La versión 1990 de la norma cambio el valor a 8 microsegundos, pero añadió una opción para los receptores capaces de reconocer la marca de 4 microsegundos después del BREAK para tener la capacidad de ser identificado. Algunos transmisores pueden seguir generando 4 microsegundos de MAB, y ellos no pueden trabajar con el equipo construido con esta norma.

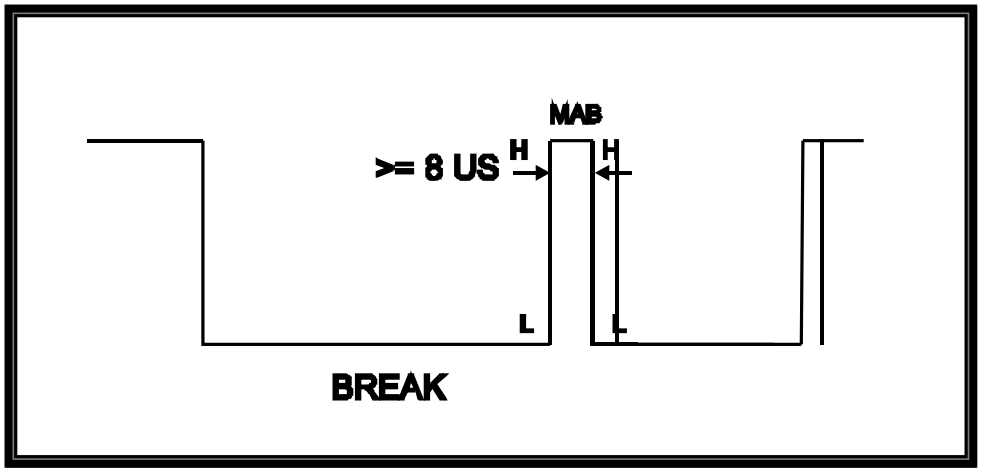


Figura 2.4. Duración del MAB

2.5 Start Code (SC)

Es el primer byte de información después del MAB. Este primer dato identifica la función de los datos posteriores a ese paquete, contiene el código de inicio NULL. Figura 2.5

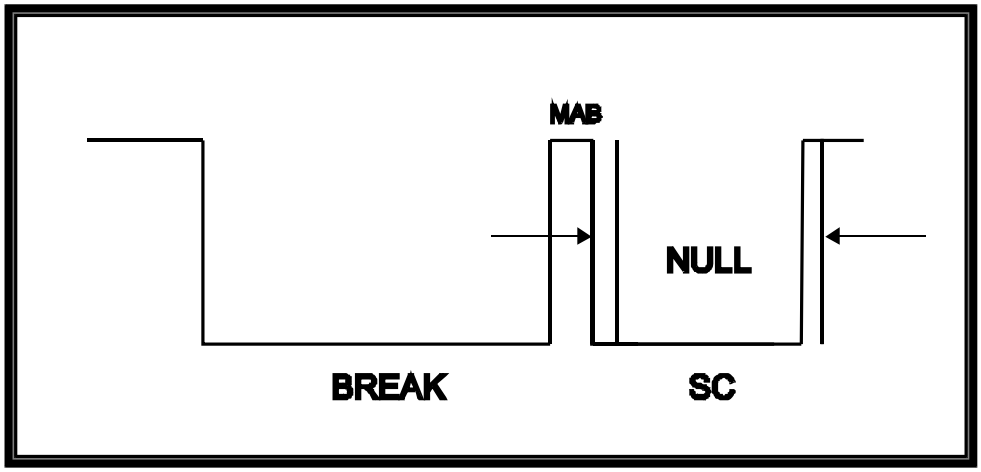


Figura 2.5. Código de inicio NULL

2.5.1 Código de Inicio NULL

El código de inicio NULL (un byte nulo – todos ceros) identifica los datos subsiguientes como secuencial información de 8 bits, pertenecientes al protocolo DMX512.

2.5.2 Otros Códigos de Inicio

Con el fin de proporcionar para la futura expansión y flexibilidad del protocolo DMX512, se prevé 255 códigos de inicio distintos del NULL (1 a 255 decimal, 01 a FF hexadecimal), en adelante se referirá como un código de inicio alterno. Cuando es requerido enviar información propia sobre un enlace de datos DMX512, un paquete inicializara con un código de inicio alterno registrado. Figura 2.6

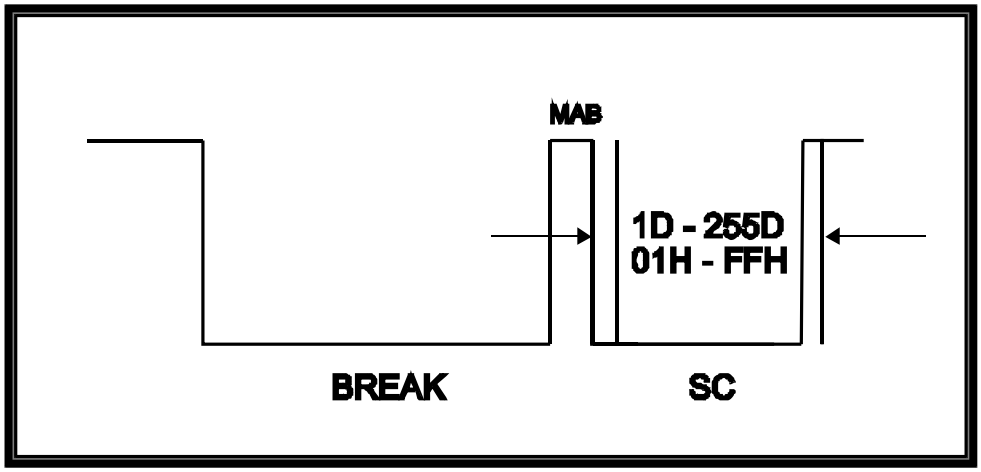


Figura 2.6. Códigos de inicio alternativos

2.5.3 Procesamiento del código de Inicio

Todos los dispositivos receptores y otros dispositivos procesadores en la línea, procesaran el código de inicio y distinguirán entre los paquetes con código de inicio NULL y los paquetes con código de inicio alterno. Los dispositivos DMX512 no deberán hacer caso omiso de los códigos de inicio, suponiendo que todos los paquetes recibidos sean paquetes de código NULL, por lo que se tendrá que identificar el código de inicio entre cada paquete de datos.

2.6 Numero Máximo de Canales de Datos

Cada enlace de datos deberá soportar hasta 512 canales de datos. Múltiples conexiones serán usadas cuando un número mayor de dispositivos sean requeridos, empleando splitter o buffer. Figura 2.7

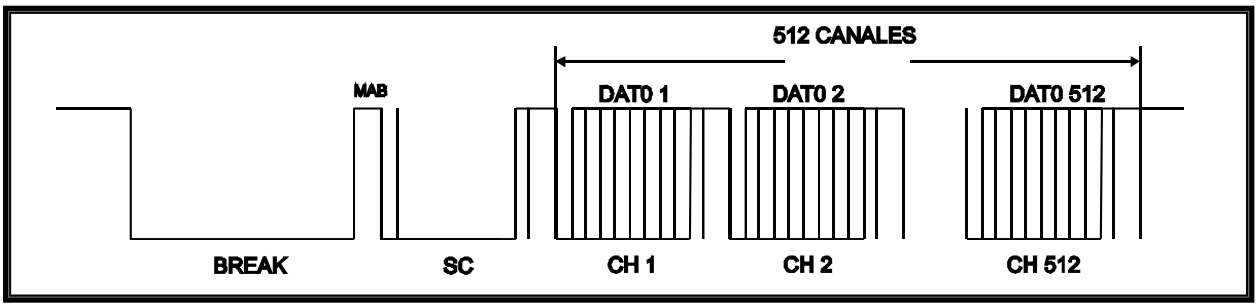


Figura 2.7. Máximo numero de canales soportados por el protocolo dmx512

2.7 Numero mínimo de Canales de Datos

No habrá número mínimo de canales de datos sobre la transmisión de datos. Los paquetes de datos DMX512 con menos de 512 canales pueden transmitirse, pero están sujetos a los requerimientos de tiempo mínimo de esta norma.

2.8 Duración entre Canales

El tiempo entre dos canales de un paquete de datos puede variar entre 0 y 1 segundo. La línea debe permanecer en un estado alto durante cualquier periodo de inactividad. Un receptor debe ser capaz de aceptar un paquete de datos que no tenga el tiempo de inactividad (0 s) entre cualquiera de los canales. Figura 2.8

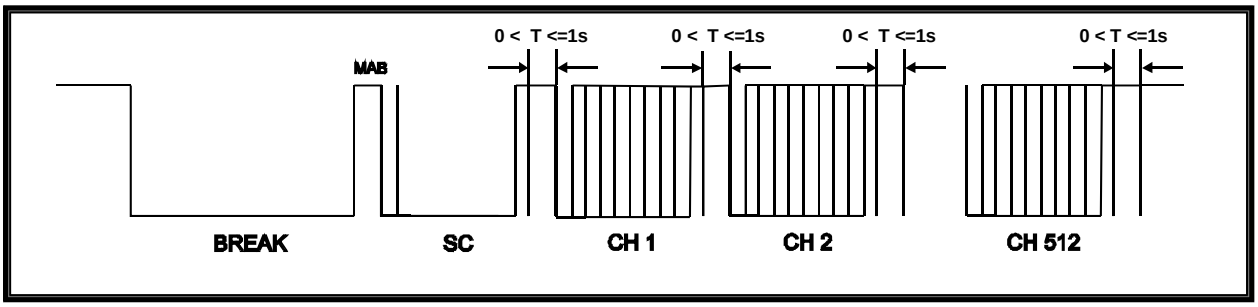


Figura 2.8. Tiempo entre canales

2.9 Mark Before Breaks (MBB)

Cada paquete de datos transmitido sobre el enlace de datos, independiente del código de inicio o la duración, debe comenzar con un BREAK, MAB y la secuencia de SC, tal como se definió. El tiempo entre el segundo bit de stop del ultimo canal de datos de un paquete y el

borde de la caída del inicio del BREAK para el próximo paquete de datos puede variar entre 0 y 1 segundo. La línea se mantendrá en un estado alto de inactividad durante todo ese periodo mayor a 0 segundos. Los transmisores, por lo tanto, no producirán múltiples pausas entre los paquetes de datos. Los receptores, sin embargo, serán capaces de recuperarse de múltiples BREAK producidos por errores en la línea de enlace de datos. Figura 2.9

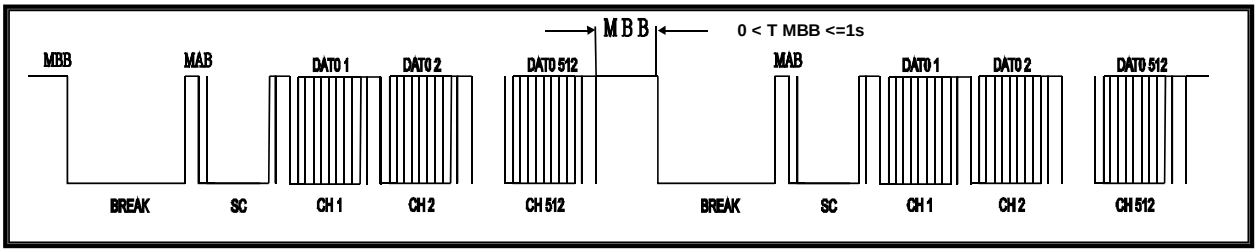


Figura 2.9. Tiempo antes del Break, después del último dato enviado

2.10 Duración BREAK-a-BREAK

El periodo comprendido entre el borde de la caída al inicio de cualquier BREAK no será inferior a 1196 microsegundos de la caída del comienzo del próximo BREAK, ni mas de 1.025 segundos. Como se ilustra en la Figura 2.10.

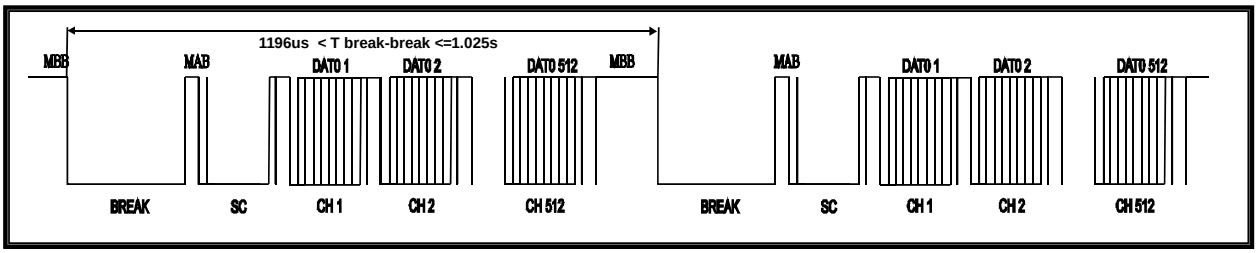


Figura 2.10. Tiempo que tarda en aparecer de nuevo el Break

2.11 Clase de Datos en Dimmer

Los niveles validos en dimmers serán de 0 a 255 decimal (00 a FF hexadecimal) que representan la entrada de control del Dimmer. 0 representa una salida del dimmer en OFF o un mínimo, y 255 representa una salida en ON o máximo. Un dimmer responderá a incrementos en los valores del canal DMX512 de 0 a 255 por desvanecimiento de su nivel mínimo (OFF) o su nivel máximo (ON). La relación exacta entre valores en canales DMX512 y salida del dimmer esta más allá del alcance de esta norma, esto depende de cada fabricante.

CAPITULO 3

3. MICROCONTROLADORES DE LA FAMILIA ATMEL

A fin de poder llevar acabo el desarrollo de diseño e implementación del sistema de control (proyecto) es necesario retomar los conceptos teóricos de microcontroladores, bajo el cual se intenta gestionar el control. En este capitulo se describirá los aspectos fundamentales de los microcontroladores y su importancia en los sistemas digitales de control. Y se referirá en particular al microcontrolador AT89C52 de la familia ATMEL.

3.1 Introducción a los Microcontroladores

Las circunstancias con las que nos encontramos hoy en el campo de los microcontroladores tienen sus raíces en el desarrollo de la tecnología de los circuitos integrados. Este desarrollo ha hecho posible contener cientos de miles de transistores en un solo chip. Ése era uno de los requisitos previos para la producción de los microprocesadores. Las primeras computadoras eran hechas empleando microprocesadores y agregando periféricos externos como la memoria, timers etc. lo que aumentaba el volumen de los circuitos integrados. Así es cómo se desarrollo el primer chip que contenía una microcomputadora, o lo que después se llegaría a conocer como un **microcontrolador**.

En el año 1969, un equipo de ingenieros japoneses de la compañía BUSICOM llegó a Estados Unidos con la idea de reducir la cantidad de circuitos integrados que se usaban en las calculadoras. La proposición se hizo a INTEL, y Marcian Hoff era el responsable del proyecto. Ya que él era quien tenía experiencia trabajando con una computadora PDP8, se le ocurrió pensar en una solución diferente en lugar de la sugerida. Esta solución presumía que la función del circuito integrado se determinaría por un programa almacenado en él. Eso significaba que la configuración sería más simple, pero que requeriría mucho más memoria de lo que requería el proyecto que propusieron los ingenieros japoneses. Después de un tiempo, aunque los ingenieros japoneses probaron soluciones más fáciles, la idea de Marcian ganó, y el primer microprocesador nació. Para transformar esta idea en un producto ya fabricado, Federico Faggin, se unió a INTEL, y en sólo 9 meses tuvo éxito. INTEL obtuvo los derechos para vender este "bloque integrado" en 1971. Durante ese año, apareció en el mercado un microprocesador que se llamó 4004, este fue el primer microprocesador de 4 bits con velocidad de 6 000 operaciones por segundo. No mucho tiempo después de eso, la compañía americana CTC pidió a INTEL y Texas Instruments que hiciera un microprocesador de 8 bits. Aunque después a CTC no le interesó mas la idea, Intel y Texas Instruments siguieron trabajando en el microprocesador y el primero de abril de 1972, el microprocesador de 8 bits aparece en el mercado con el nombre de 8008. Podía direccionar 16 Kb de memoria, con un set de 45 instrucciones y una velocidad de 300000 operaciones por segundo. Este microprocesador es el predecesor de todos los microprocesadores de hoy. Intel mantuvo sus desarrollos y saco al mercado el procesador de 8 bits bajo el nombre 8080, el cual podía direccionar 64Kb de memoria, con 75 instrucciones. En otra compañía americana, Motorola, comprendieron rápidamente lo que estaba sucediendo, así que ellos sacaron al mercado su microprocesador de 8 bits, el 6800. Su constructor principal era Chuck Peddle, y junto con el

procesador, Motorola fue la primera compañía en hacer otros periféricos como el 6820 y el 6850. En ese momento muchas compañías reconocieron la importancia de los microprocesadores y empezaron sus propios desarrollos. Chuck Peddle abandonó Motorola para unirse a la Tecnología MOS y se mantuvo trabajando intensamente en el desarrollo de los microprocesadores. Un evento muy importante tuvo lugar en la historia de microprocesadores en una exhibición de WESCON en 1975 en Estados Unidos. La Tecnología MOS anunció que estaba comercializando los microprocesadores 6501 y 6502 a 25 dlls. c/u. Esto era tan extraordinario, que algunas personas creyeron que era un escándalo, considerando que los competidores estaban vendiendo el 8080 y el 6800 a 179 dlls. c/u. Intel y Motorola bajaron sus precios en el primer día de la exhibición como una respuesta a su competidor, 69.95 dlls por microprocesador. Motorola reclama a la Tecnología de MOS y a Chuck Peddle el haberles copiado su 6800. La Tecnología MOS suspende la fabricación del 6501, pero siguen produciendo el 6502. Los 6502 eran microprocesadores de 8 bits, 56 instrucciones y la capacidad de direccionar 64Kb de memoria directamente. Para reducir el costo, el 6502 se vuelve muy popular, así que se instala en las computadoras tales como: KIM-1, Apple I, Apple II, Atari, Comodore, Acorn, Oric, Galeb, Orao, Ultra, y muchas otras. Y muy pronto aparecieron varios fabricantes del 6502 (Rockwell, Sznertek, GTE, NCR, Ricoh, y Comodore quienes toman la Tecnología MOS) el cual estaba en su momento de apogeo y se vendía a una velocidad de 15 millones de procesadores por año. Otros, sin embargo, no se rindieron. Federico Faggin deja Intel, y empieza su propia compañía Zilog Inc.

En 1976, Zilog anuncia el Z80. Durante la fabricación de este microprocesador, Faggin toma una decisión giratoria. Sabiendo que ya se han desarrollado muchos programas para 8080, Faggin sabía que muchos se quedarían fieles a ese microprocesador. Así que decide diseñar un nuevo procesador que pueda ser compatible con 8080, o que sea capaz de desarrollar todos los programas que ya se habían escrito para el 8080. Además de estas características, se agregaron muchas otras para que el Z80 fuera un microprocesador más poderoso. Podía direccionar 64 Kb de memoria, tenía 176 instrucciones, un gran número de registros, una opción para refresco de memoria dinámica de la RAM, mayor velocidad de trabajo etc. El Z80 fue un gran éxito y todos cambiaron del 8080 al Z80. Puede decirse que el Z80 fue el microprocesador comercializado más exitoso de ese tiempo. Además de Zilog, también aparecieron otros nuevos fabricantes como Mostek, NEC, SHARP, y SGS. Z80 estaba en el corazón de muchas computadoras como en Spectrum, Partner, TRS703, Z-3 etc. En 1976, Intel propone una versión mejorada del microprocesador de 8 bits, al cual nombró 8085. Sin embargo, el Z80 era tan bueno que Intel perdió la batalla. Aunque más procesadores aparecían en el mercado (6809, 2650, SC/MP etc.), ya todo estaba decidido. Ya no había grandes mejoras departe de los fabricantes para hacer algo nuevo, así que el 6502 y el Z80 junto con el 6800 permanecieron como los representantes principales de los microprocesadores de 8 bits de ese tiempo.

El microprocesador surge como el primer circuito integrado altamente complejo, totalmente estándar y relativamente fácil de utilizar. La combinación complejidad−programabilidad se ha extendido a dos famosos derivados del microprocesador: el Procesador Digital de Señal (DPS) y el **Microcontrolador.**

En 1978, Intel produjo la primera microcomputadora en un solo circuito (single chip microcomputer). La misma contenía en un encapsulado de 40 patas un rudimentario microprocesador orientado a manipulaciones de bits (privilegiando el manejo de variables de este tipo frente a complejas operaciones de bytes), 27 líneas de entrada / salida, 64 Kbytes de memoria de datos y 1 Kbyte de memoria de programa (ROM para la versión 8048, EPROM para el 8748) y un temporizador de 8 bits.

Esta microcomputadora estaba dirigida al mercado de los sistemas de control en los que se suelen manejar entradas y salidas binarias y se optimizaron las operaciones booleanas entre las mismas. Debido a su razonable costo, comenzó a desplazar a sistemas electromecánicos con mayores prestaciones, convirtiéndose rápidamente en un estándar en su tipo. Intel y National presentaron algunas variantes al 8048 como el 8049 con 2 Kbytes de ROM y 128 bytes de RAM y el 8050AH con 4 Kbytes de ROM y 256 bytes de RAM. En 1980, Intel presentó una versión mejorada del 8048 denominada 8051 al cual le incorporó más memoria de programa y de datos, un segundo temporizador con mayor versatilidad, una puerta serie asincrónica e importantes mejoras en su repertorio de instrucciones y arquitectura de registros. Fue tan importante su penetración en el mercado que continúa siendo el núcleo de los microcontroladores actuales. Podrán tener mejoras en la capacidad de memoria RAM o ROM, extender el número de instrucciones, registros, temporizadores y periféricos, pero siempre bajo la estructura básica del **8051**. Además que las herramientas de desarrollo (compiladores, simuladores, etc.) pueden ser de dominio público (gratuitos) o muy populares a moderado costo (Franklin, Keil) y siguen siendo adaptables a las versiones actuales de microcontroladores.

Entonces podemos definir que un microcontrolador es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de una computadora: Microprocesador, Memoria y Unidades de E/S, es decir, se trata de un computador completo en un solo circuito integrado.

Los microcontroladores son parte fundamental en el desarrollo industrial. Están presentes en nuestro trabajo, en nuestra casa y en nuestra vida, en general. Cada vez existen más productos que incorporan un microcontrolador con el fin de aumentar sustancialmente sus prestaciones, reducir su tamaño y costo, mejorar su fiabilidad y disminuir el consumo. Algunos fabricantes de microcontroladores superan el millón de unidades de un modelo determinado producido en una semana. Este dato puede dar una idea de la demanda de estos componentes. Los microcontroladores están siendo empleados en multitud de sistemas presentes en nuestra vida diaria, como pueden ser: juguetes, horno de microondas, frigoríficos, televisores, computadoras, impresoras, módems, el sistema de arranque de nuestro coche, etc. Y otras aplicaciones con las que seguramente no estaremos tan familiarizados como instrumentación electrónica, control de sistemas en una nave espacial, etc. Una aplicación típica podría emplear varios microcontroladores para controlar pequeñas partes del sistema. Estos pequeños controladores podrían comunicarse entre ellos y con un procesador central, probablemente más potente, podría compartir la información y coordinar sus acciones, como, de hecho, ocurre ya habitualmente en cualquier PC.

3.1.1 Arquitectura de un Microcontrolador

Al estar todos los microcontroladores integrados en un chip, su estructura fundamental y sus características básicas son muy parecidas. Todos deben disponer de los bloques esenciales: Procesador, memoria de datos y de programa, líneas de E/S, oscilador de reloj y módulos controladores de periféricos. Figura 3.1. Sin embargo, difieren en su arquitectura interna; además que cada fabricante intenta enfatizar los recursos más idóneos para las aplicaciones a las que se destinan preferentemente.

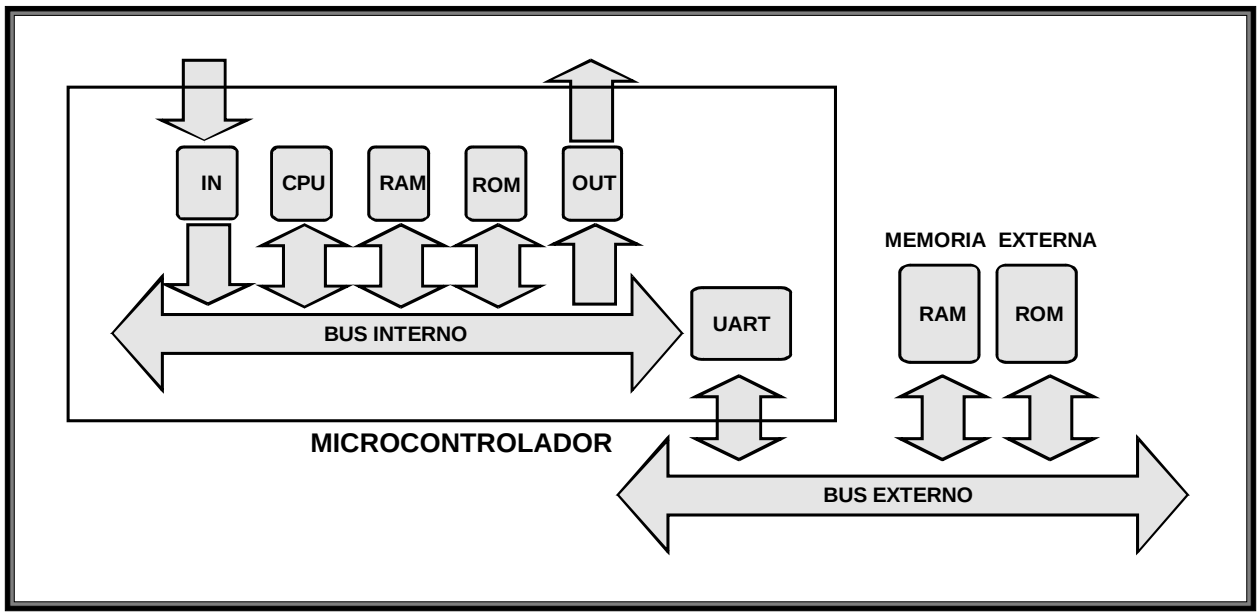


Figura 3.1. Arquitectura de un Microcontrolador

Según la arquitectura interna de la memoria del microcontrolador se puede distinguir entre:

3.1.1.1 Arquitectura de Von Neuman

Esta se caracteriza por disponer de una sola memoria principal donde se almacenan los datos e instrucciones de forma indistinta, a dicha memoria se accede por un sistema de bus único (direcciones, datos y control). Figura 3.2. Esta arquitectura presenta algunos problemas cuando se demanda rapidez.

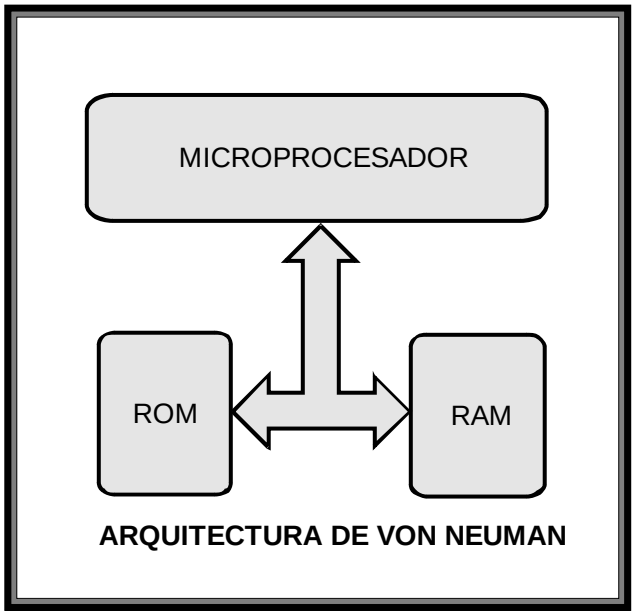


Figura 3.2 Arquitectura Von Newman

3.1.1.2 Arquitectura Harvard

La arquitectura Harvard dispone de dos memorias independientes: una, que contiene sólo instrucciones y otra sólo datos. Ambas, disponen de sus respectivos sistemas de buses de acceso (direcciones, datos y control) y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias. Figura 3.3. Esta estructura no modifica nada desde el punto de vista del usuario y la velocidad de ejecución de los programas es más eficiente.

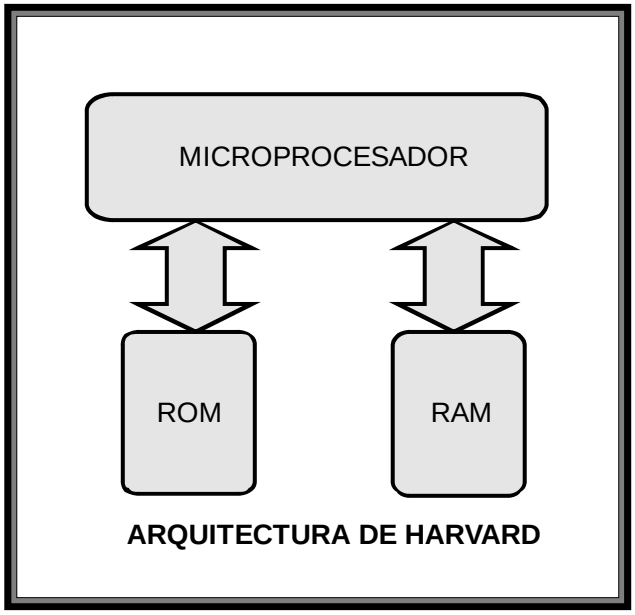


Figura 3.3. Arquitectura Harvard

Según la filosofía de la arquitectura del procesador se puede distinguir entre:

3.1.1.3 Microcontroladores CISC

Un microcontrolador basado en la filosofía CISC (Computadores de Juego de Instrucciones Complejo) dispone de más de 80 instrucciones máquina en su repertorio, algunas de las cuales son muy sofisticadas y potentes, requiriendo muchos ciclos para su ejecución.

Una ventaja de los procesadores CISC es que ofrecen al programador instrucciones complejas que actúa como macros.

3.1.1.4 Microcontroladores RISC

Tanto la industria de los computadores comerciales como los de los microcontroladores están descantándose hacia la filosofía RISC (Computadores de Juego de Instrucciones Reducido). En estos procesadores el repertorio de instrucciones máquina es muy reducido y las instrucciones son simples y, generalmente, se ejecuta en un solo ciclo de maquina.

La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del procesador.

3.1.1.5 Microcontroladores SISC

En los microcontroladores destinados a aplicaciones muy concretas, el juego de instrucciones, además de ser reducido, es específico, o sea, las instrucciones se adaptan a las necesidades de la aplicación prevista. Esta filosofía se ha bautizado con el nombre de SISC (Computadores de Juego de Instrucciones Específico).

Entre los fabricantes de microcontroladores hay dos tendencias para resolver las demandas de los usuarios.

Los microcontroladores con arquitectura cerrada poseen un determinado microprocesador, cierta cantidad de memoria de datos, cierto tipo y capacidad de memoria de instrucciones, un número de E/S y un conjunto de recursos auxiliares muy concreto. El modelo no admite variaciones ni ampliaciones. La aplicación a la que se destina debe encontrar en su estructura todo lo que precisa, y en caso contrario, hay que desecharlo.

Los microcontroladores con arquitectura abierta se caracterizan porque, además de poseer una estructura interna determinada, emplean sus líneas de E/S para sacar al exterior los buses de datos, direcciones y control, con lo que se posibilita la ampliación de la memoria y las E/S con circuitos integrados externos. Esta solución se asemeja a la que emplean los clásicos microprocesadores.

La línea que separa unos de otros es muy delgada, pero el concepto de microcontrolador se acerca posiblemente más a la arquitectura cerrada.

Los recursos especiales más comunes que pueden poseer los microcontroladores son los siguientes:

- Temporizador y/o contador
- Perro guardián o "*Watchdog*".
- Protección ante el fallo de la alimentación
- Estado de reposo o de bajo consumo
- Conversor analógico-digital (CAD)
- Conversor digital-analógico (CDA)
- Comparador analógico
- Modulador de anchura de impulsos o PWM
- Puertas de entrada y salidas digitales
- Puertas de comunicación (USART, USB, SCI, etc.)

Los temporizadores se emplean para controlar periodos de tiempo, actuando como temporizador, o para llevar la cuenta de acontecimientos que suceden en el exterior, actuando como contador.

El perro guardián consiste en un temporizador que cuando se desborda provoca un reset automáticamente en el microcontrolador, para así evitar que el sistema se quede "colgado". Se debe diseñar el programa de tal modo que refresque o inicialice el perro guardián antes de que provoque el reset.

La protección ante el fallo de la alimentación consiste en un circuito que provoca un reset al microcontrolador cuando el voltaje de alimentación sea inferior a un voltaje mínimo. Mientras el voltaje de alimentación sea inferior al mínimo, el dispositivo se mantiene reseteado, comenzando a funcionar normalmente cuando sobrepasa dicho valor.

Son abundantes las situaciones reales en las que el microcontrolador debe esperar, sin hacer nada, a que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. Para ahorrar energía, factor clave en los aparatos portátiles, los microcontrolador disponen de una instrucción especial que les pasa al estado de reposo o de bajo consumo, en el cual los requerimientos de potencia son mínimos.

Los microcontroladores que incorporan sistemas de control, pueden procesar señales analógicas, tan abundantes en las aplicaciones. Suelen disponer de un multiplexor que permite aplicar a la entrada del CAD diversas señales analógicas desde los terminales del circuito integrado.

El conversor digital-analógico transforma los datos digitales obtenidos del procesamiento del microcontrolador, en su correspondiente señal analógica, que saca al exterior por los pines del encapsulado. Existen muchos sistemas que trabajan con señales analógicas.

Algunos modelos de microcontrolador disponen internamente de un amplificador operacional que actúa como comparador analógico entre una señal de referencia fija y otra variable que se aplica por una de los terminales de la cápsula. La salida de comparador proporciona un nivel lógico 1 ó 0 según una señal sea mayor o menor que la otra.

El modulador de anchura de pulsos o PWM, es un circuito que proporciona en su salida impulsos de anchura variable, que se ofrecen al exterior a través de los terminales del encapsulado.

Todos los microcontrolador destinan algunos de sus terminales a soportar líneas de entrada y salida digitales. Por lo general, estas líneas se agrupan de ocho en ocho, formando así lo que se conoce como puertas.

Con el objeto de dotar al microcontrolador de la capacidad de comunicarse con otros dispositivos externos, otros buses de microcontrolador o microprocesadores, buses de sistemas o buses de redes y poder adaptarlos con otros elementos y con otras normas y protocolos, algunos microcontrolador disponen de puertas de comunicación. Destacan las conexiones serie UART y USART, las puertas paralelas o el moderno bus serie USB desarrollado para los PC.

3.2 El Microcontrolador AT89C52

La familia de microcontroladores de Intel conocida como MCS-51 represento el despegue en el uso y aplicación de los microcontroladores, los miembros de esta familia se encuentran en diversas presentaciones, tanto en formas físicas como en características, la selección de uno, o de otro tipo de microcontrolador dependerá principalmente de las necesidades a satisfacer. Diversos fabricantes de semiconductores tienen sus propios derivados, basados en esta familia. En este capitulo se tratará de manera particular del fabricante ATMEL con su versión AT89C52. Figura 3.4.

Es necesario relacionar a este microcontrolador con los de la familia MCS-51, pues es compatible con el estándar de la industria 80C51 y 80C52. El AT89C52 es de bajo consumo, contiene un microprocesador CMOS de 8bits de alto rendimiento, 8Kbytes de memoria flash programable y borrable de solo lectura (PEROM). El dispositivo se fabrica utilizando tecnología de alta densidad en memoria no volátil. La memoria flash contenida en el AT89C52 permite a la memoria de programa ser reprogramado en el sistema o por un programador especial. Esto lleva al AT89C52 a ser un poderoso microcontrolador que proporciona una alta flexibilidad y una solución rentable en aplicaciones de control embebido.

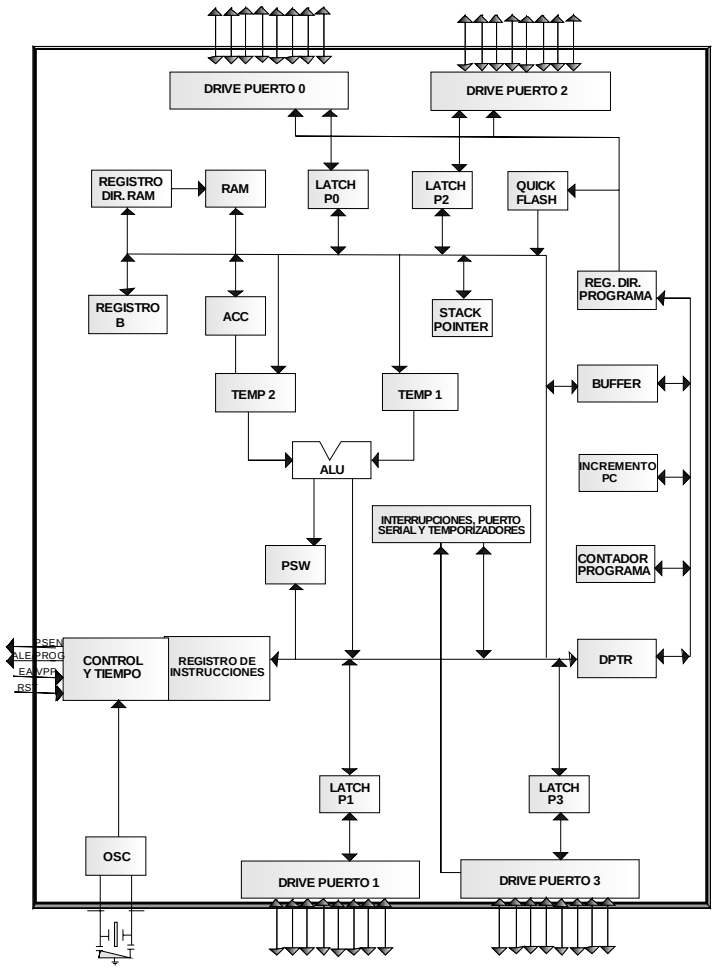


Figura 3.4. Diagrama de bloques interno del AT89C52

3.2.1 Características especificas del AT89C52

El AT89C52 ofrece las siguientes características relevantes:

- Compatibilidad con productos MCS-51
- Un CPU de 8 bits,
- 8Kbytes de memoria FLASH reprogramable
- 256 bytes de memoria RAM interna
- Resistencia a 1000 ciclos de escritura/borrado
- 32 líneas de E/S (4 puertos de entrada y salida paralelo completamente programables en forma individual como salidas o entradas),
- 3 Timer/Contadores de 16 bits
- 3 Niveles de seguridad para la protección de código
- 8 Fuentes de interrupción con dos niveles de prioridad
- 1 Puerto serial full-duplex
- Modo bajo consumo de potencia en reposo y apagado
- Un oscilador interno y circuiteria para el reloj.

3.2.2 Descripción de las Líneas del AT89C52

La presentación de este dispositivo puede encontrarse en tres tipos de encapsulado: DIP, PLCC Y PQFT/TQFP. Figura 3.5. La presentación DIP es utilizada para montaje de Pin In Hole, en tanto las presentaciones PLCC y PQFP/TQFP son especiales para la tecnología de montaje superficial (SMT).

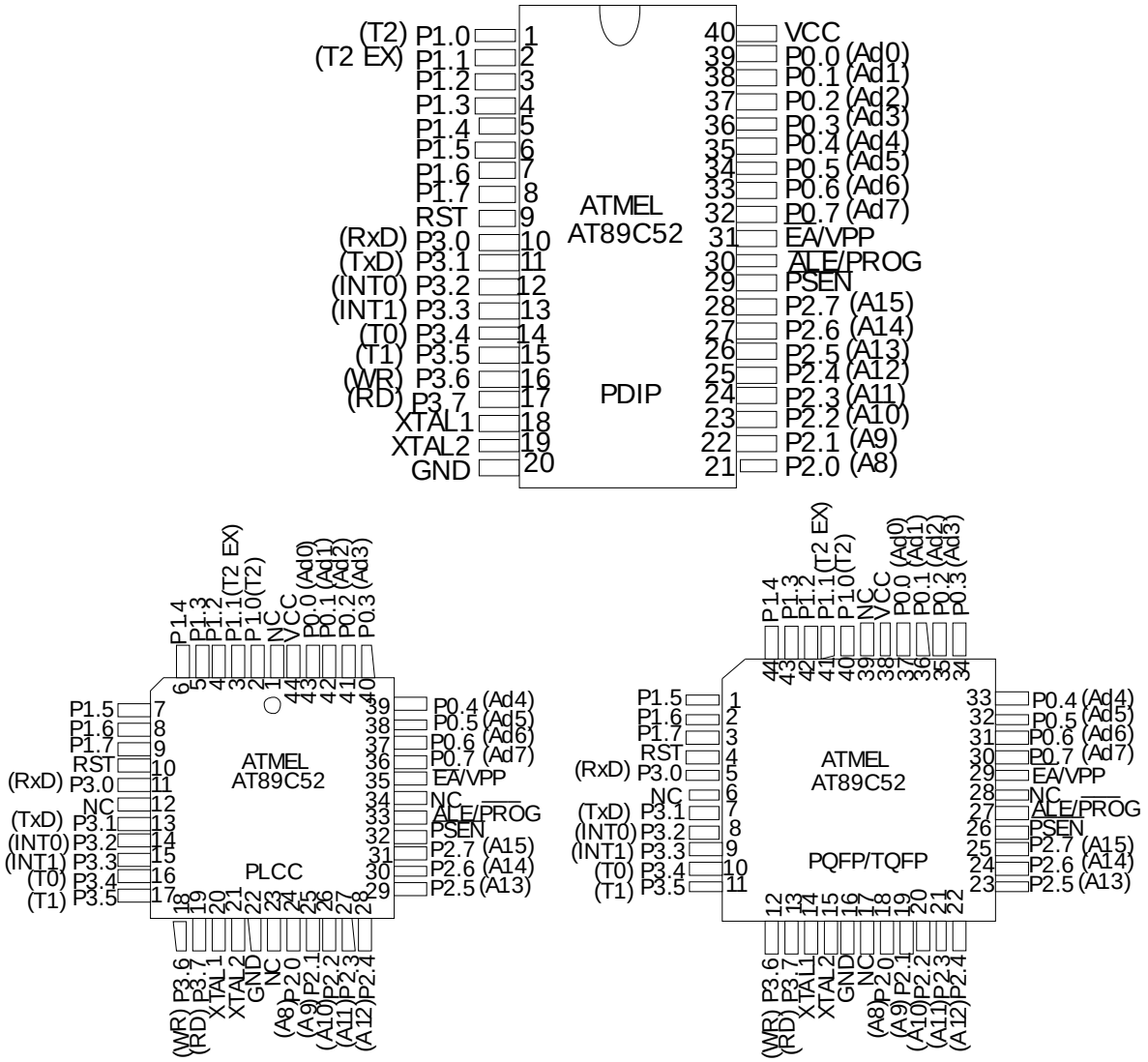


Figura 3.5. Encapsulados del AT89C52. DIP, PLCC y PQFP/TQFP

En el AT89C52, cuenta con 4 puertos distribuidos de la siguiente manera:

PUERTO0: Ubicado en los pines 32 al 39 (p0.7-p0.0, respectivamente) es un puerto bidireccional con salidas en colector abierto. Cuando tiene 1's escritos las salidas están flotadas y pueden servir como entradas en alta impedancia. Puede activar hasta 8 compuertas TTL y es el que recibe los datos en la programación, necesita de resistencias pull-ups en la programación y en la verificación.

PUERTO1: Ubicado en los pines 1 al 8 (P1.0 – P1.7, respectivamente) es un puerto bidireccional con pull-ups internas, puede activar hasta 4 compuertas TTL, cuando se escriben 1's en el puerto, este puede ser usado como entradas en alta impedancia, además p1.0 (T2) y p1.1 (T2 EXT) pueden ser configurados como entradas del timer2/contador2, también recibe la dirección baja durante la programación y la verificación.

PUERTO2: Ubicado en los pines 21 al 28 (P2.0 – P2.7, respectivamente) es un puerto bidireccional con pull-ups internas. Puede activar hasta 4 compuertas TTL, cuando se escriben 1´s este puede ser utilizado como entradas de alta impedancia. Como entradas las líneas que son externamente colocadas en la posición baja, proporcionaran una corriente hacia el exterior. Este puerto es usado para direccionar la memoria externa. Este puerto, emite la parte alta de la dirección durante la búsqueda de datos en la memoria de programa externa y durante el acceso a la memoria de datos externa que usa direcciones de 8 bits; además recibe las direcciones altas durante la programación y la verificación.

PUERTO3: Ubicado en los pines 10 al 17 (P3.0 – P3.7), es un puerto quasi-bidireccional con fijadores de nivel interno (pull-up). Cuando se escriben 1's sobre el puerto las lineas se pueden emplear como entradas de alta impedancia. Como entradas las lineas que son externamente colocadas en la nivel bajo proporcionan una corriente hacia el exterior. El puerto 3 es usado para producir señales de control de dispositivos externos como: RxD (P3.0), TxD (P3.1), INT0 (P3.2), INT1 (P3.3) y T0 (P3.4)

VCC: Utiliza el pin 40, es una línea para el voltaje de alimentación positiva, +5v
VSS: Utiliza el pin 20, es la línea para la conexión a tierra, 0v
RST: Utiliza el pin 9, es una línea de entrada de nivel alto, durante dos ciclos de maquina, mientras el oscilador esta funcionando, detiene el programa en ejecución e inicializa los registros internos del microcontrolador, así como el programa.
XTAL1: Utiliza el pin 19, es una línea de entrada del cristal para el circuito oscilador (generador de reloj interno) que amplifica e invierte la entrada.
XTAL2: Utiliza el pin 18, es una línea de salida para la señal del amplificador oscilador inversor.

Como es un microcontrolador de arquitectura abierta, cuenta con lineas de control para los dispositivos externos.

EA: Ubicado en el pin 31, External Access Enable. EA = 0 habilita el direccionamiento para la memoria de programa externo (0000h – FFFFh). Si EA = 1 se habilita el direccionamiento a

memoria de programa interno (PEROM) a menos que se exceda la dirección 1FFFh. Este pin recibe el pulso de +12vpp durante la programación.
PSEN: Ubicado en el pin 29, Program Store Enable. Cuando el AT89C52 esta ejecutando un código de memoria de programa externo PSEN es activada dos veces cada ciclo de maquina, excepto cuando se acceda a la memoria de datos externa que omite las dos activaciones del PSEN, este no es activado cuando se usa la memoria de programa interno.

ALE: Ubicado en el pin 30, Addres Latch Enable. Genera un pulso positivo de salida, permite fijar la dirección baja durante el acceso a memoria externa. En operación normal ALE es emitido en un rango constante de 1/6 de la frecuencia del oscilador y puede ser usada para cronometrar. Durante la programación este pin es usado para el pulso de programación.

WR: Ubicado en el pin 16, este pin se comparte con el puerto 3 (p3.6). Genera la señal de habilitación para la escritura en memoria de datos externa.

RD: Ubicado en el pin 17, este pin se comparte con el puerto 3 (p3.7). Genera la señal de habilitación para la lectura en memoria de datos externa.

RxD: Ubicado en el pin 10, compartido con el puerto 3 (p3.0) es empleado como entrada para el puerto serie

TxD: Ubicado en el pin 11. Compartido con el puerto 3 (p3.1) es empleado como salida para el puerto serie.

INT0: Ubicado en el pin 12. Compartido con el puerto 3 (p3.2) es empleado como entrada para la interrupción externa 0.

INT1: Ubicado en el pin 13. Compartido con el puerto 3 (p3.3) es empleado como entrada para la interrupción externa 1.

T0: Ubicado en el pin 14. Compartido con el puerto 3 (p3.4) es empleado como entrada para el timer0/contador0.

T1: Ubicado en el pin 15. Compartido con el puerto 3 (p3.5) es empleado como entrada para el timer1/contador1.

3.2.3 Descripción de Conexiones Básicas de operación

Para el funcionamiento básico del AT89C52 debe tener como mínimo las siguientes conexiones:

La conexión de la fuente de alimentación, este soporta un voltaje típico de +5v en Vcc, conectado al pin 40 y una referencia de 0v o GND localizado en el pin 20.

Además, se debe configurar el circuito generador de reloj, en nuestro caso se emplea un cristal de cuarzo de 24Mhz, con dos capacitores de 22pF para estabilizar la frecuencia del cristal, son conectados a los pines 18 y 19 (XTAL1 y XTAL2).

Debe contener una conexión para el reset, este se produce al aplicar una tensión de +5v en el pin 9 (RST) este se puede obtener mediante la conexión de un capacitor de 10uF y una resistencia 8.2KOhms como se ve en la Figura 3.6.

Como el sistema mínimo de funcionamiento no requiere de memoria de programa externo para funcionar, el pin 31 perteneciente a EA, deberá estar conectado a un nivel alto como Vcc para asegurar que se ubique en la memoria de programa interno.

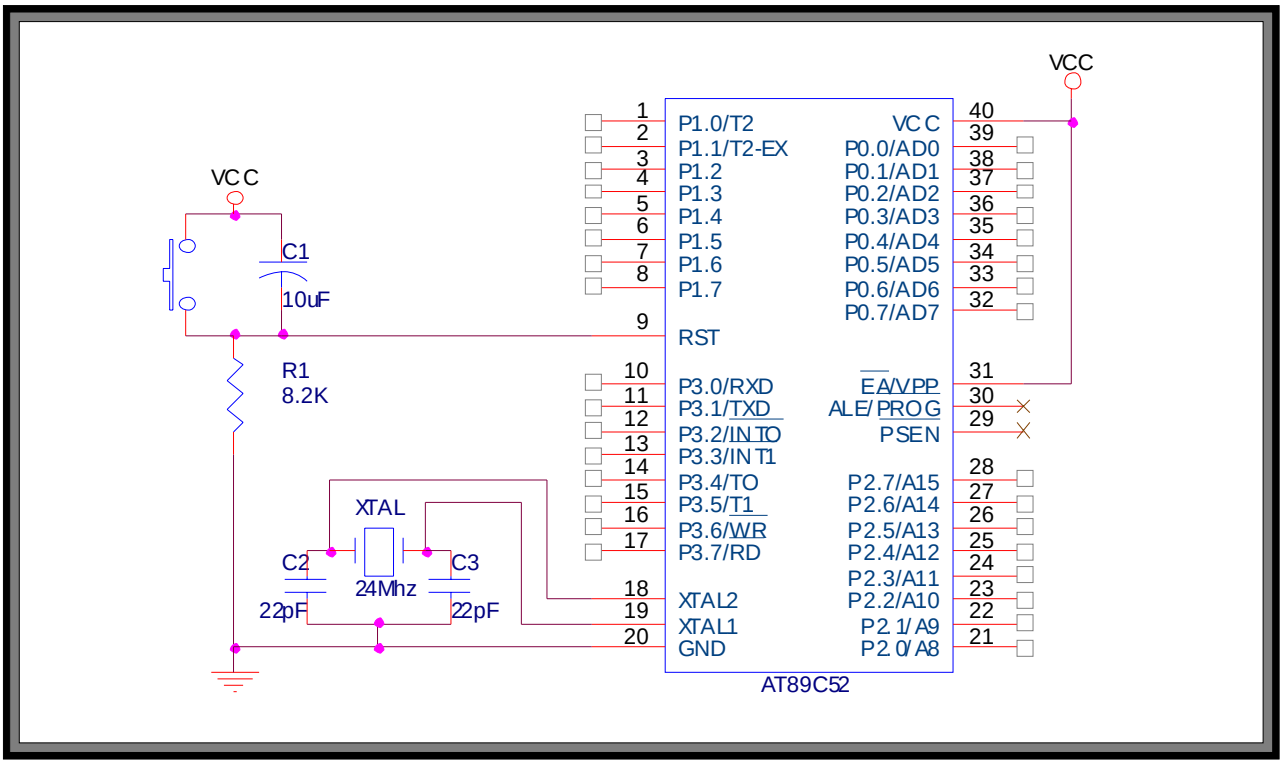


Figura 3.6. Conexión básica del AT89C52, sin memoria externa

3.2.4 Descripción de Espacios de Memoria

El AT89C52 tiene dividido su sistema de memoria en tres secciones fundamentales como: **MEMORIA DE PROGRAMA, MEMORIA DE DATOS y MEMORIA DE RAM INTERNA**.

3.2.4.1 Memoria de Programa

Es empleada para almacenar el código del programa, es decir, son todas las instrucciones que van a ser ejecutadas por el microcontrolador. En el caso del AT89C52 posee una memoria de programa interna de 8Kbytes, que va de la localidad 0000h hasta 1FFFh; reservado para el código de programa, se trabaja con este espacio cuando EA=1. Cuando se requiera trabajar con un espacio de memoria mayor al que tiene internamente, se puede seleccionar la memoria de programa externa mediante la activación de EA=0. El máximo espacio de memoria de programa externo que se puede acceder es de 64 Kbytes, localidades 0000h – FFFFh. Figura 3.7. Con los puertos 0 y 2, se puede direccionar todo este espacio de localidades de memoria externa de programa.

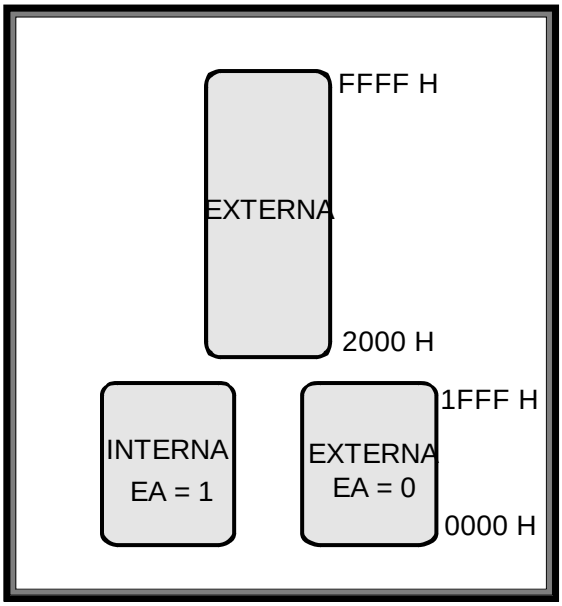


Figura 3.7. Mapa de Memoria de Programa

3.2.4.2 Memoria de Datos

Este segundo espacio de memoria es accesado mediante la activación de las señales RD (p3.6) y WR (p3.7) durante la lectura o escritura de datos respectivamente. En este espacio el microcontrolador únicamente puede tomar o alterar los valores que se encuentran y no puede ejecutar ninguna instrucción que se encuentre ahí almacenada. El AT89C52 puede direccionar 64 Kbytes de memoria de datos a igual que puede direccionar la memoria de programa externo. Figura 3.8

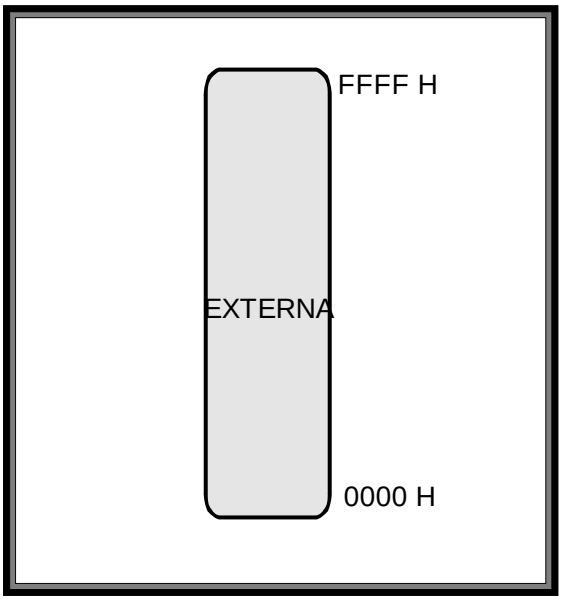


Figura 3.8. Mapa de Memoria de Datos Externa

3.2.4.3 Memoria Ram Interna

En el tercer espacio de memoria, el AT89C52 implementa 256 bytes de memoria RAM interna como memoria de datos para el usuario, mas 128 bytes de RAM interna destinada a la configuración y trabajo del microcontrolador en la que se encuentran los Registros de funciones Especiales (SFR). Los 128 bytes bajos de la memoria de usuario son destinados para manipular datos que son importantes, estos es, que solo se ocupan para la toma de decisiones lógicas o de señalización y simplifican los programas. Y los 128 bytes altos también son destinados al usuario como memoria de datos, pero ocupan direcciones paralelas al espacio destinado a los Registros de Funciones Especiales, es decir, tiene direcciones iguales pero están físicamente separados. El contenido de los Registros de Funciones Especiales esta formado por las localidades de memoria asignadas a: Puertos, Registros de control, Acumuladores, Registros de Interrupciones etc. Por lo tanto el AT89C52 tiene un espacio total de memoria RAM interna de 384 bytes, de los cuales 256 bytes son para el usuario y 128 bytes para el microcontrolador. Figura 3.9

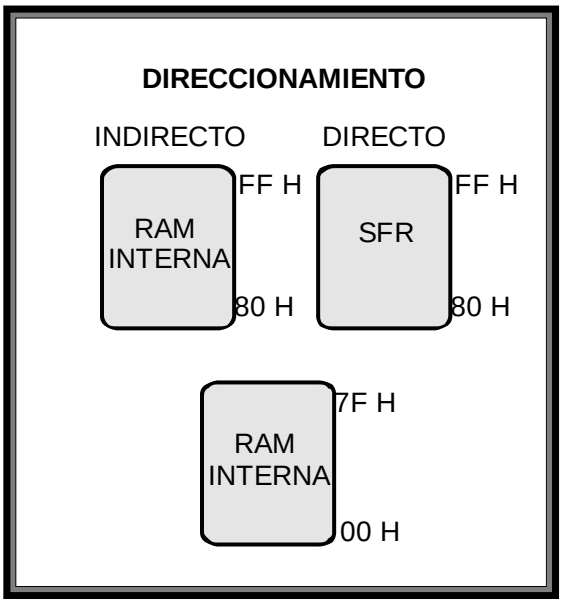


Figura 3.9. Mapa de Memoria de Datos Interna

3.2.5 Registro de Funciones Especiales

Los Registros de Funciones Especiales (SFR), son localidades de memoria RAM interna. Estas se asignan para configurar los recursos del microcontrolador; asi como para manipular la información en aplicaciones específicas del microcontrolador. Figura 3.10.

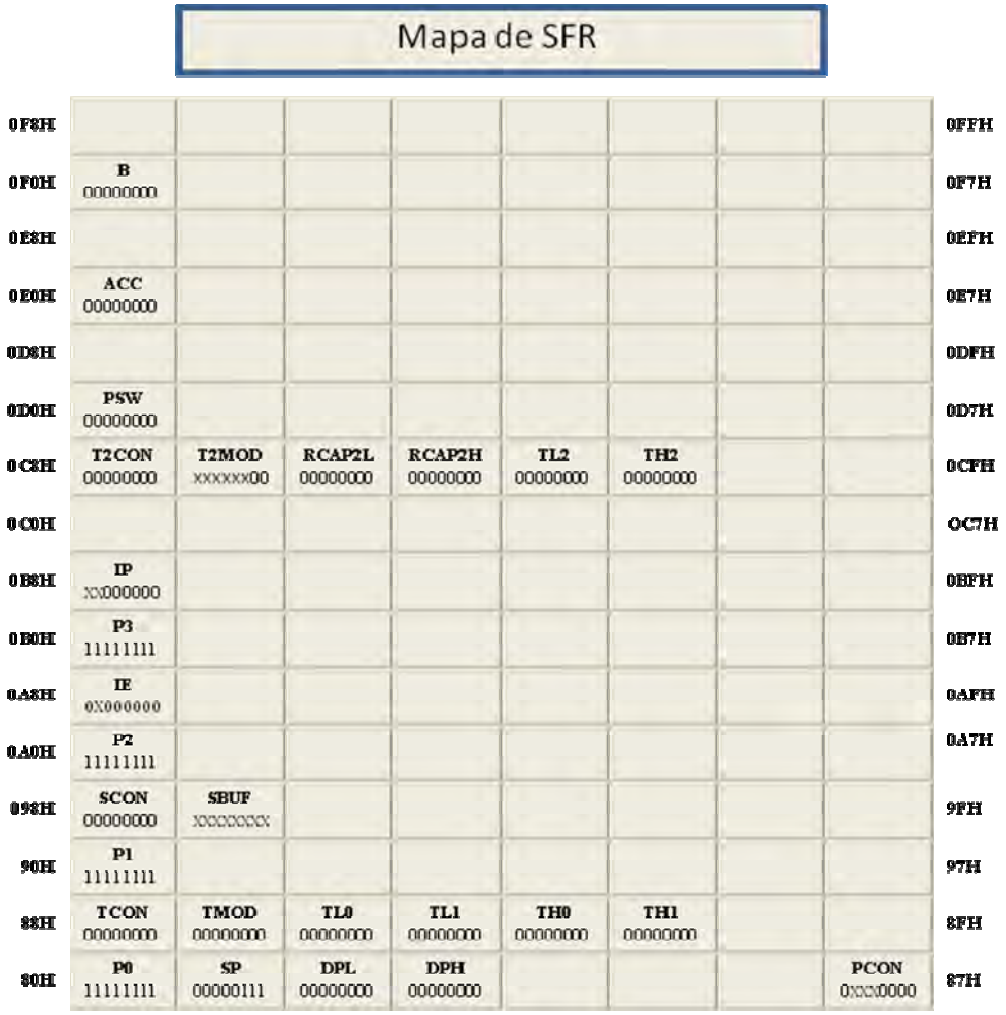


Figura 3.10. Mapa de Registro de Funciones Especiales

A continuación se describen algunos de los más empleados:

ACC: El Acumulador, es un registro de propósito general y por su frecuencia de intervención es el mas importante; en torno a el se realiza la mayoría de operaciones aritméticas y lógicas, es usado como contenedor de los resultados.

B: Este registro es empleado como un acumulador auxiliar, es usado en operaciones aritméticas como multiplicación y división. Puede utilizarse como un registro común.

PSW: Contiene información sobre el estado del microcontrolador en cada momento, cada ciclo de maquina.

SP: Stack Pointer, este registro se ocupa para guardar la dirección de memoria RAM interna, en la que guarda el dato al utilizar la instrucción PUSH, o caso contrario, en donde el dato contenido en esta dirección, será obtenido al ejecutar la instrucción POP. Después de un reset, este se ubica en la dirección 07h, por lo que la primera dirección por default es 08h. Cabe mencionar que puede ser movido a cualquier dirección de la memoria RAM interna; además es usado en todas las llamadas a subrutinas o interrupción para recuperar los datos almacenados con las instrucciones RET y RETI.

DPTR: Data Pointer, este registro se compone de un byte alto (DPH) y un byte bajo (DPL). Esta diseñado para retener direcciones de 16 bits. Se puede usar como registro de 16 bit, o como dos registros independientes de 8 bits.

P0, P1, P2, P3: Son registros latches de lo puertos 0, 1, 2, 3 respectivamente. Escribir un 1, causa la correspondiente salida de estado alto en el pin del puerto, al igual ocurre si es cero, tendrá un estado bajo a la salida del pin. Y viceversa si se lee un pin del puerto, o puerto completo.

SBUF: Destinado al buffer serial, son dos registros separados. Un buffer es usado para transmitir y otro para recibir. Cuando un dato es movido al SBUF, este se va al buffer para transmitir, así se inicia la transmisión serial. Cuando un dato es movido desde el buffer, viene del buffer de recepción.

3.2.6 Modos de Direccionamiento

La forma en que se codifica los códigos de operación depende del modo de direccionamiento, la forma en que se proveen los operandos al microcontrolador. En esta familia de microcontroladores los códigos de operación son de 8 bits, lo que nos proporcionaría 256 códigos de operación posible, pero no todos están implementados.

Como en el estándar de la familia MCS-51 existen seis modos de direccionamiento, estos son: **IMPLICITO, INMEDIATO, DIRECTO, INDIRECTO, POR REGISTRO E INDEXADO**.

3.2.6.1 Direccionamiento Implícito

Este modo es el más sencillo, consta de un solo código de operación, no requiere operandos adicionales ya que el mismo código de operación implica los datos que modifica como:

```
Clr A          ; Limpia el contenido del acumulador

Div AB         ; Realiza la operación A/B, coloca el cociente en A y el resto en B
```

3.2.6.2 Direccionamiento inmediato

Aquí el byte que sigue inmediatamente al código de operación es tratado como una constante, es decir, no se utiliza como una dirección o referencia a un registro, como:

```
Mov A,#3eh     ;Carga el acumulador con el valor 3eh y no con el contenido del
               registro 3eh. El símbolo "#" denota que se trata de una constante
```

3.2.6.3 Direccionamiento Directo

En este modo, el código de operación va acompañado de un operando que contiene la dirección de un byte de la RAM interna o de la zona de SFR, como:

```
Mov 3fh, A     ; Mueve el contenido de Acc a la localidad 3fh
```

Cabe mencionar, que del mismo modo pertenecen algunos códigos que operan sobre bits en lugar de bytes como.

```
Setb 21h.7     ; Pone 1 en el bit 7 de la localidad de memoria RAM interna 21h

Setb Acc.2     ; Pone 1 en el tercer bit de la localidad del Acumulador
```

3.2.6.4 Direccionamiento Indirecto

Se podría clasificar el direccionamiento indirecto en dos submodos, direccionamiento indirecto a través de registro de 8 bits y registro de 16 bits. Básicamente ya sea mediante registros de 8 o 16 bits, la idea es hacer referencia a un dato apuntando a la posición de memoria que lo contiene con el contenido de un tercer registro (8 o 16 bits), como:

Haciendo uso del direccionamiento inmediato.

```
Mov r0, #2fh   ; Se carga r0 con el valor 2fh
```

Ahora usando el direccionamiento indirecto, movemos el contenido de la posición apuntada por r0, entonces:

```
Mov A, @r0   ; Mueve el contenido de la dirección dada por el registro r0 al A
```

El símbolo "@" nos indica que el dato es "apuntado" por el registro r0 y no lo "contenido" en el.

3.2.6.5 Direccionamiento por Registro

Este modo es particular del MCS-51, ya que se aplica en código de operación que utilizan el numero de registro r0 a r7 del banco actualmente seleccionado. Los códigos de operación que pertenecen a este modo llevan en su byte el número de registro seleccionado, como:

```
Djnz r5, loop  ; Decrementa r5 y salta a "loop" si r5 no llega a ser cero en el decremento.
Mov r5, A      ; El contenido del acumulador es transferido a r5 (05h)
```

3.2.6.6 Direccionamiento Indexado

Este modo solo es posible sobre la memoria de programa y solo permite la lectura. Es bastante útil para armar tablas estáticas en la memoria, esto es, no solo ocuparemos la memoria para guardar código de operación; sino para almacenar datos que podamos emplear para diversas soluciones. Por ejemplo, guardar leyendas que podamos ocupar para visualizarlas en un display, tablas de direcciones de salto a subrutinas en base a un dato que actúa como base, etc.…

Para el manejo de estas tablas son necesarios dos elementos: una dirección base de la tabla y tamaño definido de la tabla.

El primero es un punto de referencia, ya que la tabla puede ubicarse dentro de 64Kbytes disponibles para la memoria de programa, entonces es necesario el empleo de un registro de 16 bits como DPTR (puntero de datos) o PC (contador de programa).

El segundo elemento para manejar la tabla es un índice para recorrerla desde la dirección base hasta su tamaño máximo. Aquí también hay un limite de 256 bytes, de esta manera se emplea un registro de 8 bits como puede ser el acc.

```
Movc A,@A+DPTR      ; El contenido de la dirección de memoria de programa, es indicada
```

por la suma del valor actual del Acumulador y del Data Pointer, es transferido al Acumulador.

3.2.7 Set de Instrucciones

La ejecución de una instrucción comienza en el primer ciclo de maquina, el cual tiene doce estados, cuando el código de operación es almacenado en el registro de instrucción. Como norma general, la ejecución de una instrucción requiere de uno o más ciclos de maquina en función de: código de operación, el número de bytes

El set o conjunto de instrucciones del AT89C52, se agrupan según su operación. Tabla 3.1

OPERACIONES ARITMETICAS	5
OPERACIONES LOGICAS	10
TRANSFERENCIA DE DATOS	7
SALTO O CONTROL DE PROGRAMA	13
MANIPULACION A BIT O ALGEBRA BOOLEANA	11

Tabla 3.1. Instrucciones del AT89C52

Las instrucciones aritméticas, lógicas, de transferencia y salto son comunes a la mayoría de microprocesadores, estas están orientadas al byte como dato. En los Microcontroladores en cambio, la unidad de manejo de datos es el bit. Encontraremos entonces instrucciones que nos permiten interactuar directamente con un pin determinado de un puerto de E/S. También por lo general, los registros internos de la CPU del microcontrolador y algunas posiciones de memoria de datos interna son accesibles a nivel de bit, esto es, disponemos de un código de operación para cambiar el estado de un bit en particular de una posición de memoria de datos.

Esta característica particular de los Microcontroladores es llamada “direccionamiento a nivel de bit” (Bit Addressable); pero esta “filosofía del bit” no solo implica el cambio de estado de bits pertenecientes a puertos o posiciones de memoria, sino que también existen códigos de operación para procesar a nivel lógico y aritmético determinados conjuntos de bits y lo que es muy importante, tomar decisiones en base a los resultados.

CAPITULO 4

4 COMUNICACIÓN SERIAL

En este capitulo trataremos los aspectos fundamentales de la comunicación serial y su importancia en la transmisión de información ya que una de las operaciones más comunes que se presentan en cualquier sistema digital es la transmisión de información de un lugar a otro. Para fines informativos, en este capitulo se contemplaran algunos de los protocolos mas empleados en la industria, enfocándonos en los protocolos RS-232 y 2D-RS-485 utilizados en el desarrollo del presente trabajo.

4.1 Introducción

La información que se transmite, está en forma binaria y se representa por voltaje a las salidas del circuito de envío, conectadas en las entradas del circuito de recepción.

Los elementos básicos en la comunicación son:

FUENTE: Este dispositivo genera los datos a transmitir, un ejemplo puede ser un teléfono o PC.

TRANSMISOR: Normalmente los datos generados por la fuente no se transmiten directamente tal y como son generados. Al contrario, el transmisor transforma y codifica la información, generando señales electromagnéticas susceptibles de ser transmitidas a través de algún sistema de transmisión. Por ejemplo, un MODEM convierte las cadenas de bits generadas por un computador general y las transforma en señales analógicas que pueden ser transmitidas a través de la red de telefonía.

SISTEMA DE TRANSMISIÓN: Puede ser desde una sencilla línea de transmisión (cable conductor) hasta una compleja red que conecte la fuente con el destino.

RECEPTOR: El receptor capta la señal proveniente del sistema de transmisión y lo transforma de tal manera que pueda ser interpretada por el dispositivo de destino. Por ejemplo, un modem captara la señal analógica de la red o línea de transmisión y la convertirá en una cadena de bits.

DESTINO: Toma los datos ya interpretados por el receptor.

La transmisión se refiere al número de unidades de información elementales que se pueden enviar simultáneamente a través de los canales de comunicación, ya sean bits o bytes. La transmisión de datos en los medios informáticos se realiza de dos maneras: transmisión en paralelo o transmisión en serie.

La transmisión en serie y paralelo son protocolos muy comunes en la comunicación entre dispositivos. Se incluyen de manera estándar en prácticamente cualquier computadora. Sin embargo para nuestro propósito solo abarcaremos la comunicación serial.

El canal de comunicación es el recurso físico que hay que establecer entre transmisor y receptor para efectuar la comunicación, también se denomina vínculo o enlace; además existen tres modos en el que se puede establecer la comunicación y depende de la dirección del intercambio de información, como son: simplex, dúplex o semi-duplex y full dúplex.

Modo Simplex es una comunicación en la que los datos fluyen en una sola dirección, desde el transmisor hacia el receptor. Este tipo de comunicación es útil si los datos no necesitan fluir en ambas direcciones, por ejemplo: desde el equipo hacia la impresora o desde el ratón hacia el equipo.

Modo Duplex ó Semi-dúplex es una comunicación en la que los datos fluyen en una u otra dirección, pero no las dos al mismo tiempo. Con este tipo de comunicación, cada extremo de la conexión transmite uno después del otro. Este tipo de comunicación hace posible tener una comunicación bidireccional utilizando toda la capacidad de la línea.

Modo full dúplex es una comunicación en la que los datos fluyen simultáneamente en ambas direcciones. Así, cada extremo de la conexión puede transmitir y recibir al mismo tiempo; esto significa que el ancho de banda se divide en dos para cada dirección de la transmisión de datos si es que se está utilizando el mismo medio de transmisión para ambas direcciones de la transmisión.

4.2 Conceptos de Transmisión Serie

La transmisión serie, consiste en enviar los datos un bit por vez a través del canal de transmisión y la secuencia es por orden de peso creciente y generalmente el último bit es de paridad. Figura 4.1. Sin embargo ya que los procesadores manejan la información en paralelo, el transmisor necesita transformar los datos paralelos entrantes en datos seriales y el receptor necesita hacer lo contrario. Estas operaciones son normalmente realizadas por un controlador de comunicaciones llamado UART (Universal Asynchronous Receiver Transmiter).

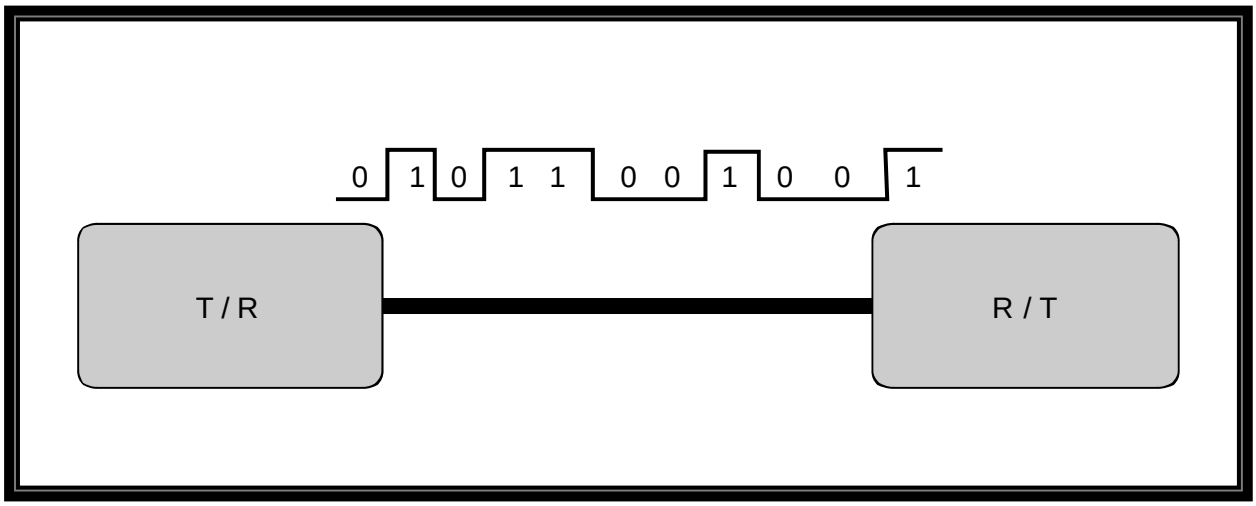


Figura 4.1. Transmisión en serie.

La comunicación serie reduce la complejidad del cableado con respecto a una comunicación paralela y costo del sistema, pero obteniendo a cambio una menor eficacia: es necesario un intervalo de tiempo ocho veces mayor para transmitir ocho bits individuales que para transmitirlo simultáneamente como ocurre en una transmisión en paralelo.

El aspecto fundamental de la transmisión serie es el sincronismo, entendiéndose como tal al procedimiento mediante el cual transmisor y receptor reconocen el inicio y el final del paquete de información.

El sincronismo puede tenerse a nivel de bit, de byte o de bloque, donde en cada caso se identifica el inicio y finalización de los mismos.

La transmisión serie se divide en dos tipos: Transmisión síncrona y asíncrona.

4.2.1 Transmisión Síncrona

Se produce cuando la trama de datos transmitidos se envían a un ritmo constante, exige la transmisión de los datos y además una señal de reloj para sincronizar emisor y receptor (base de tiempos común); cada bloque de información se configura comenzando por un conjunto de bits de sincronismo denominados (SYN) y termina con otro conjunto de bits de final de bloque denominados (ETB). En este caso, los bits de sincronismo tienen la función de sincronizar los relojes existentes tanto del emisor como del receptor, de tal forma que estos controlan la duración de cada bit y carácter. Figura 4.2

En este tipo de transmisión es necesario que tanto el transmisor como el receptor utilicen la misma frecuencia de reloj. En este caso la transmisión se efectúa en bloques, debiéndose definir dos grupos de bits denominados delimitadores, mediante los cuales se indica el inicio y el fin de cada bloque.

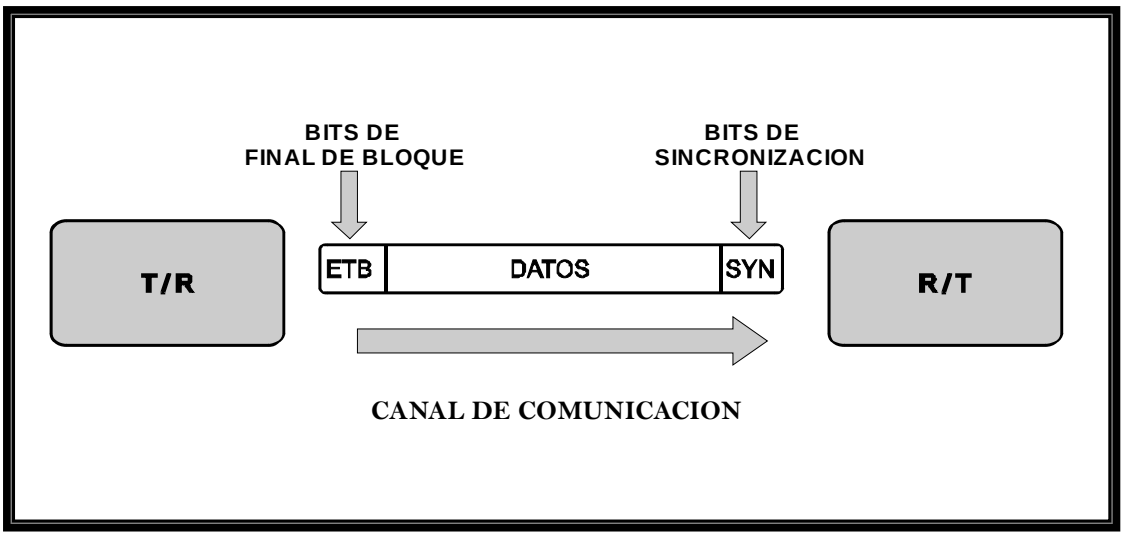


Figura 4.2. Transmisión Síncrona.

Esta transmisión es más efectiva por que el flujo de información ocurre en forma uniforme, con lo cual es posible lograr velocidades de transmisión altas.

4.2.2 Transmisión Asíncrona

La transmisión asíncrona ocurre cuando el proceso de sincronización entre emisor y receptor se realiza en cada palabra del código transmitida. Esto se produce debido a que se lleva a cabo a través de unos bits especiales que definen el entorno de cada código.

Es también conocida como inicio/parada. Requiere de un bit que identifique el inicio del dato, a esta se le llama bit de inicio. También se requiere de otro bit denominado bit de parada que indica la finalización del dato. Figura 4.3

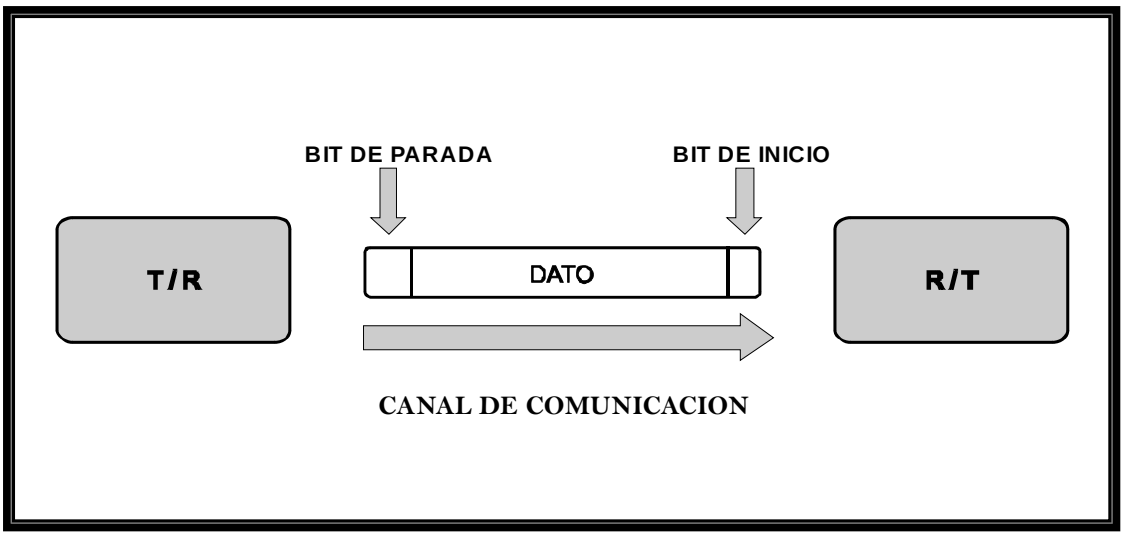


Figura 4.3. Transmisión Asíncrona

Las características más importantes son: la velocidad de transmisión, los bits de datos, los bits de parada, y la paridad. Para que dos puertos se puedan comunicar, es necesario que las características sean iguales.

Velocidad de transmisión *(baud rate)*: También conocida como baudaje, es el número de unidades de señal por segundo. Un baudio puede contener varios bits. Aunque a veces se confunden los baudios con los bits por segundo, estos son conceptos distintos.

Bits de datos: Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits. Las cantidades más comunes de bits por paquete son 5, 7 y 8 bits. El número de bits que se envía depende del tipo de información que se transfiere. Un paquete se refiere a una transferencia de byte, incluyendo los bits de inicio/parada, bits de datos, y paridad. Debido a que el número actual

de bits depende en el protocolo que se seleccione, el término paquete se usa para referirse a todos los casos.

Bit de parada: Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1 o 2 bits. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj, es posible que los dos dispositivos no estén sincronizados. Por lo tanto, los bits de parada no sólo indican el fin de la transmisión sino además dan un margen de tolerancia para esa diferencia de los relojes. Mientras más bits de parada se usen, mayor será la tolerancia a la sincronía de los relojes, sin embargo la transmisión será más lenta.

Paridad: Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marca y espacio. La opción de no usar paridad alguna también está disponible. Para paridad par e impar, el puerto serial fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico. Por ejemplo, si la información a transmitir es 011 y la paridad es par, el bit de paridad sería 0 para mantener el número de bits en estado alto lógico como par. Si la paridad seleccionada fuera impar, entonces el bit de paridad sería 1, para tener 3 bits en estado alto lógico. La paridad marca y espacio en realidad no verifican el estado de los bits de datos; simplemente fija el bit de paridad en estado lógico alto para la marca, y en estado lógico bajo para el espacio. Esto permite al dispositivo receptor conocer de antemano el estado de un bit, lo que serviría para determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos, o si los relojes de los dispositivos no están sincronizados.

4.3 Estándares de Nivel Físico.

Los estándares de nivel físico, son normas que designan los requerimientos óptimos y características técnicas de la comunicación. Entre las más usuales dentro de la transmisión serie encontramos las normas: RS-232, RS-422 y RS-485. Tabla 4.1

Características RS232, RS422, RS423 y RS485				
Diferencial	no	no	si	si
Numero máx. de Transmisores Numero máx. de Receptores	1 1	1 10	1 10	32 32
Modos de Operación	half duplex full duplex	half duplex	half duplex	half duplex
Topología de red	Punto a punto	multipunto	multipunto	multipunto
Distancia Max. (acc. standard)	15 m	1200 m	1200 m	1200 m
Velocidad Max. a 12 m Velocidad Max. a 1200 m	20 kbs 1 kbs	100 kbs 1 kbs	10 Mbs 100 kbs	35 Mbs 100 kbs
Max slew rate	30 V/µs	ajustable	n/a	n/a
Resistencia de Entrada del Receptor	3..7 kΩ	≧ 4 kΩ	≧ 4 kΩ	≧ 12 kΩ
Impedancia de Carga del Transmisor	3..7 kΩ	≧ 450 Ω	100 Ω	54 Ω
Sensibilidad de Entrada del Receptor	±3 V	±200 mV	±200 mV	±200 mV
Rango de Entrada del Receptor	±15 V	±12 V	±10 V	–7..12 V
Voltaje de Salida Max del Transmisor	±25 V	±6 V	±6 V	–7..12 V
Voltaje de Salida Min. del Transmisor (Con Carga)	±5 V	±3.6 V	±2.0 V	±1.5 V

Tabla 4.1. Tabla comparativa de estándares

4.3.1 Norma RS-232

Desarrollado por EIA (Asociación de industrias Electrónicas), conjuntamente con los Laboratorios Bell y los fabricantes de equipos de comunicación, formularon la norma EIA RS-232. Con el propósito inicial de conexión entre un PC o DTE (Equipo Terminal de Datos) y un Modem o DCE (Equipo de Comunicación de Datos), empleando un intercambio de datos binario en serie.

Actualmente el RS-232 es uno de los medios principales mediante el cual se pueden conectar equipos auxiliares a los ordenadores personales.

Esta norma designa:

Las características de la señal eléctrica.

Esta señal se transmite a través de una línea asimétrica unidireccional, es una línea que se encuentra referida a tierra. Los niveles de voltaje se encuentran comprendidos entre -15v y 15v. Al contrario de las convenciones lógicas de uso corriente aquí un voltaje positivo representa un 0, mientras que un voltaje negativo representa un 1.

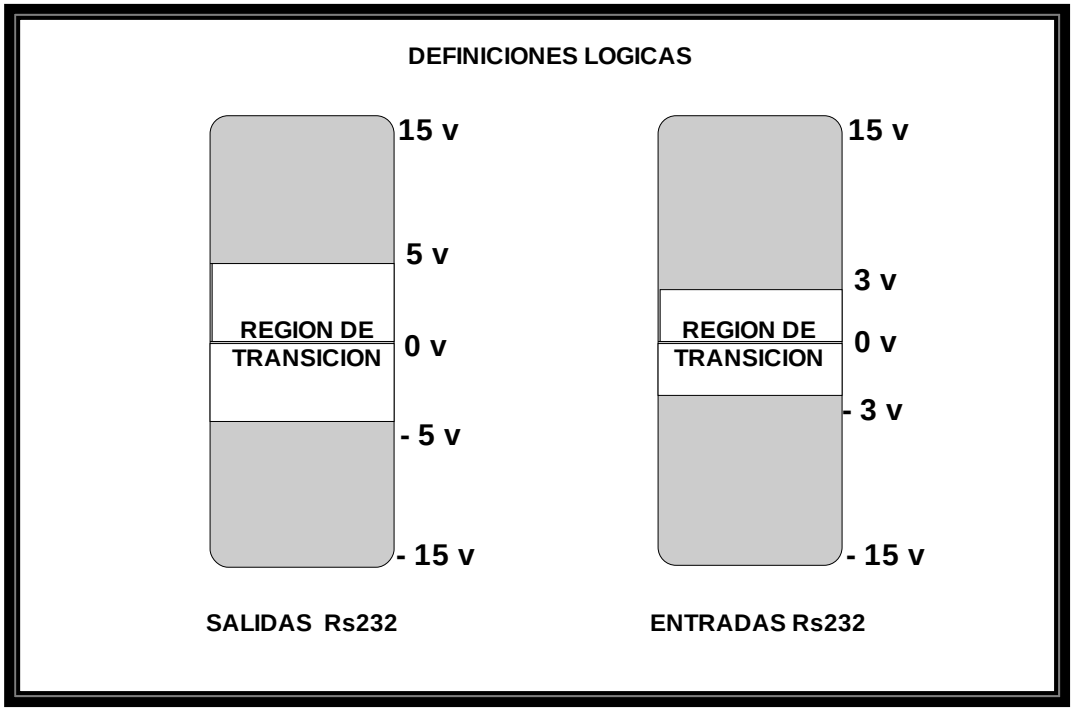


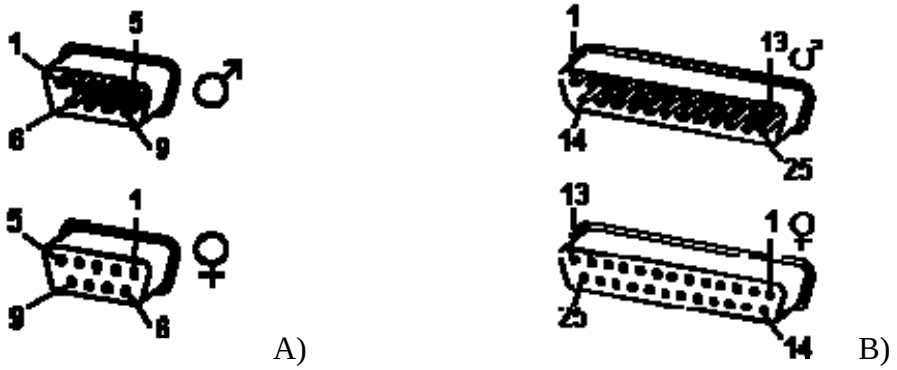
Figura 4.4. Niveles de voltajes admitidos

La única diferencia entre la definición de salida y de entrada es el ancho de la región de transición, de -3v a +3v en la entrada y de -5v a +5v de salida.

Si aumentamos la velocidad de transmisión, las señales de datos se vuelven susceptibles a pérdidas de voltaje causadas por la capacidad, resistencia e inductancia del cable. Estas pérdidas son conocidas como efectos de alta frecuencia, y aumentan con la longitud del cable. El ancho de la zona de transición (-3V a +3V en la entrada) determina el margen de ruidos, que limita directamente la velocidad máxima a la que se pueden transmitir datos sin degradación. Entre dos equipos RS-232 esta velocidad es de 19200 bits por segundo, para longitudes de cable inferiores a 15 metros, pero disminuyendo la velocidad pueden utilizarse longitudes mayores de cable.

Características mecánicas de la conexión.

Establece el tipo de conector que tendrá el TRANSMISOR y el RECEPTOR. También especifica la asignación de número de pines, tipo y medidas del conector. Los más utilizados son los de 9 pines (DB-9) y los de 25 (DB-25). Figura 4.5



A) B)

Figura 4.5. Conectores tipo DB, A) Db9 y B) Db25

Descripción funcional de los circuitos de intercambio. Tabla 4.2. Define y da nombre a las señales que se utilizaran.

CONECTOR DB 25 / PIN	CONECTOR DB 9 / PIN	NOMBRE	FUNCION
2	3	TxD	Transmisión de Datos (SALIDA)
3	2	RxD	Recepción de Datos (ENTRADA)
4	7	RTS	Petición de Envío
5	8	CTS	Dispuesto para Enviar (ENTRADA)
6	6	DSR	Dispositivo de Datos Listo (ENTRADA)
7	5	COMUN	Común (REFERENCIA)
8	1	DCD	Detección de Portadora de Datos (ENTRADA)
20	4	DTR	Terminal de Datos Lista (SALIDA)
22	9	RI	Indicador de Llamada (ENTRADA)

Tabla 4.2. Descripción de pines RS-232

- 1 pin para enviar datos (TXD).
- 1 pin para recibir datos (RXD).
- 1 pin común a todos los circuitos (GND).
- 4 pines para señales de acoplamiento para poder enviar datos (CTS, DSR, DCD, RI).
- 2 pines para señales de acoplamiento para poder recibir datos (RTS, DTR).

Algunas de las señales (DCD, RI) provienen de características necesarias para poder detectar el estado de un Módem, pero no suelen ser necesarias para aplicaciones normales en PC.

Las configuraciones de interfaces seleccionadas en este apartado se denominan NULL MODEM. Con estas interfaces se puede conectar directamente dos DTE o PC's usando un cable serie RS-232. Para elaborar el cable las líneas de transmisión y recepción están cruzadas. Figura 4.6 y Figura 4.7

Existe más de una forma de realizar una conexión Null Modem ya que no hay ningún estándar que defina esta conexión.

Para conectar dos ordenadores personales con señales de acoplamiento con conector Db9 y Db25. Figura 4.6.

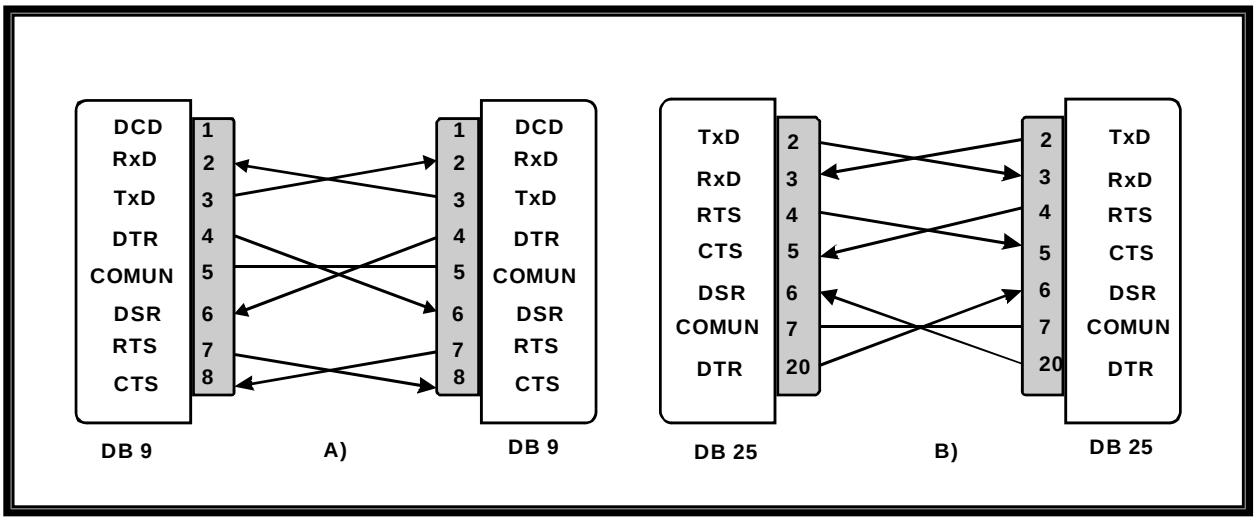


Figura 4.6. Interfaz con señales de acoplamiento RS-232, A) Db9 y B) Db25

En el caso de no desear utilizar estas señales de acoplamiento, puede optarse por proporcionarlas por un medio físico, pues algunos programas de comunicación pueden requerir su presencia. Figura 4.7

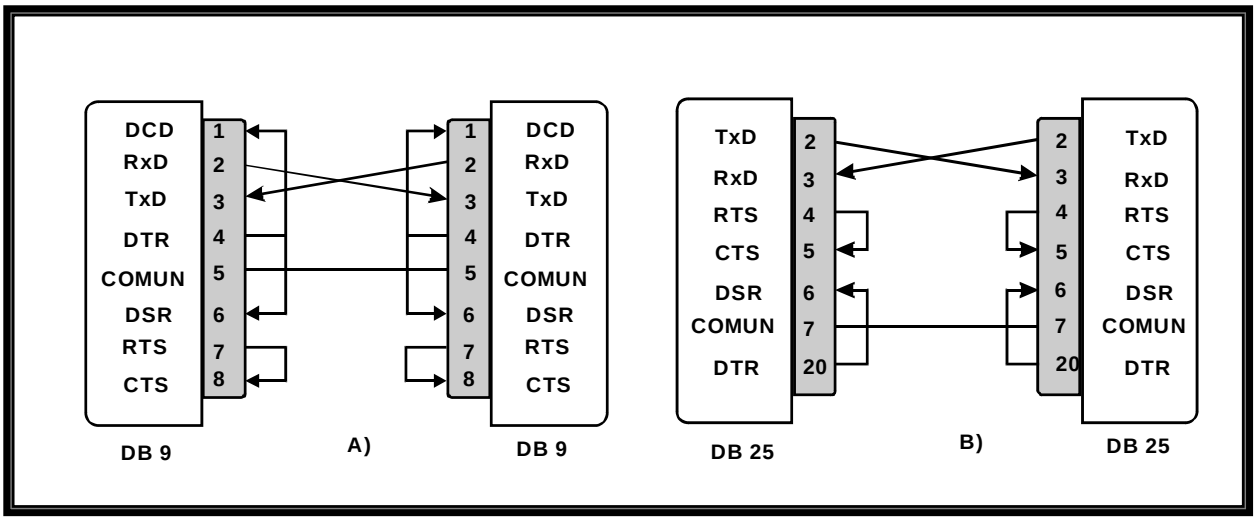


Figura 4.7. Interfaz sin señales de acoplamiento RS-232, A) Db9 y B) Db25

4.3.2 Normas RS-422 y RS-423

La EIA (Asociación de Industrias Electrónicas) desarrolló y formuló en 1975 otro estándar de transmisión de datos más robusto, el RS-422-A.

El modo de aumentar la eficacia de transmisión se consigue utilizando circuitos balanceados, es decir, no utiliza un único hilo para transmitir cambiando la polaridad con referencia a un circuito común, sino que utiliza dos hilos para cada señal. Las condiciones de 0 y 1 lógico son determinadas por cambios en la polaridad de los dos hilos, por referencia del uno con el otro.

Este cambio del sistema permite un incremento notable de las prestaciones; utilizando cable trenzado apantallado es posible alcanzar distancias de 1200m y velocidades de transmisión cercanas a 1 megabit por segundo (1Mbps).

El conjunto completo de señales correspondientes al estándar RS-422-A tiene la distribución mostrada. Tabla 4.3

CONECTOR DB 25 PIN	NOMBRE	FUNCION
1	TIERRA	Tierra
2	TxD (+)	Transmisión de Datos (SALIDA +)
3	RxD (+)	Recepción de Datos (ENTRADA +)
4	RTS (+)	Petición de Envío (SALIDA +)
5	CTS (+)	Dispuesto para Enviar (ENTRADA +)
6	DSR (+)	Dispositivo de Datos Listo (ENTRADA +)
7	TIERRA	Tierra
8	DCD (+)	Detección de Portadora de Datos (ENTRADA +)
9	DTR (-)	Terminal de Datos Listo (SALIDA -)
10	RI (-)	Indicador de llamada (ENTRADA -)
14	TxD (-)	Transmisión de Datos (SALIDA -)
15	RxD (-)	Recepción de Datos (ENTRADA -)
16	RTS (-)	Petición de Envío (SALIDA -)
17	CTS (-)	Dispuesto para Enviar (ENTRADA)
18	DSR (-)	Dispositivo de Datos Listo (ENTRADA -)
19	DCD (-)	Detección de Portadora de Datos (ENTRADA -)
20	DTR (+)	Terminal de Datos Listo (SALIDA +)
22	RI (+)	Indicador de Llamada (ENTRADA +)

Tabla 4.3. Descripción de pines RS-422

Si queremos efectuar una conexión punto a punto RS-422 a RS-422 completa deberemos utilizar 14 cables diferentes. Figura 4.8.

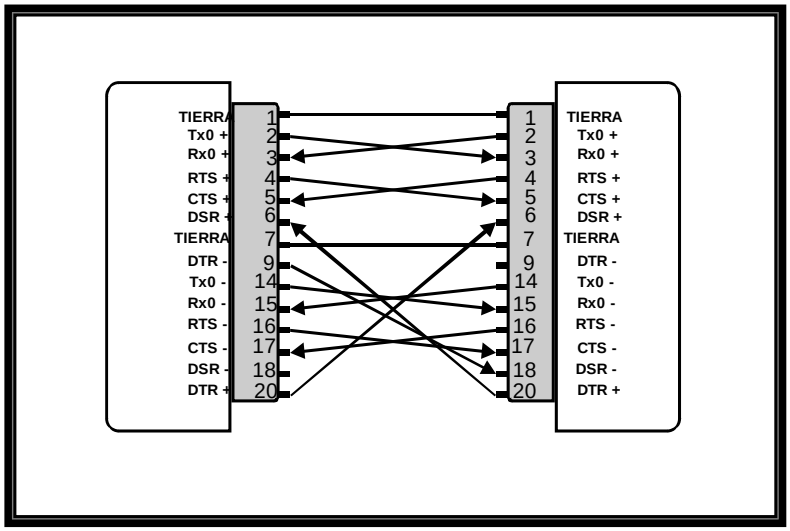


Figura 4.8. Interfaz con señales de acoplamiento RS-422

Al igual que en el caso del RS-232-C, si no queremos utilizar las señales de acoplamiento, podemos optar por proporcionarlas por un medio físico, conectando las salidas de acoplamiento con las entradas del mismo dispositivo. Figura 4.9

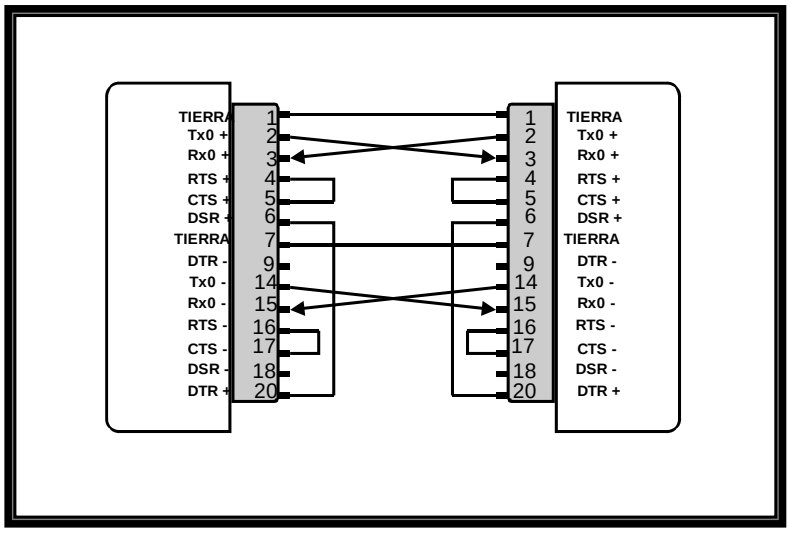


Figura 4.9. Interfaz sin señales de acoplamiento RS-422

4.3.3 Norma RS-485

El modelo EIA-RS-485 permite características no previstas en el estándar RS-422. Mantiene ventajas del RS-422, al permitir velocidades de transmisión cercanas a 1 Mbps, así como longitudes de la línea de hasta 1200 metros. Además permite el alargamiento de la red en otros 1200 metros al insertar un repetidor RS-485 en la línea.

También tiene otra característica muy importante en ambientes industriales, puede soportar hasta 32 nodos (equipos emisores/receptores) conectados por cada segmento de red. Estos distintivos lo hacen muy adecuado para el trabajo que fue diseñado, aplicaciones industriales.

Dentro del estándar RS-485 existen diferentes variantes, una de las cuales es la conocida como 4D-RS-485. En este caso se mantienen por separado los pares de cables de recepción y transmisión. Las únicas señales que son necesarias para transmitir se muestran en la siguiente Tabla 4.4.

Nombre	Función
TXD (+)	TRANSMISIÓN DE DATOS (SALIDA +)
TXD (-)	TRANSMISIÓN DE DATOS (SALIDA -)
RXD (+)	RECEPCIÓN DE DATOS (ENTRADA +)
RXD (-)	RECEPCIÓN DE DATOS (ENTRADA -)
TIERRA	TIERRA

Tabla 4.4. Señales utilizadas en la conexión 4D-RS-485

De este modo el sistema de conexión queda simplificado respecto al RS-422, además de permitir la presencia de más de dos equipos, como se describe en Figura 4.10.

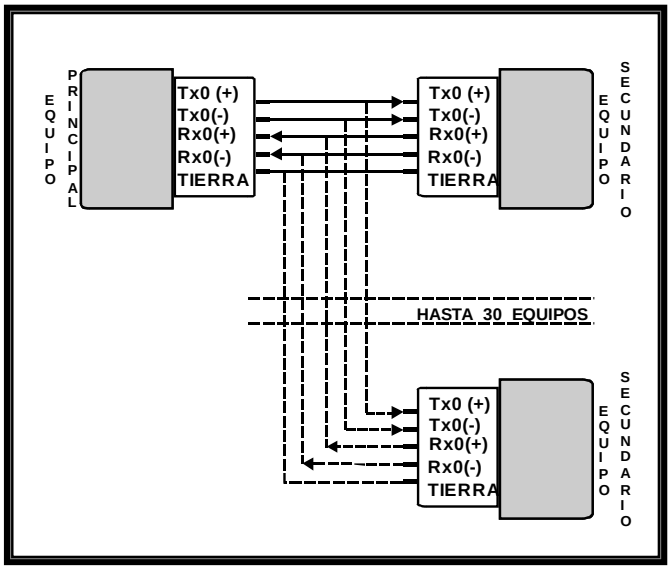


Figura 4.10. Conexión entre equipos 4D-RS-485.

El siguiente paso en simplificación es el estándar 2D-RS-485, el más comúnmente conocido por estándar RS-485. En este caso se elimina uno de los pares transmisión/recepción. Se utiliza una sola línea de transmisión balanceada bidireccional. Las características físicas de la línea se mantienen (longitud y velocidades de transmisión admisibles). La diferencia con el anterior es que los dispositivos deben conmutar entre modo receptor y modo transmisor, para evitar que varios dispositivos emitan simultáneamente.

Las señales necesarias en este caso son las mostradas en Tabla 4.5.

Nombre	Función
TXD / RXD (+)	TRANSMISIÓN/RECEPCION DE DATOS (SALIDA +)
TXD / RXD(-)	TRANSMISIÓN/RECEPCION DE DATOS (SALIDA -)
TIERRA	TIERRA

Tabla 4.5. Señales utilizadas en la conexión 2D-RS-485

El esquema de interconexión entre equipos debería establecerse como muestra la siguiente figura 4.11.

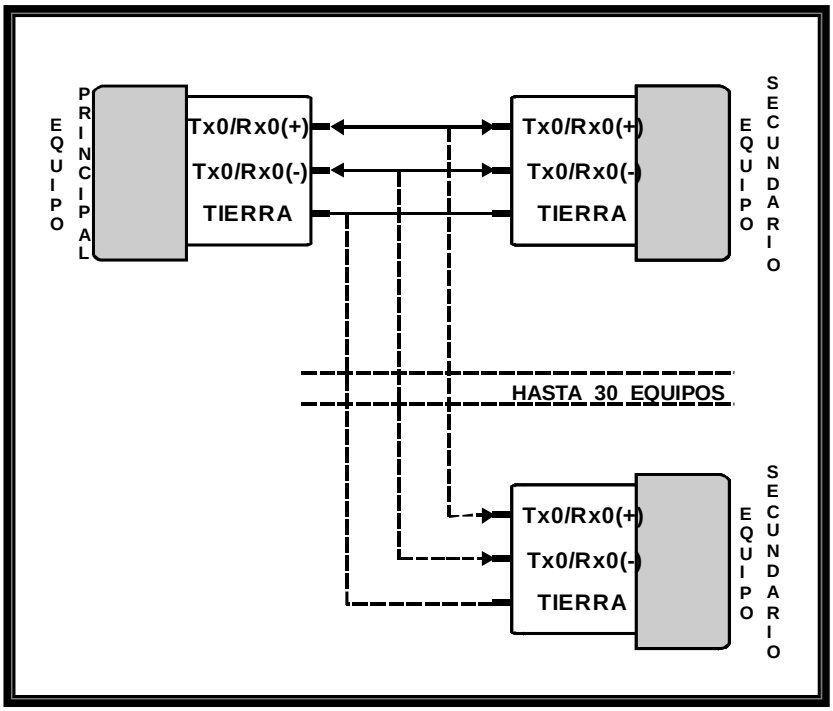


Figura 4.11. Conexión entre equipos 2D-RS-485.

CAPITULO 5

5. Herramientas de Desarrollo

Las herramientas de desarrollo en las que nos apoyaremos para conseguir nuestro objetivo, se describirán a continuación como lo es el uso de programas para la edición, depuración y simulación de código ensamblador para la programación del AT89C52, así como un programador para realizar la descarga del código maquina al microcontrolador. Este capitulo también abarca la descripción y uso de visual Basic para poder realizar la aplicación en PC.

Las herramientas de desarrollo para microprocesadores y microcontroladores pueden ser clasificadas dentro de dos categorías: software y hardware.

Para el desarrollo de software de cualquier proyecto basado en microprocesador o microcontrolador se necesita de la ayuda de un ensamblador, esto no es más que una herramienta del software diseñada para simplificar la tarea de escritura de programas para computadoras, se encarga de traducir el código simbólico en un código ejecutable. Entonces este código puede almacenarse en un microcontrolador o en memoria externa en el caso de un microprocesador para que pueda ser ejecutado.

Cuando se ensamblan un conjunto de programas lo que se hace es traducirlos en un idioma mediante el cual se pueda instruir al microprocesador o microcontrolador para que este ejecute eficazmente las funciones que se desea realizar; por tanto al momento de escribir un programa se debe estar totalmente familiarizado con las arquitecturas y el lenguaje que puede interpretar.

Un programa escrito en lenguaje ensamblador consta de tres partes:

- Instrucciones de Máquina
- Directivas del ensamblador
- Control del ensamblador

Una instrucción de máquina es un código que puede ser ejecutado por la máquina.

Las directivas del ensamblador se usan para definir la estructura del programa, los símbolos y generar código no ejecutable como datos, mensajes, etc.

El control del ensamblador controla los modos del ensamblado y dirige el flujo del ensamblado.

Muchos programas son demasiado largos o complejos como para escribirlos como una sola unidad. El programar se vuelve mucho más simple cuando el código es dividido en pequeñas unidades funcionales o módulos, además este tipo de programas son más fáciles de codificar, depurar y realizar cambios o actualizaciones.

Las herramientas de software incluyen editor de programa, ensamblador, compilador, simulador, depurador y software de integración.

Las herramientas de hardware incluyen tarjeta de evaluación, analizadores lógicos, emulador de microprocesadores y microcontroladores, osciloscopio, puntas lógicas, etc.

5.1 Descripción del KEIL μ Visión 3

KEIL es un software, Figura 5.1, que tiene herramientas de desarrollo y se usan para compilar y codificar un lenguaje de programación. KEIL soporta dos lenguajes: Assembler y C. Los archivos fuente creados pasan por el compilador C51 o el ensamblador A51 dependiendo del tipo de lenguaje que se está usando.

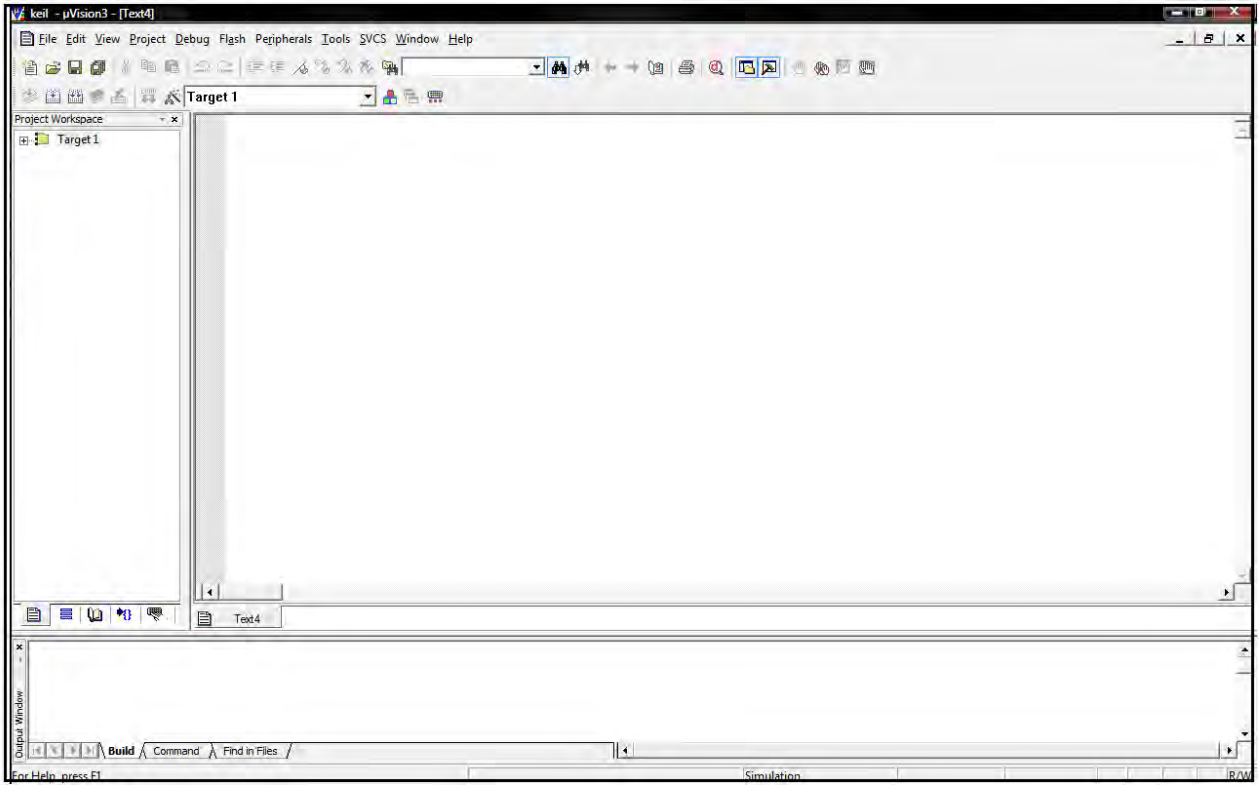


Figura 5.1. Entorno Grafico de Desarrollo, KEIL

El compilador C51 es una aplicación que maneja el lenguaje ANSI - C, además incluye varias herramientas que son específicas para ayudar al desarrollo de software para la arquitectura 8051, mientras que el ensamblador A51 apoya el juego completo de instrucciones para programación en bajo nivel de este tipo de arquitecturas.

KEIL se encarga de ensamblar los códigos fuente, enlaza y localiza los objetos, módulos y librerías, crea archivos HEX y pone a punto el programa desarrollado para cada tarjeta. Este software permite escoger el tipo de procesador que se va usar antes de empezar con el desarrollo del programa; es decir, se adapta a cada uno de los diseños que se desea crear.

KEIL proporciona numerosas herramientas que ofrecen ciertas características y ventajas para el desarrollo de aplicaciones en sistemas embebidos. Es una plataforma de desarrollo de software bajo Windows basado en la combinación de un editor robusto y un administrador de proyecto, lo que brinda muchas facilidades, entre ellas se tiene:

- Editor de código fuente completo.
- Base de datos del dispositivo para configurar el ambiente de desarrollo de herramientas.
- Un administrador de proyecto que le permite crear y mantener los proyectos que se van desarrollando.
- Facilidad de desarrollo integrada para ensamblar, compilar, depurar y unir las aplicaciones para sistemas embebidos.
- Diálogos para todos los ambientes de herramientas para desarrollo.
- Verdadera corrección de errores integrado con un simulador de CPU periférico de gran velocidad.
- Vínculos a los manuales, herramientas de desarrollo, hojas de datos de los dispositivos a ser utilizados y guías de desarrollo.

El compilador C51 del Keil es un compilador ANSI 51 que se escribió específicamente para generar un código rápido y compacto para la familia de microcontroladores 8051. El compilador C51 genera un código objeto que se asemeja a la eficacia y velocidad de la programación en Assembler.

Al utilizar un lenguaje de alto nivel como es el lenguaje C se tiene muchas ventajas sobre el lenguaje Assembler ya que no se requiere conocer todo el juego de instrucciones que se necesita para usar el ensamblador, tan solo se requiere un conocimiento básico de la estructura de memoria del microcontrolador 8051, adicionalmente existen muchas funciones que están integradas en las librerías del compilador. Detalles tales como la asignación de registros y el direccionamiento de los diferentes tipos de memoria y de datos son manejados por el compilador.

Los programas mantienen una estructura formal (estructura impuesta por el lenguaje de programación C) pero se puede dividir en funciones separadas. Esto contribuye a la reutilización del código fuente así como a una mejor estructura de aplicación global gracias a la habilidad de combinar estas funciones separadas con rutinas específicas mejorando la legibilidad del programa, sin mencionar el ahorro en cuanto a tiempo al momento de desarrollar el programa.

El A51 es un macro ensamblador de la familia de microcontroladores 8051. Se encarga de traducir el lenguaje mnemónico simbólico a un lenguaje ejecutable de máquina. El ensamblador A51 maneja el juego de instrucciones existentes para un microcontrolador 8051, usando la velocidad máxima, tamaño de código pequeño y el hardware exacto, además del control que es esencial. Las facilidades que brinda este ensamblador permiten ahorrar tiempo en el desarrollo y mantenimiento del programa porque las rutinas que son comunes necesitan ser desarrolladas una sola vez. Figura 5.2.

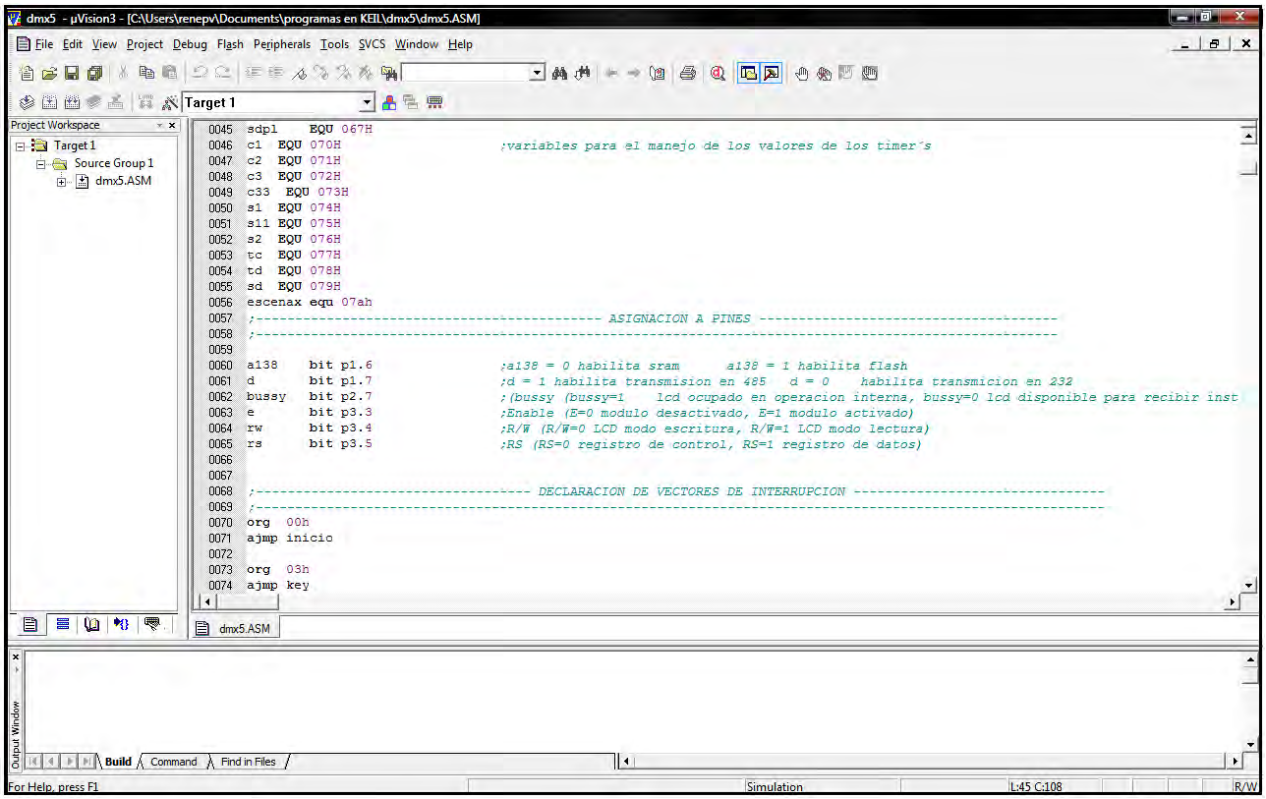


Figura 5.2. Edición de código en Ensamblador A51

5.1.1 Creación de un Proyecto Usando KEIL.

Para crear un proyecto usando KEIL debe seguir los siguientes pasos:

- En el menú **Project** seleccione la opción **New Project**
- Inmediatamente se asigna un nombre al proyecto.
- En la pantalla que aparece a continuación, se escoge el tipo de microcontrolador con el que se va a trabajar.
- En el menú **File** se escoge **New** para obtener un nuevo archivo de texto para empezar con la programación.
- En el mismo menú está la opción guardar con el que se asigna un nombre al archivo de texto y salvar los cambios realizados.

5.1.2 Ensamblado del Proyecto

Se pueden implementar cuantos archivos de texto sean necesarios, cada uno de estos corresponde a un módulo del programa (que podría corresponder a lenguaje C o Assembler), para ensamblar estos módulos como uno solo se deben realizar los siguientes pasos:

- En la ventana de la izquierda cuando se ha seleccionado un nuevo proyecto aparece una carpeta llamada **Target 1**, debajo está una subcarpeta llamada **Source Group 1**, haciendo clic con el botón secundario del mouse se escoge la opción **Add Files to group "Source Group 1"** y se agregan los archivos de texto o módulos que se han creado al desarrollar el programa. Una vez que están todos los archivos necesarios se procede a construir el **Target**, en el menú **Project** se escoge la opción **Build Target** y en la opción **Options for Target** se puede configurar la alternativa para crear el archivo HEX que se grabará en el microcontrolador. Este archivo se creará únicamente si no existen errores de sintaxis, en el caso de que esto suceda en la parte inferior de la pantalla llamada **Output Window** se muestran los mensajes de error existentes. Figura 5.3.

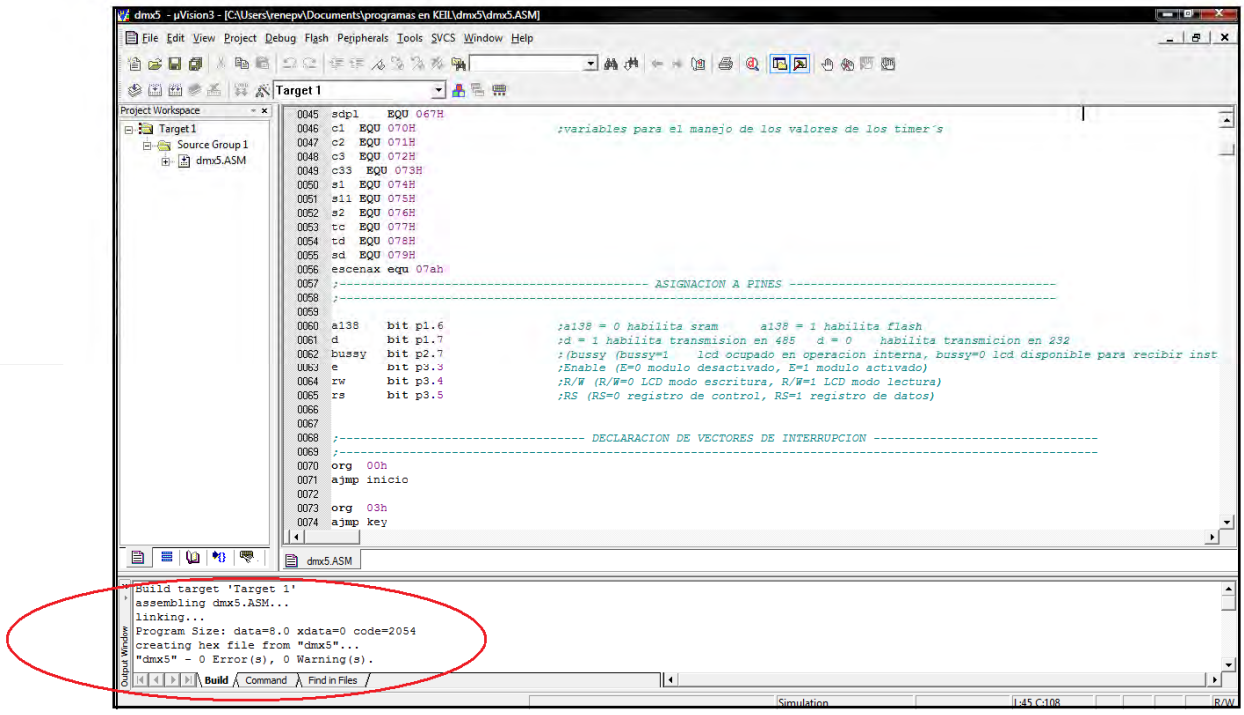


Figura 5.3. Compilación del Código Fuente en Ensamblador A51

- En el menú **Debug** está la opción **Start/Stop Debug Session**, la misma que nos permite realizar la simulación para ver lo que ocurre en el microcontrolador cuando se implementa este programa. Figura 5.4.

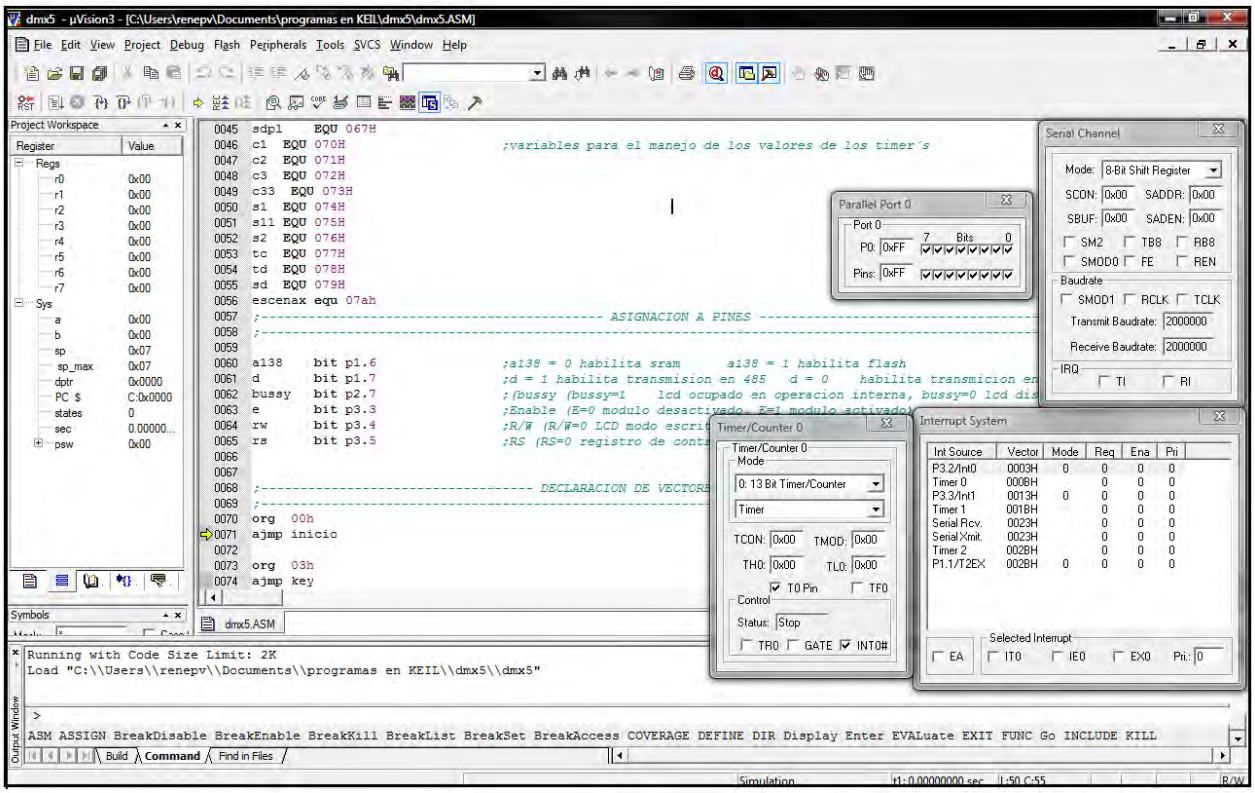


Figura 5.4. Simulación del Código Fuente Compilado

- En la pantalla de la izquierda, ahora se tienen los estados de los registros en lugar de los archivos que se ensamblaron para el proyecto.

5.2 Aplicación EZ-DOWNLOADER

Dentro de las herramientas de hardware se tiene una tarjeta programadora del AT89C51 y 52 de Atmel y su aplicación Ez-Downloader en PC, con la cual se pueden leer, programar y bloquear el código transferido a los microcontroladores.

La aplicación Ez-Downloader, Figura 5.5, toma el código hexadecimal generado por la aplicación keil, cuando ha sido ensamblado el código fuente y no existe error alguno, el cual envía a la tarjeta programadora vía RS-232 para la descarga de información al microcontrolador.

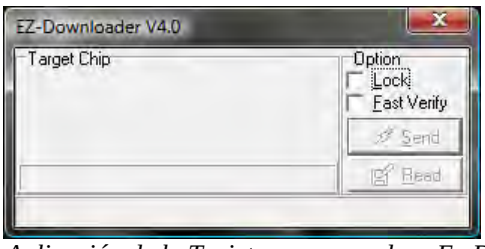


Figura 5.5. Aplicación de la Tarjeta programadora Ez-Downloader

Dentro de las utilidades que cuenta, esta la de poder verificar el contenido del micro, esto es para revisar que la información cargada es la deseada.

También es capaz de bloquear el código cargado para poder mantener la información restringida.

5.3 Descripción de Visual Basic 6.0

Visual Basic 6.0 es uno de los lenguajes de programación más populares del mundo. Es el sueño del programador de aplicaciones. Es un lenguaje con una interfaz grafica de usuario para crear aplicaciones en Windows, basado en el lenguaje Basic y la programación orientada a objetos.

La palabra **Visual** hace referencia al método que utiliza para crear la interfaz grafica de usuario. En lugar de escribir numerosas líneas de código para implementar una interfaz, se utiliza el ratón para arrastrar y colocar los objetos prefabricados al lugar deseado dentro de un formulario (zona de trabajo).

La palabra **Basic** hace referencia al lenguaje Basic (Beginners All purpose Symbolic Instruction Code), un lenguaje utilizado por muchos programadores.

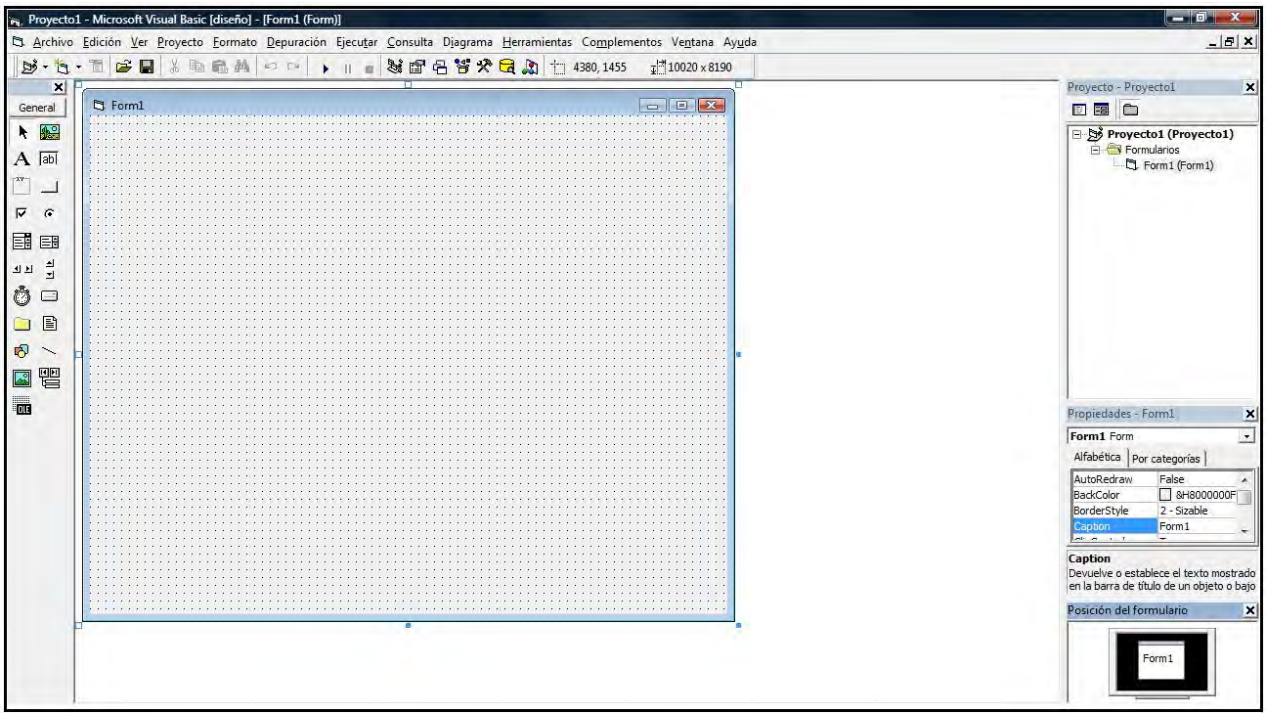


Figura 5.6. Entorno grafico de Desarrollo Visual Basic 6.0

Visual Basic ha evolucionado a partir del lenguaje Basic original y ahora contiene centenares de instrucciones, funciones y palabras clave, muchas de las cuales están directamente relacionadas con la interfaz grafica de Windows. Esto hace que abarque otras áreas, es utilizado también por Microsoft Excel, Microsoft Access y muchas otras aplicaciones.

Visual Basic permite crear programas de uso personal, para grupo de trabajo, para una empresa, aplicaciones distribuidas a través de Internet, aplicaciones de base de datos y muchas más.

5.3.1 Programas Secuenciales, Interactivos y Orientados a Eventos.

Existen distintos tipos de programas. En los inicios los programas eran secuénciales, los cuales al ejecutarse, leen los datos que necesitan, realiza los cálculos e imprime o guarda en el disco los resultados; entonces un programa secuencial no necesita ninguna intervención del usuario al ejecutarse. A este tipo de programas se les llama programas basados u orientados a procedimientos o algoritmos. Estos programas siguen utilizándose ampliamente en la actualidad, pero la difusión de la PC ha puesto de actualidad otros tipos de programación.

Programas interactivos exigen la intervención del usuario en tiempo de ejecución, bien para suministrar datos o para indicar al programa lo que debe hacer por medio de menús. Los programas interactivos limitan y orientan la acción del usuario.

Los programas orientados a eventos son los programas típicos de Windows, tales como Word, Excel y Power Point. Cuando uno de los programas ha arrancado, lo único que hace es quedarse en espera de las acciones del usuario, que en este caso son llamadas a eventos. Estos programas pasan la mayor parte de su tiempo esperando las acciones del usuario (eventos) y respondiendo a ellas. Las acciones que el usuario puede realizar en un momento determinado son variadísimas, y exigen un tipo especial de programación: la programación orientada a eventos.

5.3.2 El Entorno de Programación Visual Basic 6.0

Cuando se arranca VB 6.0 aparece en la pantalla una configuración similar a la mostrada en la Figura 5.7. En ella se pueden distinguir los siguientes elementos:

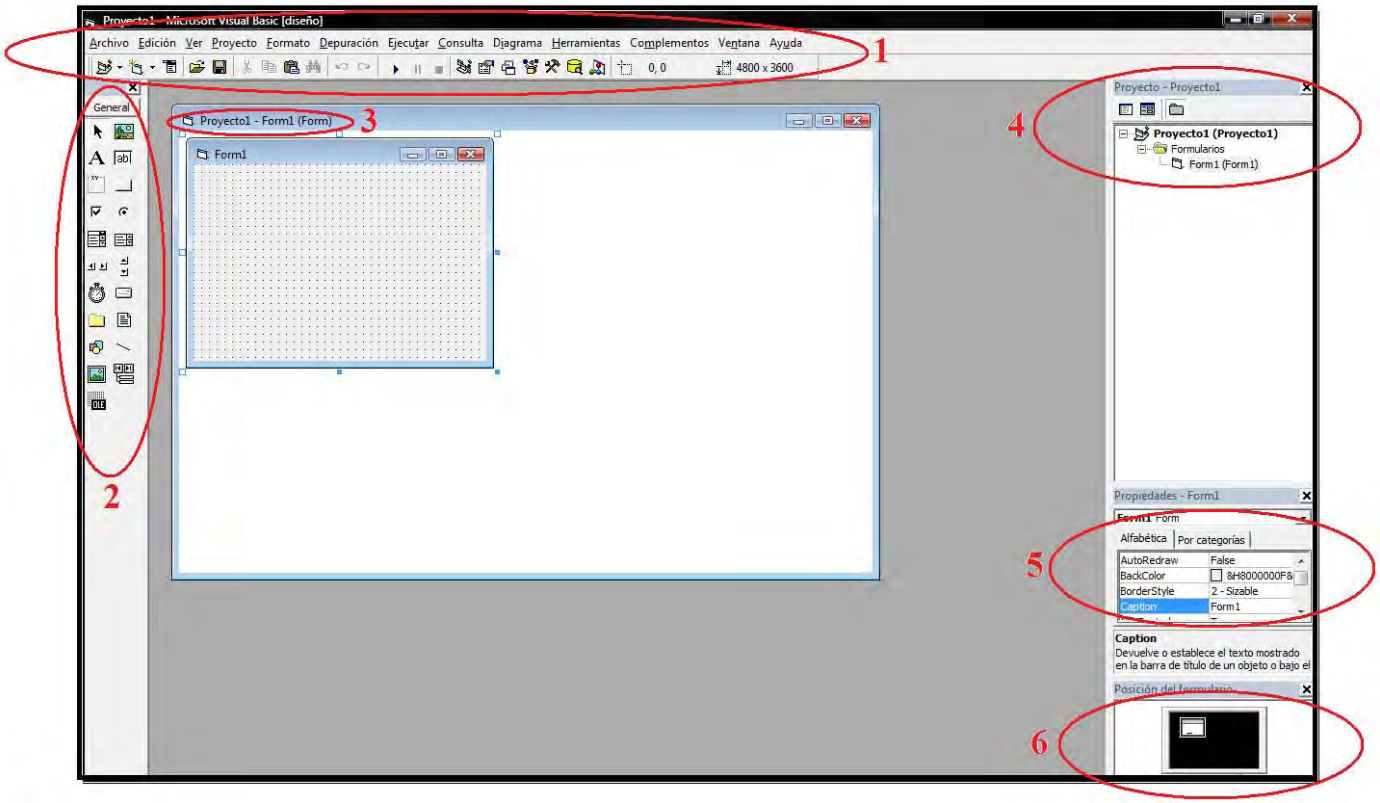


Figura 5.7. Elementos basicos del Entorno Grafico de Desarrollo VB 6.0

1. La barra de títulos, la barra de menús y la barra de herramientas de VB 6.0 en modo diseño (parte superior de la pantalla).
2. Caja de herramientas (toolbox) con los controles disponibles (a la izquierda de la ventana).
3. Formulario (form) en gris, en que se puede ir situando los controles (en el centro). Esta dotado de una rejilla (grid) para facilitar la alineación de los controles.
4. Ventana de proyecto, que muestra los formularios y otros módulos de programas que forman parte de la aplicación (arriba a la derecha).
5. Ventana de propiedades, en la que se pueden ver las propiedades del objeto seleccionado o del propio formulario (en el centro a la derecha). Si esta ventana no aparece, se puede hacer visible con la tecla (f4).
6. Ventana formlayout, que permite determinar la forma en que se abrirá la aplicación cuando comience a ejecutarse (abajo a la derecha). Existen otras ventanas para edición de código (code editor) y para ver variables en tiempo de ejecución con el depurador o debugger (ventanas immediate, locals y match). Todo este conjunto de herramientas y ventanas se lo que se llama un entorno integrado de desarrollo o IDE (integrated development environment).

Construir aplicaciones con VB 6.0 es muy sencillo: basta crear los controles en el formulario con ayuda de la toolbox y del ratón, establecer sus propiedades con ayuda de la ventana de

propiedades y programas el código que realice las acciones adecuadas en respuesta a los eventos o acciones que realice el usuario.

5.3.3 Programas para el Entorno Windows

VB 6.0 esta orientado a la realización de programas para Windows, pudiendo incorporar todos los elementos de este entorno informático: ventanas, botones, cajas de dialogo y de texto, botones de opción y de selección, barras de desplazamiento, gráficos, menús, etc.

5.3.4 Modo de Diseño y Modo de Ejecución

VB 6.0 puede trabajar de dos modos distintos: en modo de diseño y en modo de ejecución.

En modo de diseño el usuario construye interactivamente la aplicación, colocando controles en el formulario, definiendo sus propiedades, y desarrollando funciones para gestionar los eventos. Figura 5.8.

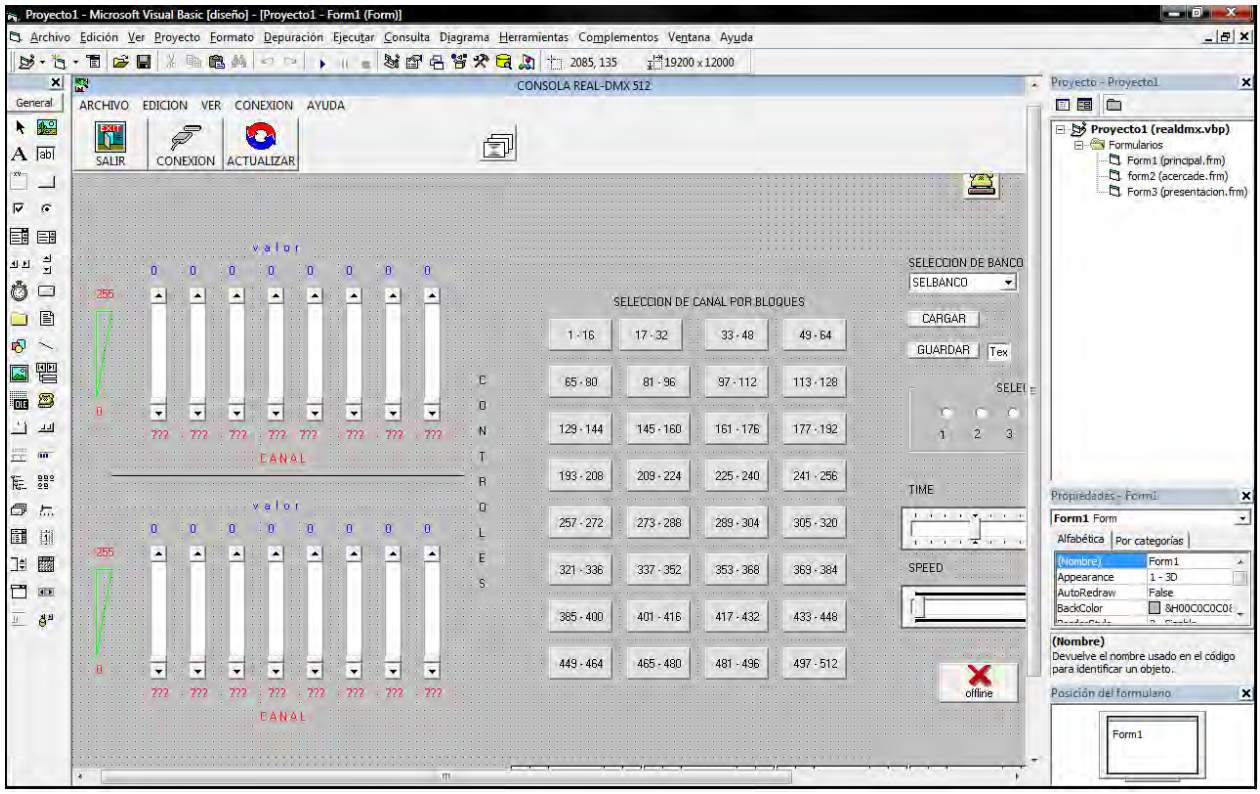


Figura 5.8. Entorno grafico, Modo de diseño VB 6.0

La aplicación se prueba en modo de ejecución: en este caso el usuario actúa sobre el programa (introduce eventos) y prueba como responde el programa. Hay algunas propiedades de los controles que deben establecer en modo de diseño, pero muchas otras pueden cambiarse en tiempo de ejecución desde el programa escrito. Figura 5.9

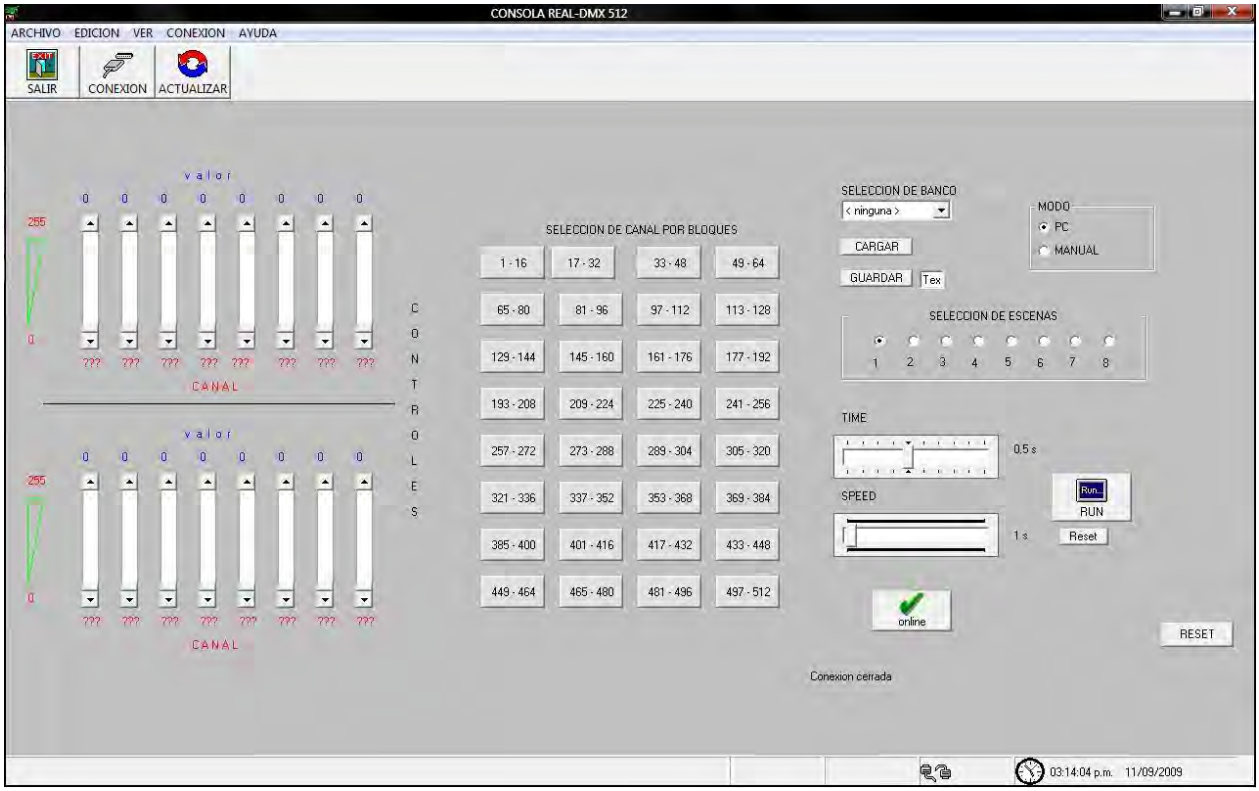


Figura 5.9. Entorno Grafico, Modo de Ejecución VB 6.0

También hay propiedades que solo pueden establecerse en modo de ejecución y que no son visibles en modo de diseño.

5.3.5 Formularios y Controles

Cada uno de los elementos gráficos que pueden formar parte de una aplicación típica de Windows son: los botones, las cajas de dialogo 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 VB6.0. Cada control debe tener un nombre a través del cual se puede hacer referencia a el en el programa. VB 6.0 proporciona nombres por defecto que el usuario puede modificar.

En la terminología de VB6.0 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 tener también un nombre, que puede crearse siguiendo las mismas reglas que para los controles.

5.3.6 Objetos y Propiedades

Los formularios y los distintos tipos de de controles son entidades genéricas de las que puede haber varios ejemplares concretos en cada programa. En programación orientada a objetos (más bien basada en objetos, habría que decir) se llama clase a estas entidades genéricas, mientras que se llama objeto cada ejemplar de una clase determinada. Por ejemplo, en un programa puede haber varios botones, cada uno de los cuales es un objeto del tipo de control command button, que seria la clase.

Cada formulario y cada tipo de control tiene un conjunto de propiedades que definen su aspecto grafico (tamaño, color, posición en la ventana, tipo y tamaño de letra, etc.) y su forma de responder a las acciones del usuario (si esta activo o no) por ejemplo cada propiedad tiene un nombre que viene ya definido por el lenguaje.

Por lo general, las propiedades de un objeto son datos que tienen valores lógicos (true, false) o numérico concretos, propios de ese objeto y distinto de las de otros objetos de su clase. Así pues, cada clase, tipo de objeto o control tiene su conjunto de propiedades, y cada objeto o control concreto tiene unos valores determinados para las propiedades de su clase. Casi todas las propiedades de los objetos pueden establecerse en tiempo de diseño y también casi siempre en tiempo de ejecución. En este segundo caso se accede a sus valores por medio de las sentencias del programa, en forma análoga a como se accede a cualquier variable en un lenguaje de programación. Para ciertas propiedades esta es la única forma de acceder a ellas. Por supuesto VB6.0 permite crear distinto tipos de variables. Se puede acceder a una propiedad de un objeto por medio del nombre del objeto a que pertenece, seguido de un punto y el nombre de la propiedad, como por ejemplo: optcolor.objname.

5.3.7 Nombres de Objetos

En principio cada objeto de VB6.0 debe tener un nombre, por medio del cual se hace referencia a dicho objeto. El nombre puede ser el que el usuario desee, e incluso VB6.0 proporciona nombres por defecto para los diversos controles. Estos nombres por defecto hacen referencia al tipo de control y van seguidos de un número que se incrementa a medida que se van introduciendo mas controles de ese tipo en el formulario (por ejemplo VScroll1, para una barra de desplazamiento –scroll bar- vertical. HScroll 1, para una barra horizontal, etc.).

Los nombres por defecto no son adecuados porque hacen referencia al tipo de control, pero no al uso que de dicho control esta haciendo el programador. Por ejemplo, si se utiliza una barra de desplazamiento para introducir una temperatura, conviene que su nombre haga referencia a la

palabra temperatura, y así cuando haya que utilizar ese nombre se sabrá exactamente a que control corresponde.

5.3.8 Eventos

Ya se ha dicho que las acciones del usuario sobre el programa se llaman eventos. Son eventos típicos al presionar sobre un botón, el hacer doble clic sobre nombre de un fichero para abrirlo, el arrastrar un icono, el pulsar una tecla o combinación de teclas, el elegir una opción de un menú, el escribir en una caja de texto, o simplemente mover el ratón. Cada vez que se produce un evento sobre un determinado tipo de control, VB6.0 arranca una determinada funcion o procedimiento que realiza la acción programada por el usuario para ese evento concreto. Estos procedimientos se llaman con un nombre que se forma a partir del nombre del objeto y el nombre del evento, separados por el caracter (_), como por ejemplo txtBox_click, que es el nombre del procedimiento que se ocupara de responder al evento clic en el objeto txtBox.

5.3.9 Métodos

Los métodos son funciones que también son llamadas desde programa, pero a diferencia de los procedimientos no son programadas por el usuario, sino que vienen ya preprogramadas con el lenguaje. Los métodos realizan tareas típicas, previsibles y comunes para todas las aplicaciones. De ahí que vengan con el lenguaje y que se libere al usuario de la tarea de programarlos. Cada tipo de objeto o de control tiene sus propios métodos.

Por ejemplo, los controles gráficos tienen un método llamado line que se encarga de dibujar líneas rectas. De la misma forma existe un método llamado circle que dibuja circunferencias y arcos de circunferencias. Es obvio que el dibujar líneas rectas o circunferencias es una tarea común para los programadores y que VB6.0 da ya resuelta.

5.3.10 Proyectos y Ficheros

Cada aplicación que se empieza a desarrollar en VB6.0 es un nuevo proyecto. Un proyecto comprende otros componentes mas sencillos, como por ejemplo los formularios (que son las ventanas de la interface de usuario de la nueva aplicación) y los módulos (que son conjuntos de funciones y procedimientos sin interfase grafica de usuario).

¿Cómo se guarda un proyecto en el disco? Un proyecto se compone siempre de varios ficheros (al menos de dos) y hay que preocuparse de guarda cada uno de ellos en el directorio adecuado y con el nombre adecuado. Existe siempre un fichero con extensión *. Vbp(visual basic Project) que se crea con el comando file/save Project as. El fichero del proyecto contiene toda la información del conjunto. Además hay que crear un fichero por cada formulario y por cada

modulo que tenga el proyecto. Los ficheros de los formularios se crean con **file/save** filename as teniendo como extensión ***.frm**. Los ficheros de código o módulos se guardan también con el comando **file/save** filename as y tienen como extensión ***.bas** si se trata de un modulo estándar o ***.cls** si se trata de un modulo de clase (class module). Presionando el botón save en la barra de herramientas se actualizan todos los ficheros del proyecto. Si no se habían guardado todavía en el disco, VB6.0 abre cajas de dialogo **Save as** por cada uno de los fichero que hay que guardar.

CAPITULO 6

6 DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE CONTROL

En este capitulo describiremos los circuitos integrados empleados y se definirá la forma de conexión de las líneas de control, datos y alimentación de cada uno de los elementos dentro del diseño del sistema.

6.1 Introducción

El campo de la iluminación escenografica es tan extensa, que los equipos diseñados para cumplir con necesidades especificas de trabajo, no siempre satisfacen los requerimientos de uso para alguna otra aplicación parecida, por lo que es frecuente que nos veamos en la necesidad de rediseñar, implementar y/o adaptar circuitos electrónicos que realicen las funciones o tareas específicas para que satisfagan los requerimientos adicionales sin interferir en las funciones para las cuales fue diseñado el equipo en su funcionamiento general.

Estos controles deben ser versátiles, es decir, su aplicación no debe restringirse solamente a las características requeridas por ciertos modelos o marcas de algunos equipos, además su implementación debe de ser sencilla y con componentes de fácil adquisición y bajo costo. Esto da la facilidad para poder integrar funciones adicionales que cubran algunas necesidades.

En el presente capitulo se diseña e implementa un sistema electrónico de control dmx512 que consta de tres etapas: entrada/salida de datos, almacenamiento de datos y comunicación.

A continuación se representan las etapas de forma general. Figura 6.1.

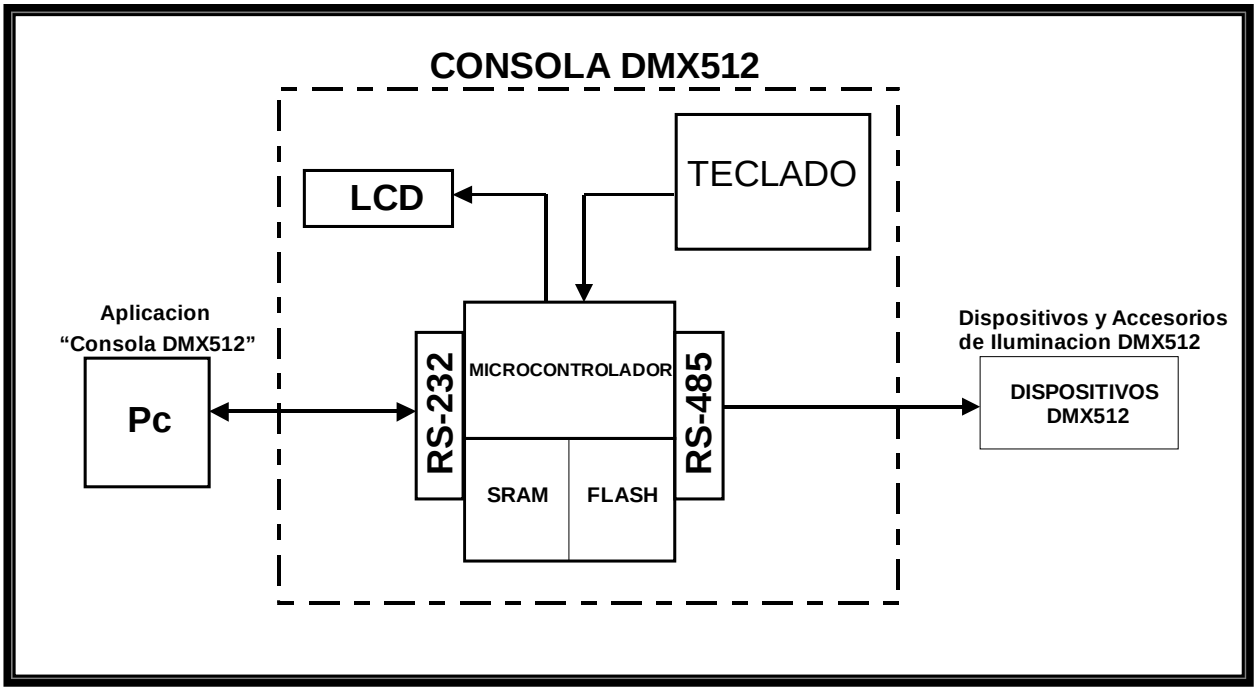


Figura 6.1. Diagrama de bloques del sistema de control DMX512

En la etapa de entrada/salida se requiere poder comunicar al usuario con el sistema, para lo cual se hace uso de un teclado matricial mediante el cual podemos ingresar datos al sistema y para poder visualizar los datos ingresados y resultados se necesitara un display de LCD, en el cual se representa la información al usuario.

En la siguiente etapa, surge la necesidad de almacenar los datos ingresados por el usuario y los datos procesados por el sistema de forma temporal o permanente; para ello se emplearan dos tipos de memorias: una para la transferencia de información de rápido acceso por el sistema y otra de amplia capacidad para el resguardo permanente de los cambios hechos por el usuario.

Después de ingresar, visualizar y guardar la información en el sistema, se requiere de hacer fluir la información a los dispositivos DMX512 mediante un canal de comunicación con protocolo RS-485 y cuando se necesita del apoyo de una PC se enlazara a través de otro canal de comunicación con protocolo RS-232.

6.1.1 Configuración Básica para Funcionamiento

Para poder iniciar nuestro diseño es necesario configurar las conexiones básicas de funcionalidad del microcontrolador AT89C52 descrito en el capitulo 3. A continuación se muestran las conexiones de alimentación la cual debe tener 5vdc a Vcc y tierra en GND, el reloj esta compuesto de un cristal de cuarzo de 24Mhz conectado entre XTAL1 Y XTAL2, un arreglo lógico para la terminal RESET y de no contar con memoria externa de programa debe de conectarse un nivel alto el pin $\overline{EA}$/Vpp. Figura 6.2.

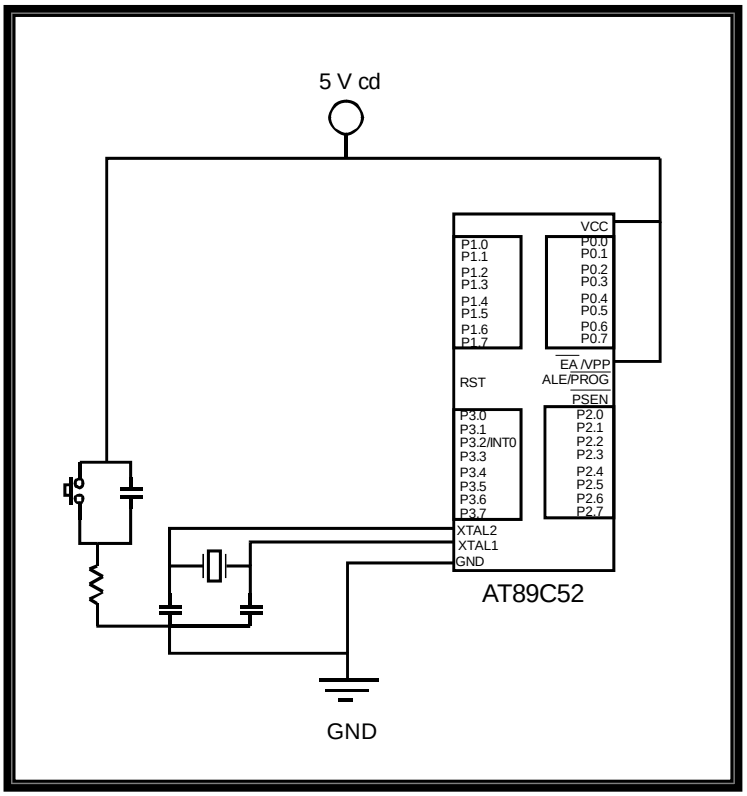


Figura 6.2. Conexión básica de operación del AT89C52, sin memoria de programa externo

6.2 Etapa de entrada y salida de datos

Todo sistema de control, que requiera la manipulación de información o visualización, se apoya en periféricos externos desde los más sencillos como un botón y una lámpara hasta más complejos como un: teclado, ratón, apuntador, lector óptico, scanner, monitor, impresora, etc.

Para el diseño del trabajo hemos escogido un teclado matricial 4x4, que no es mas que un arreglo de botones con una configuración líneas-columnas, para la introducción de datos y un display inteligente conocido como LCD de 16x2, es decir 2 filas de 16 caracteres cada una, para la visualización de la información.

6.2.1 Conexión del teclado matricial

La introducción de datos al sistema, se hará por medio de un teclado matricial 4x4, este nos proporciona una interfaz de entrada óptima para el usuario.

La gestión del teclado matricial puede hacerse directamente por medio del microcontrolador, sin embargo esto implica el uso de un puerto completo 8 bits, es decir 8 pines del microcontrolador, lo cual nos reduce líneas de control que pudieran emplearse en otro elemento. Por lo que se ha optado por el uso de un decodificador de teclado matricial. El circuito integrado 74C922 nos servirá para este propósito, ya que nos reduce el numero de pines a 4 dejando libre 4 pines para uso de señales de control.

La forma de conectarse se presenta en la Figura 6.3.

Se conectara las filas y columnas del teclado a las entradas del codificador (ROW Y COLUMN respectivamente), las salidas (DATA OUT) de este se conectaran al puerto 1 (P1.0 – P1.3).

El decodificador 74C922 explorara el teclado, cuando se halla oprimido una tecla generara un señal de interrupción (Data Available, DA), la cual indica que hay un dato valido y se tendrá que habilitar sus salidas por medio de $\overline{OE}$, en nivel bajo, para colocar el código correspondiente de la tecla presionada a la salida del circuito. De igual manera esta interrupción será atendida por el microcontrolador a través de la interrupción INT0, en el instante en que el decodificador valide un dato y habilite sus salidas, este será leído por el nible bajo del puerto 1 y a su vez será interpretado por el microcontrolador para dar como resultado una acción.

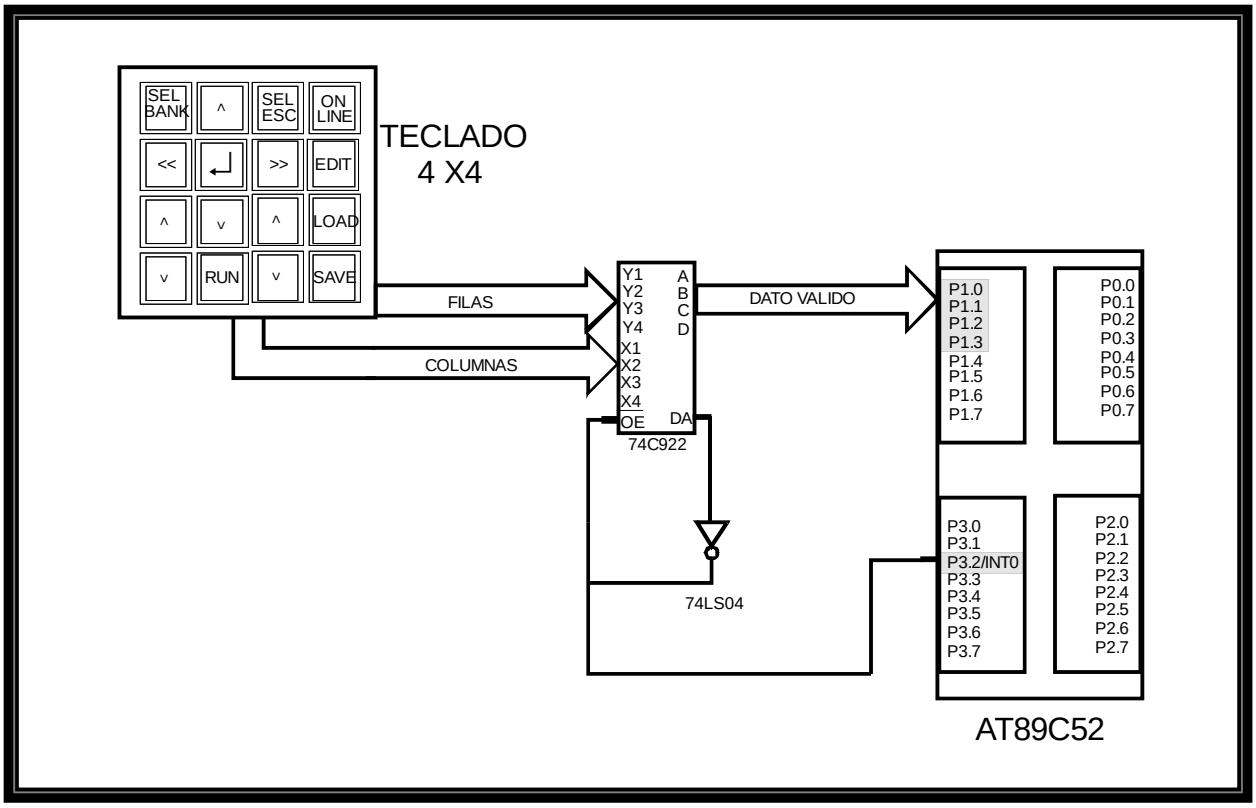


Figura 6.3. Conexión del teclado matricial con decodificador

El uso del circuito integrado 74LS04 tiene la finalidad de reducir líneas de control del microcontrolador, haciendo que el decodificador atienda su señal de interrupción de dato valido (Data Available, DA) y se responda habilitando la salida del dato (DATA OUT, $\overline{OE}$).

6.2.2 Conexión de Display LCD

En la visualización de información se empleara un display LCD de 16 x 2. Estos son compactos y necesitan muy pocos componentes externos para su funcionamiento. Cuenta con 3 líneas de alimentación, un bus de 8 bits para datos y 3 líneas de control.

El modulo del LCD es alimentado por 5 Vdc en el pin 1 VCC y tierra en el pin 2 GND; además tiene en el pin 3 VSS que sirve para contraste del LCD por medio de un potenciómetro ajustable.

Para poder enviar la información al LCD se utilizará el puerto 2 del microcontrolador, aunque el LCD admite datos en bloques de 4 bits y 8 bits, se enviaran los datos en bloque de 8 bits. Cabe señalar que el puerto 2 esta compartido con el bus de direcciones altas para el direccionamiento de la memoria externa.

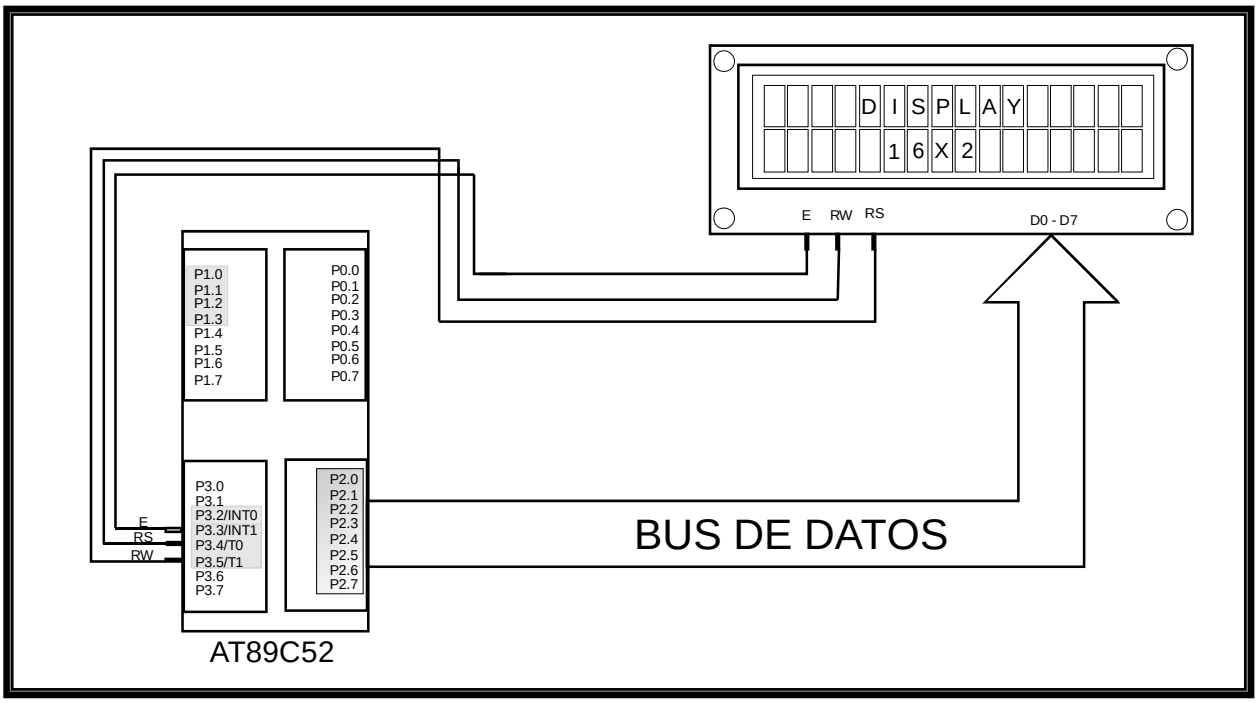


Figura 6.4. Conexión de LCD

Las señales de control del LCD: E, RW, RS serán manipuladas por medio del microcontrolador con los pines p3.3, p3.4 y p3.5 respectivamente. Figura 6.4

La señal de control E habilita el LCD cuando E = 1, esto sucede cuando se tiene un dato o comando valido procedente del microcontrolador a través del puerto 2 para la captura en el display. Tabla 6.1

E	**A138**	**GND**		$\overline{CS1}$	$\overline{CE}$	**DISPOSITIVO**
G2A, G2B	**A**	**B**	**C**	**Y0**	**Y1**	
0	0	0	0	0	1	MEM. RAM
0	1	0	0	1	0	MEM. FLASH
1	0	0	0	1	1	LCD
1	1	0	0	1	1	LCD

Tabla 6.1. Tabla de verdad para el control del LCD

La señal de control RW establece el modo en el que opera, ya sea modo de lectura (RW = 1) o escritura (RW = 0).

La señal de control RS indica la selección de registros. Cuando RS = 1 se reciben datos, cuando RS = 0 se recibe un comando.

6.3 Conexión de Unidades para el Almacenamiento de Datos

En los sistemas digitales es necesario contar con medios de almacenamiento de datos, en el caso particular, el uso de una memoria SRAM y FLASH para poder guardar y recuperar datos de forma rápida; además de no perder información cuando el circuito no cuente con alimentación.

6.3.1 Uso de Memoria SRAM

Debido a que el protocolo DMX puede controlar 512 dispositivos, esto implica 512 datos por universo o 512 bytes, y considerando que un banco esta formado por 8 universos esto serian 4096 bytes por banco. Muchas veces los microcontroladores no disponen de tanto espacio de memoria por lo que es necesario ampliarla externamente. Para complementar la capacidad de memoria, requerimos que esta sea una SRAM puesto que es de rápido acceso, lo que ayudara a actualizar y manipular los datos más ágilmente.

Se utilizara una memoria HM6264A de 8K bytes (8192 bytes) de capacidad, para poder manipular los datos pertenecientes a un banco. Esta dispone de 13 líneas de direccionamiento (A0-A12), un bus de datos de 8 bits (D0-D7) y 4 líneas de control ($\overline{CS1}$, CS2, $\overline{OE}$ y $\overline{WR}$).

El microcontrolador dispone del puerto 0 y 2, para direccionar memoria externa, además que el bus de datos esta multiplexado en tiempo con el puerto 0. Es decir comparte el mismo bus para salida de direcciones bajas y para salida/entrada de datos.

Para conectar la memoria al microcontrolador, se demultiplexara la parte de las direcciones bajas y los datos, para ello emplearemos el latch tipo D 74HC573 para separar los buses.

La forma de conectar los componentes se describe a continuación, se conectara el puerto0 (p0.0 – p0.7) del microcontrolador a las entradas (D0 – D7) del latch y sus salidas (Q0 –Q7) se conectaran a las líneas de direcciones bajas de la memoria SRAM (A0 – A7). Las direcciones altas (A8 – A12) faltantes para direccionar la memoria se toman del puerto2 (p2.0 – p2.4) del microcontrolador. La línea de control ALE del microcontrolador se conecta a línea LE del latch, esta línea indica el momento en el que esta presente una dirección, es decir el momento en el cual el latch debe capturar la parte baja de una dirección. El puerto0 (p0.0 – p0.7) del microcontrolador se conectara también al bus de datos de la memoria SRAM (D0 – D7).

Para poder compartir el puerto 2 del microcontrolador como bus de datos y direcciones con el LCD y la memoria SRAM respectivamente se realizara una lógica de control apoyándonos con el demultiplexor 74LS138 y las líneas de control generadas por el microcontrolador, etiquetadas como a138 (p1.6) y E (p3.3). Estas líneas de control generadas nos servirán para seleccionar entre el uso del LCD o memoria, siendo a138 empleada para seleccionar el tipo de memoria SRAM o FLASH y E empleada para diferenciar entre el uso de memoria o LCD.

Las líneas de control de la memoria para la escritura y lectura de la misma, son generadas por el microcontrolador en sus respectivas líneas de control de salida p3.6 para $\overline{WR}$ y p3.7 para $\overline{RD}$, empleadas en el uso de memoria externa. Figura 6.5

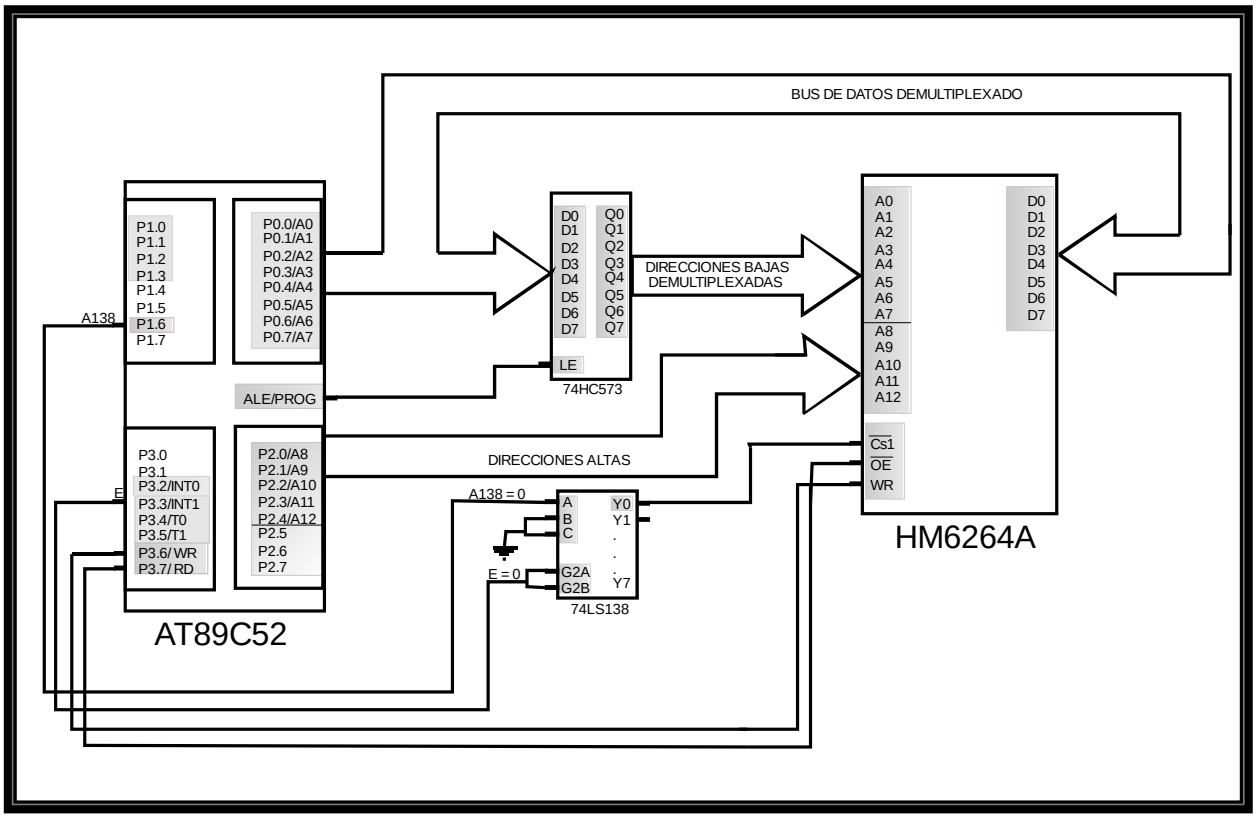


Figura 6.5. Conexión de la memoria SRAM

Las consideraciones para las líneas de control de la memoria SRAM en la lógica de control es la siguiente:

1) Las líneas de control $\overline{CS1}$ habilita la memoria SRAM, esta señal es generada por un demultiplexor 74LS138 en la salida Y0 = 0 cuando A138 = 0 y E = 0. La línea de control CS2 siempre esta habilitada en conexión a Vcc. Tabla 6.2

E	A138	GND		$\overline{CS1}$	$\overline{CE}$	DISPOSITIVO
G2A, G2B	A	B	C	Y0	Y1	
0	0	0	0	0	1	MEM. RAM
0	1	0	0	1	0	MEM. FLASH
1	0	0	0	1	1	LCD
1	1	0	0	1	1	LCD

Tabla 6.2. Tabla de verdad para el control de Memoria RAM

2) La línea de control $\overline{OE}$ habilita la salida de datos de la memoria, y esta controlada directamente por la señal $\overline{RD}$ del microcontrolador cuando se hace la petición de lectura externa de memoria.

3) La línea de control $\overline{WE}$ habilita la entrada de datos a la memoria, y esta controlada directamente por la señal $\overline{WR}$ del microcontrolador cuando se desea guarda un dato en memoria externa.

6.3.2 Uso de Memoria FLASH

Para guardar los cambios realizados en las escenas de los bancos de forma permanente se hace necesario contar con una memoria capaz de ser grabable y borrable dentro del sistema sin tener que desmontarla. Con la finalidad de retener la información aun sin estar alimentado.

Recordando que un banco requiere de 4096 bytes de espacio en memoria para ser guardado y para el sistema se contempla almacenar 6 bancos, esto como sistema básico de almacenaje, por lo tanto se requiere un espacio mayor a 24576 bytes (24 Kbytes).

Se utilizara una memoria FLASH de 1Mbyte (1024 Kbytes) de capacidad, para poder almacenar los datos permanentemente de los bancos. Esta dispone de 17 líneas de direccionamiento (A0-A16), un bus de datos de 8 bits (D0-D7) y 3 líneas de control ($\overline{CE}$, $\overline{OE}$ y $\overline{WR}$).

Al igual que en la memoria SRAM, comparte el mismo bus de direccionamiento y el bus de datos multiplexado en tiempo, las direcciones altas (A8 – A15) faltantes para direccionar la memoria se toman del puerto2 (p2.0 – p2.7). La dirección A16 para este proyecto no la usamos por lo que solo se conecta a GND. Además que dos de tres líneas de control son también compartidas como lo son $\overline{WR}$ y $\overline{OE}$. Figura 6.6.

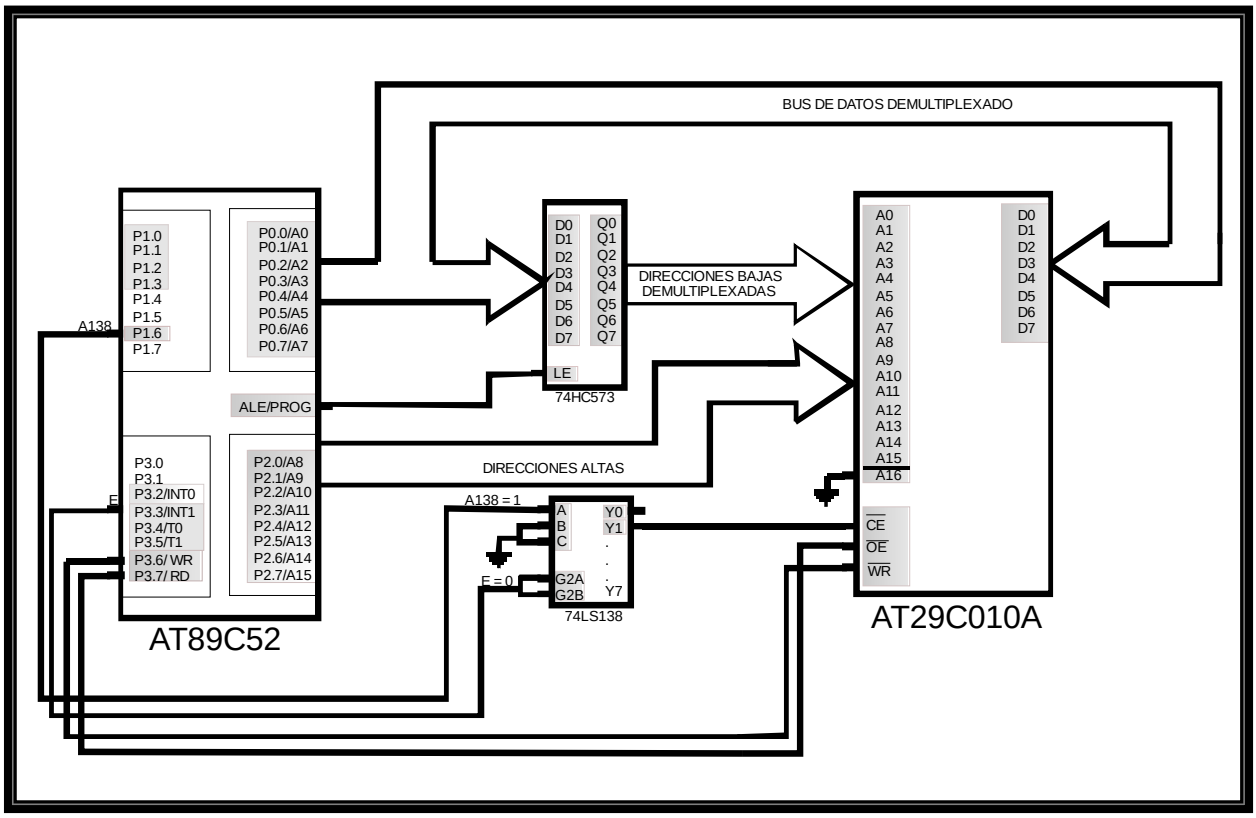


Figura 6.6. Conexión de la memoria FLASH

Las consideraciones para las líneas de control de la memoria FLASH en la lógica de control es la siguiente:

1) La línea de control $\overline{CE}$ habilita la memoria FLASH, esta señal es generada por un demultiplexor 74LS138 en la salida Y1 = 0 cuando A138 = 1 y E = 0. Tabla 6.3

E	A138	GND		$\overline{CS1}$	$\overline{CE}$	DISPOSITIVO
G2A, G2B	A	B	C	Y0	Y1	
0	0	0	0	0	1	MEM. RAM
0	1	0	0	1	0	MEM. FLASH
1	0	0	0	1	1	LCD
1	1	0	0	1	1	LCD

Tabla 6.3. Tabla de verdad para el control de Memoria FLASH

2) La línea de control $\overline{OE}$ habilita la salida de datos de la memoria, y esta controlada directamente por la señal $\overline{RD}$ del microcontrolador cuando se hace la petición de lectura externa de memoria.

3) La línea de control $\overline{WE}$ habilita la entrada de datos a la memoria, y esta controlada directamente por la señal $\overline{WR}$ del microcontrolador cuando se desea guarda un dato en memoria externa.

6.4 Conexión de interfaces de Comunicación

La comunicación entre sistemas digitales es importante para el intercambio y flujo de información. Para el sistema de control se requiere de compartir información con la PC y de enviar información a todos los dispositivos a controlar por lo que nos apoyaremos en el uso de una interfaz que emplee el estándar RS-485 para envío de datos a los dispositivos DMX512 y una interfaz que emplee el estándar RS-232 para comunicarse con PC.

6.4.1 Uso de estándar RS-485

El empleo del estándar RS-485, se debe a los requerimientos implementados para transmisión del protocolo DMX512, tema descrito anteriormente.

Esta comunicación será unidireccional tipo simplex, ya que únicamente requeriremos transmitir la información a los equipos y accesorios de iluminación; por lo que no se espera respuesta alguna de estos elementos.

Se utilizara el CI SN75176B que es un transmisor/receptor bidireccional, diseñado para transmisión de datos en líneas balanceadas. Este cuenta con entrada y salida de datos controlables; en nuestro caso, la línea correspondiente a la recepción de datos no se empleara.

Este dispositivo se conectara a una línea diferencial balanceada por medio de un conector xlr3 hembra tipo cannon (A= +, B= -, GND). Dispone de una entrada de datos (D) para transmitir, habilitada por una señal de control (DE) activa en nivel alto que permite la transmisión de los datos.

La conexión de la línea de entrada D recibirá los datos a transmitir de la línea TxD, pin p3.1, del microcontrolador y será controlada la transmisión por la señal DE, pin p1.7 del micro, esta será la señal de entrada de control DE. Figura 6.7.

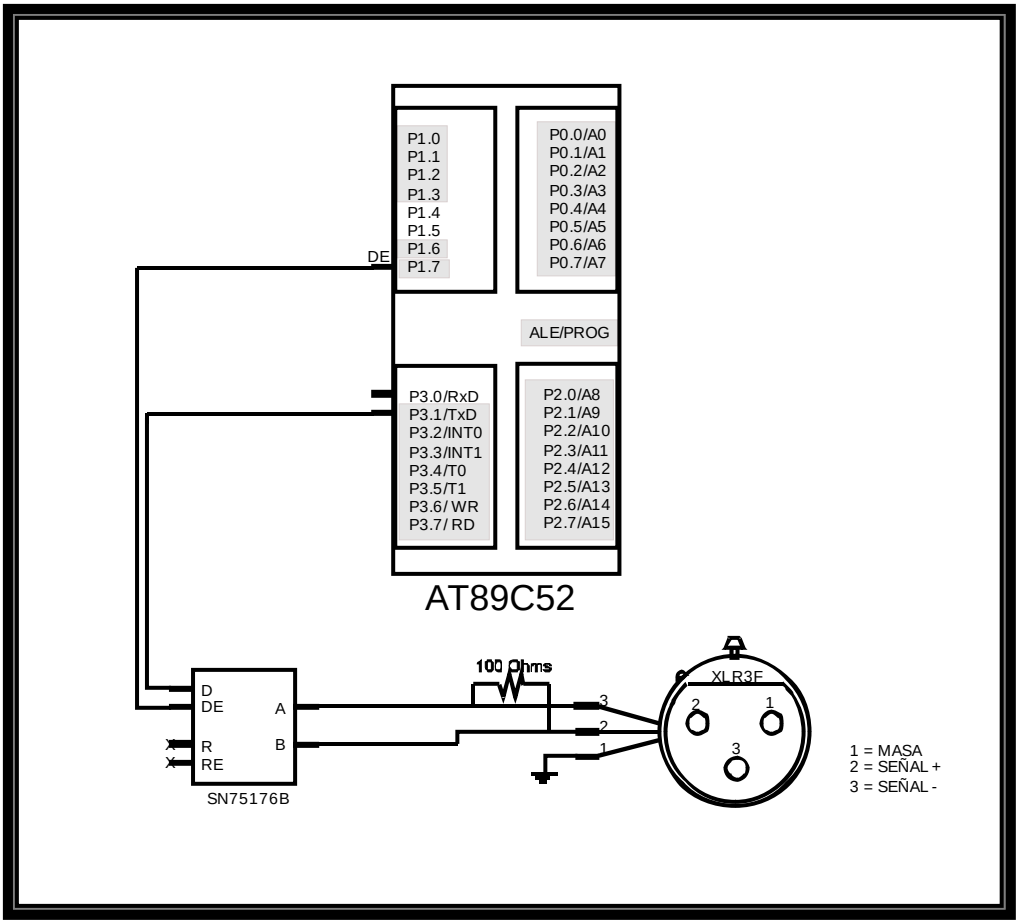


Figura 6.7. Conexión del transmisor/receptor SN75176B y Conector XLR3 Hembra

6.4.2 Uso de estándar RS-232

La comunicación con la PC, se establecerá por medio del popular estándar RS-232. Utilizaremos el CI MAX232, que es un transmisor/receptor dual. Tiene una línea para la entrada de datos en niveles TTL de 5v, T1IN y su respectiva salida T1OUT en niveles del estándar 15v. En la parte de recepción de datos tiene una entrada de datos R1IN en niveles del estándar y de igual manera su correspondiente salida R1OUT en niveles TTL, este dispositivo no dispone de señales de control, es decir, no cuenta con habilitadores que controlen el flujo de transmisión/recepción de datos.

Este es un problema al momento de separar los datos a transmitir, por que el microcontrolador únicamente disponemos de una línea de transmisión y esta debe compartirse con los dos dispositivos de comunicación, pero no así la información.

Para poder diferenciar la información correspondiente a cada dispositivo, se incluyo un buffer triestado cuádruple 74LS125, para poder controlar el instante en cual la información le corresponde a cada uno de los dispositivos, este problema únicamente lo tenemos cuando se

transmiten datos y no cuando se reciben. Cada buffer tiene una entrada (A1), salida (Y1) y línea de control ($\overline{C1}$), entrada activa en bajo.

Para hacer la conexión nos apoyaremos en la línea de control DE, empleada en el SN75176B. La línea TxD del micro llegara a la entrada del buffer A1 y saldrá por Y1 hacia T1IN del MAX 232; la línea $\overline{C1}$ será controlada por la señal DE proveniente del microcontrolador la que deberá deshabilitar el paso de información cuando la línea se encuentre en DE = 1 y dejarla pasar cuando sea DE = 0. La línea de entrada RxD del micro con la línea de salida R1OUT del MAX232.

El enlace se realizara a través del conector db9, sin acoplamiento por hardware, utilizando una comunicación directa punto a punto. Figura 6.8

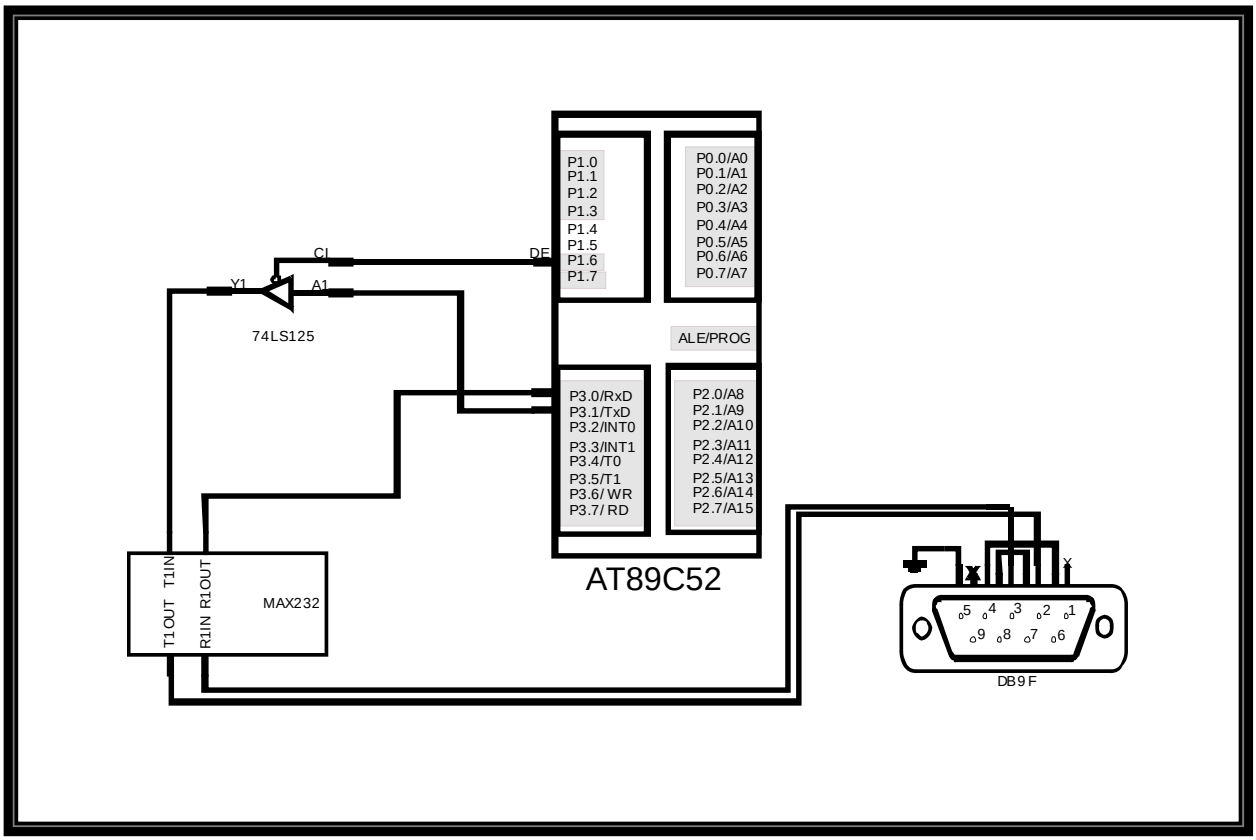


Figura 6.8. Conexión del transmisor/receptor MAX232 y conector DB9 hembra

6.5 Esquema general del sistema.

Se anexa el sistema completo para referencia general de conexiones y C.I´s. Figura 6.9

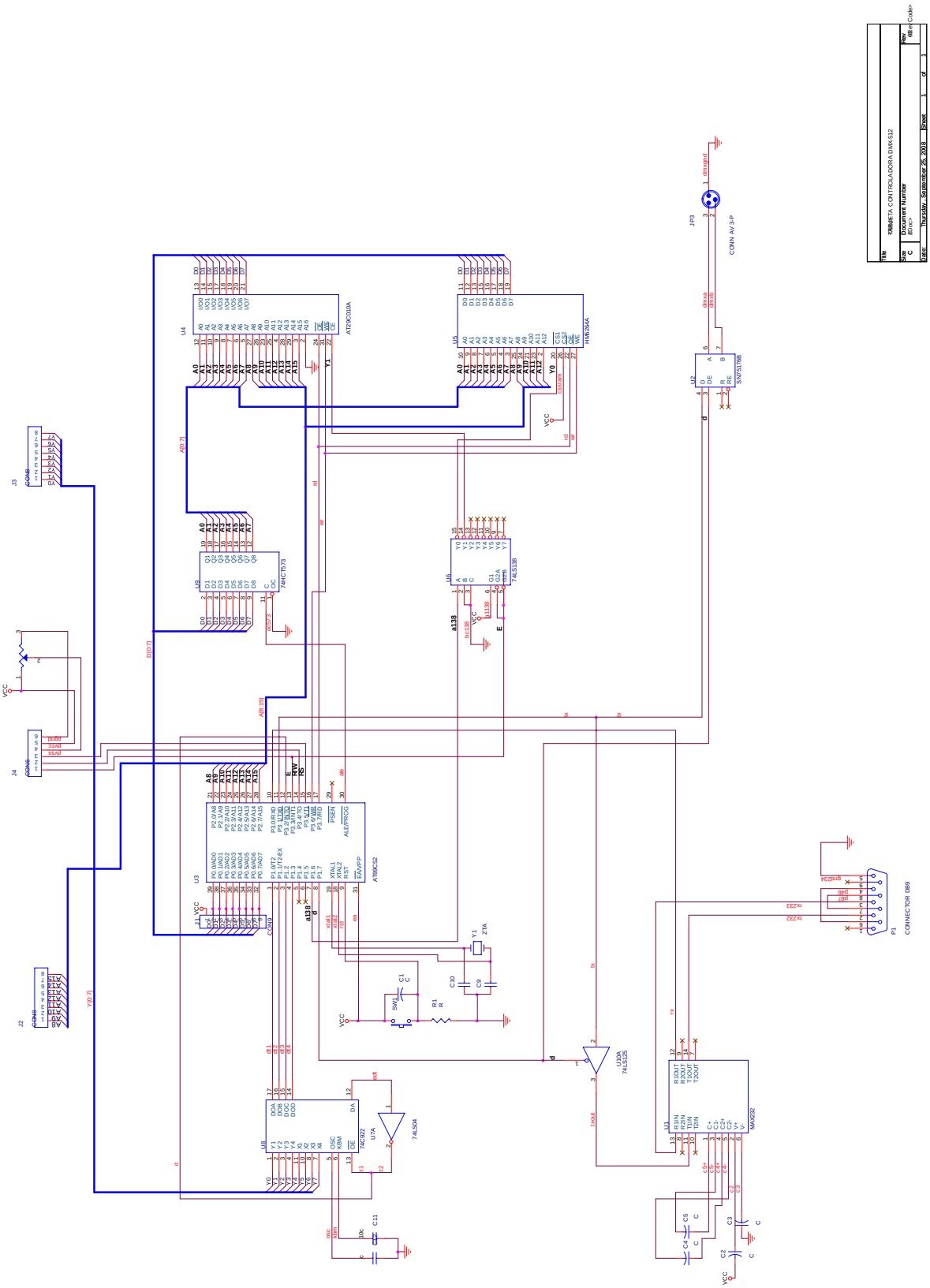


Figura 6.9. Diagrama completo de conexiones

CAPITULO 7

7 INTERFAZ GRAFICA DE USUARIO

En este capitulo se desarrolla una aplicación con interfaz grafica con la cual el usuario pueda interactuar con la PC y enlazarse por medio de la interfaz RS-232 hacia el sistema de control. En la elaboración de la interfaz grafica se busca tener la misma funcionalidad que tiene el sistema de control DMX512 (consola), por lo que se programara las mismas funciones a través de controles gráficos.

7.1 Programación

Para poder llevar acabo una aplicación con interfaz grafica de usuario que pueda ejecutarse a través de un PC, Figura 7.1, es necesario hacer algunas consideraciones como: bajo que sistema operativo queremos ejecutar la aplicación, ya que hoy en día son diversos los sistemas operativos con los que un PC puede contar y también existen diversos lenguajes de programación en los que puede elaborarse la aplicación, por lo que se tiene que elegir el que nos facilite llevar a cabo la aplicación.

Para el desarrollo de la aplicación, se ha elegido al programa Visual Basic 6.0 por ser un lenguaje de programación orientado a objetos y eventos en Windows, además de estar familiarizados con este lenguaje, y es posible ejecutarlo bajo el entorno del sistema operativo Windows XP y VISTA siendo estos muy populares en las PC's.

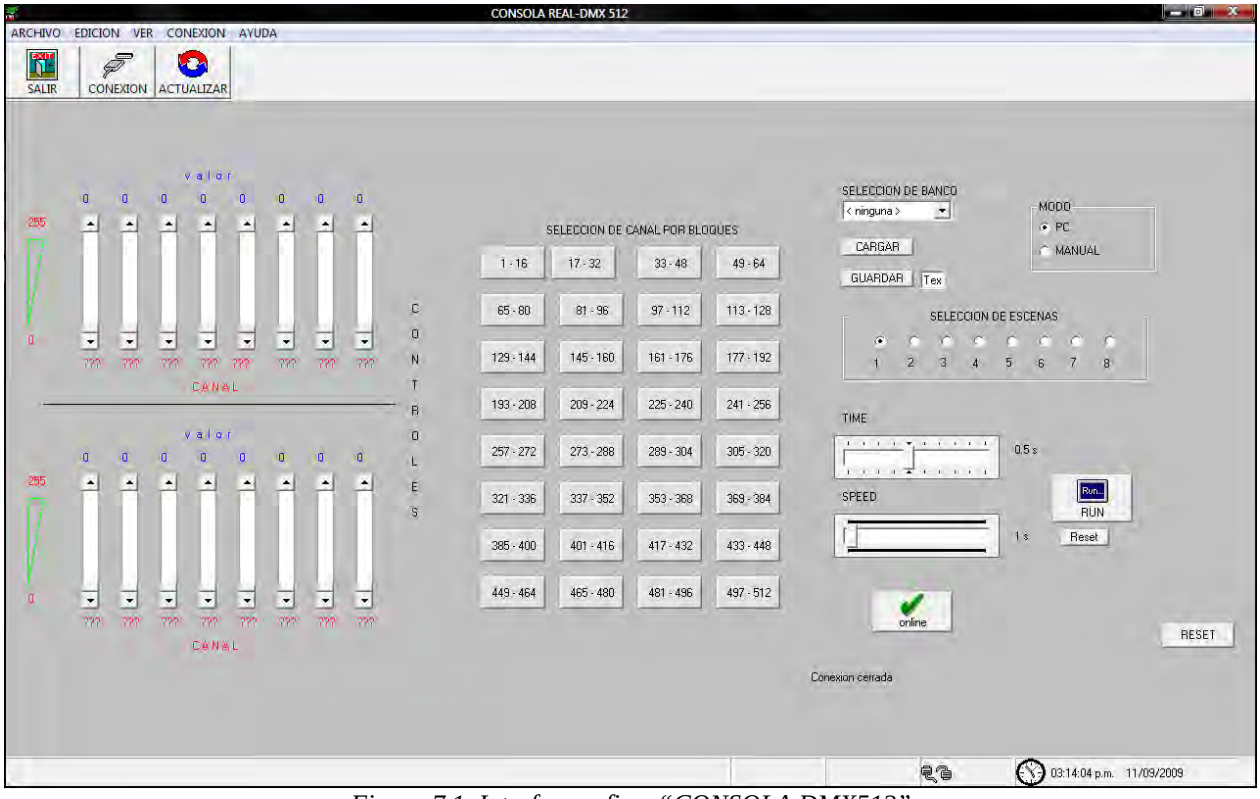


Figura 7.1. Interfaz grafica "CONSOLA DMX512"

Para poder llevar acabo la interfaz grafica de usuario (GUI) con la que contara la aplicación, se tiene tomar en cuenta las funciones que deseamos controlar y ver el medio por el cual lo podemos representar en la interfaz grafica.

Funciones a controlar:

1) Selección de bancos
2) Cargar bancos
3) Selección de universo (escena)
4) Guardar universos (escenas)
5) Selección de canales
6) Variación de valores
7) Modo de control
8) Variación de velocidad
9) Variación de tiempo
10) Reset de tiempos
11) Reset general

A cada una de estas funciones se asignara un medio para poder llevar acabo el control por parte de usuario a través de la aplicación.

A continuación se describe la asignación a cada una de las funciones.

7.1.1 Selección y Carga de Banco

Un banco esta compuesto de 8 universos. Un banco es proyectado cuando cada uno de los universos que lo componen envía su información a los dispositivos DMX512 en un lapso de tiempo ajustable y repitiéndose esto cíclicamente.

Para seleccionar un banco, lo podemos hacer mediante una caja de texto multilínea, usando el control combobox “selbanco”, etiquetado como <SELECCIÓN DE BANCO>. Este control despliega una lista de los bancos disponibles que se encuentran almacenados. Figura 7.2.

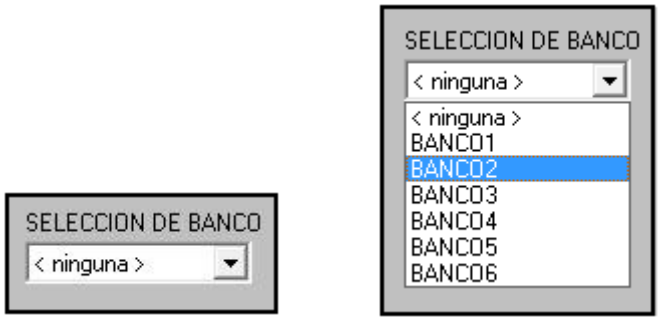


Figura 7.2. Selección de bancos

Una vez seleccionado el banco con el que se quiere trabajar, se tendrá que cargar la información contenida. Para ello se dispondrá del control commandbutton “cargar”, etiquetado como <CARGAR>, con esto se podrá cargar la información del banco seleccionado en pantalla. Figura 7.3.

Figura 7.3. Carga de banco

7.1.2 Selección y Guardado de Universo (escena)

Un universo, como se menciono anteriormente, es el conjunto de los 512 canales de datos que componen al protocolo DMX512.

Para elegir el universo que queremos editar perteneciente a un banco precargado, se tendrá que optar entre las ocho escenas existentes. Se utilizara un arreglo de controles optionbutton “esc ( )”, etiquetado como <SELECCIÓN DE ESCENAS>, la elección de este control, es debido a que se puede seleccionar únicamente una escena de entre los 8 posibles. Figura 7.4

Figura 7.4. Selección de escena

La escena seleccionada, cargara sus datos en los controles deslizables, actualizándose estos cada que se elija entre las ocho escenas. Después que haber cargado y editado la escena, se podrán salvar los cambios hechos, mediante el control commandbutton “guardar”, etiquetado como <GUARDAR>. Figura 7.5.

Figura 7.5. Guardado de escena

Al finalizar el guardado de la escena, se avisara a través de la caja de texto, control textbox “text1” la escena que ha sido guardada recientemente.

7.1.3 Selección de Canales

Para poder seleccionar un canal de entre los 512 canales que conforman el universo DMX512, se dispondrá de 32 bloques en los cuales se distribuirán los 512 canales quedando los bloques de 16 canales cada uno. Para este propósito se empleara una matriz de controles commandbutton “bloques ( )”, etiquetada como <SELECCIÓN DE CANAL POR BLOQUES>, cada botón esta etiquetado con el rango de canales que contiene. Figura 7.6

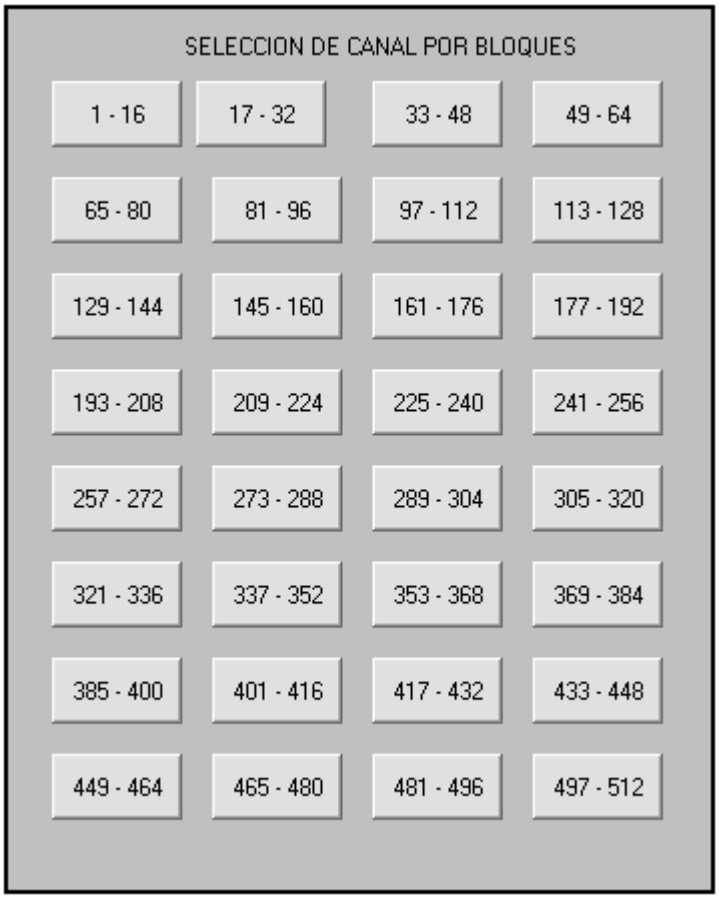


Figura 7.6. Selección de bloque

Dependiendo del bloque seleccionado, este actualizara los deslizables correspondientes que en el siguiente apartado se mencionan.

7.1.4 Variación de Valores

Una de las partes importantes en el control, radica en los valores de los canales. Para poder variar el valor contenido en cada canal y diferenciarlo de entre los 512 canales pertenecientes al universo, se utilizara un arreglo de controles vscrollbar “deslizable ( )”, representados por barras de desplazamiento y etiquetado con <CONTROLES>. Figura 7.7.

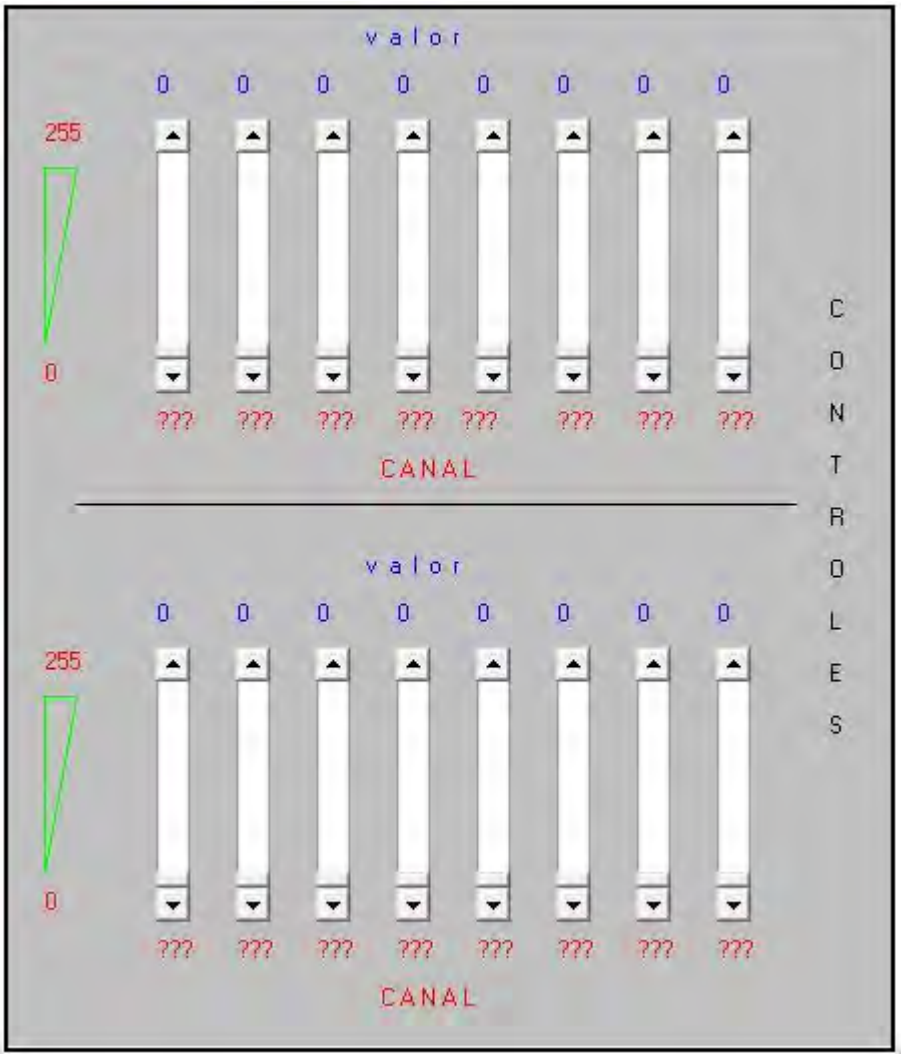


Figura 7.7. Variación de valores por medio de controles deslizables

Los deslizables deberán variar los valores entre 0 y 255 por canal, estos valores son permitidos en el protocolo DMX512; además estos deslizables están ligados a la elección de canales por bloque, por que actualizan los datos pertenecientes al bloque seleccionado.

Las etiquetas inferiores en cada deslizable llamadas <CANALES >, representa el canal actual. Este se actualiza dependiendo del bloque seleccionado y se asigna el canal según la ubicación en la que se encuentre.

Las etiquetas superiores en cada deslizable llamada <VALORES>, representan el valor actual del deslizable. Figura 7.8.

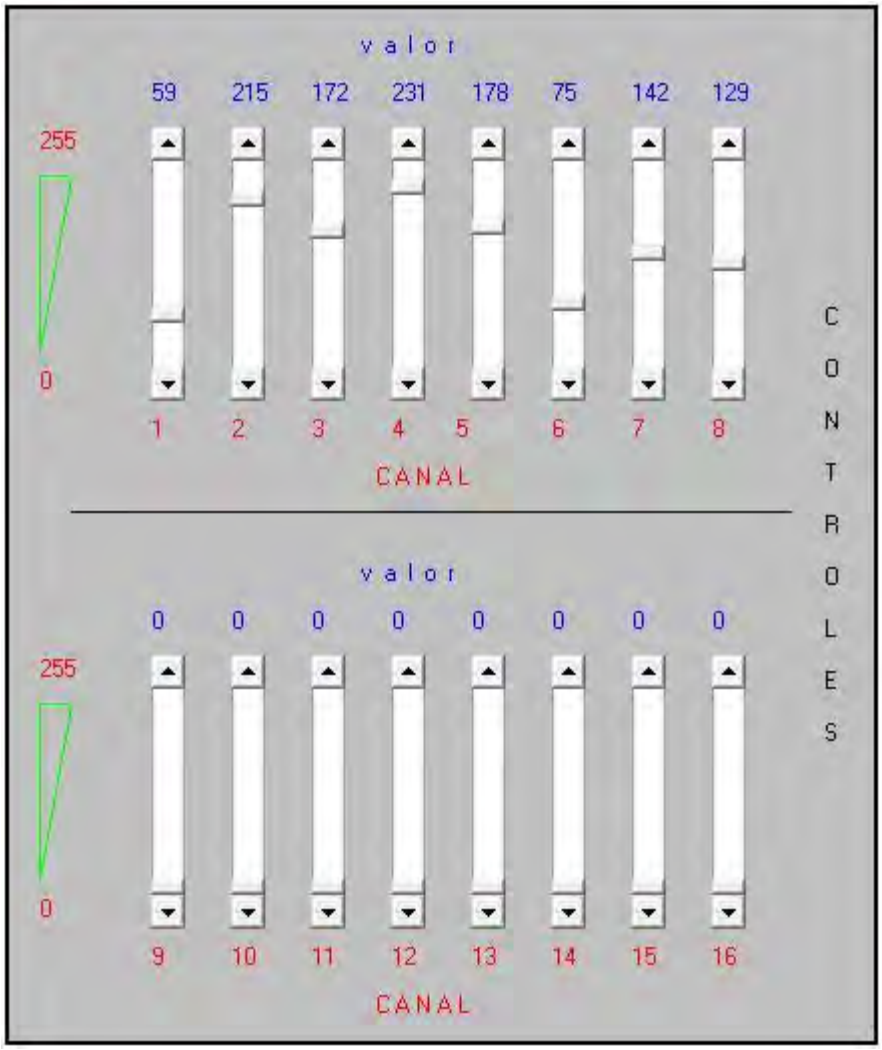


Figura 7.8. Variación de valores

7.1.5 Automático

Terminado la edición de un banco, se puede generar el envío de datos de las escenas pertenecientes al banco de forma automática y con ello se habilitan los controles de tiempo. Para este propósito se emplea un control commandbutton “run”, con etiqueta <RUN>, con el cual se

habilitan los deslizables de tiempo. Este botón alterna su función entre arranque y parada del modo automático, cambiando así la etiqueta <RUN> por <STOP>. Figura 7.9

Figura 7.9. Activación del modo automático

En modo automático tiene preestablecidos valores de tiempo y velocidad en los que se presentan las escenas; sin embargo, pueden ajustarse estos parámetros según se requiera.

Para volver a restablecer los valores por default se emplea el control commandbutton “reset”, con etiqueta <RESET>, y restablece los deslizables de tiempo y velocidad a sus valores de inicio.

7.1.5.1 Variación de Velocidad

El tiempo que tarda en sacar una escena después de otra, se denomina velocidad, el rango que se puede variar, se encuentra entre 1s y 10 min. Esto se logra mediante el control slider “Speed” con el cual podemos variar el tiempo en intervalos de 1s. Figura 7.10.

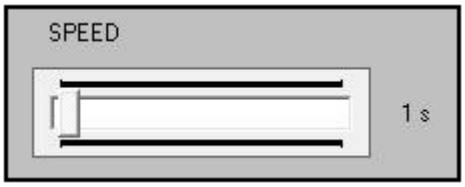


Figura 7.10. Variación de velocidad

7.1.5.2 Variación de Tiempo

Es el tiempo que tarda en volver a presentar el banco, esto es una vez sacadas las ocho escenas que contiene un banco. Figura 7.11.

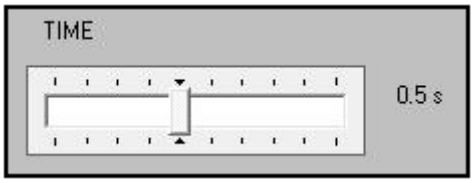


Figura 7.11. Variación de tiempo

El rango en el que varia para volver actualizar el banco se encuentra entre 0.1s y 1s

Además el entorno grafico de usuario cuenta con una barra de estado, en la cual se puede observa las operaciones seleccionadas.

7.1.6 Modo de control

La función del modo de control es permitir el control de las funciones descritas anteriormente en la interfaz grafica por medio del PC o regresar el mando completo al sistema de control (consola) forma manual.

Para llevar acabo esta selección entre PC y MANUAL se emplea el conjunto de controles optionbutton “pcman ( )”, etiquetados como < MODO >, con lo cual únicamente puede estar seleccionado uno solo y no pueden trabajar ambos al mismo tiempo. Figura 7.12.

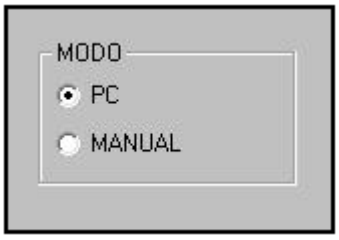


Figura 7.12. Modo de control

CAPITULO 8

8 MANUAL DE OPERACIÓN

El siguiente manual pretende introducir al usuario a conocer las funciones básicas con las que cuenta el sistema de control digital DMX512 (CONSOLA). Se abarcara la explicación de cada una de los botones respecto a la función que realiza dentro del sistema, mensajes desplegados en el LCD y los procedimientos a seguir para la elaboración de escenas móviles de iluminación.

Esto nos ayudara a familiarizarnos en el control de iluminación escenografica, usados en foros de TV, conciertos, lugares de baile y canto, centros nocturnos, eventos al aire libre, etc.

8.1 Sistema Completo

En este capitulo se pretende dar a conocer además de sus funciones básicas de manejo, la forma de conexión entre los distintos elementos que conforman el sistema. Para poder realizar las correctas conexiones y que el funcionamiento sea el óptimo. Figura 8.1.

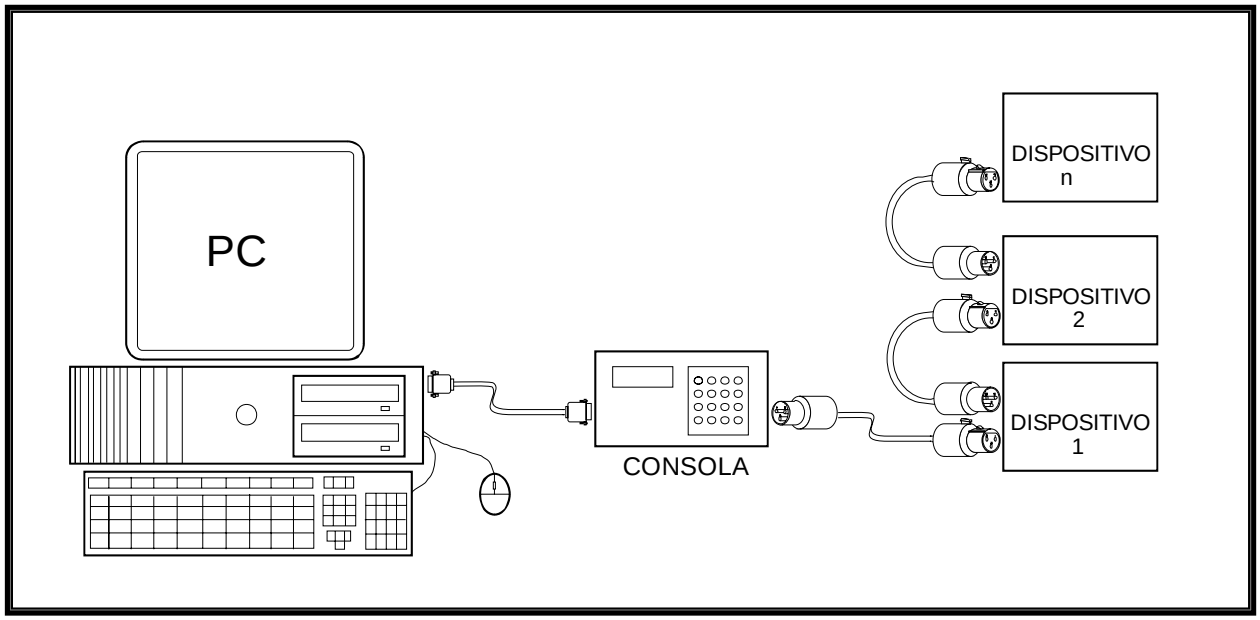


Figura 8.1 Sistema final

8.2 Descripción de Teclado y LCD en Consola

Se describirá la función de cada uno de los botones del teclado y los mensajes desplegados en el LCD.

8.2.1 Funciones del Teclado

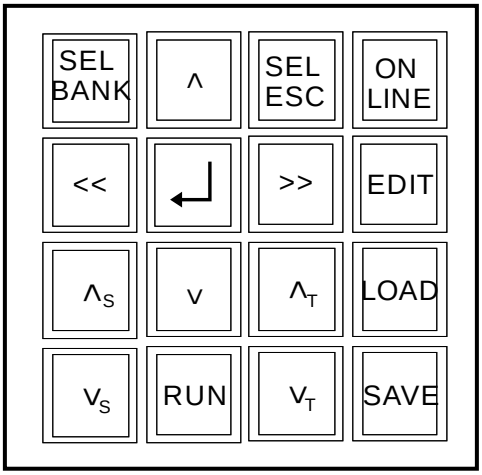


Figura 8.2. Teclado matricial

ONLINE: Botón de encendido, permite seleccionar entre en modo manual o por PC.
SELBANK: Selección de bancos
SELESC: Selección de escenas
<< : Decremento del número de canal
>> : Incremento del número de canal
↵ : Selección de canal y salida
Λ : Incremento del valor del canal seleccionado
V : Decremento del valor del canal seleccionado
Λ_T : Incremento del tiempo en modo RUN
V_T : Decremento del tiempo en modo RUN
Λ_S : Incremento de la velocidad en modo RUN
V_S : Decremento de la velocidad en modo RUN
EDIT: Edita una escena preseleccionada
LOAD: Carga un banco preseleccionado
SAVE: Guarda una escena editada
RUN: Habilita el modo automático

8.2.2 Mensajes en el LCD

MENSAJE DE PRESENTACION: Presenta el nombre del dispositivo y la versión. Permanece en espera de selección de algún modo. Figura 8.3.

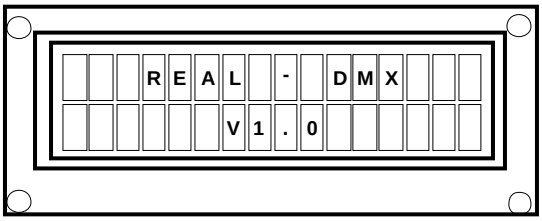


Figura 8.3. LCD en inicio

MENSAJE MODO MANUAL: Visualiza las funciones básicas de operación. En la primera línea presenta el número de canal actual y banco. En la línea 2 se representa el valor del canal seleccionado y la escena a la cual pertenece. Por default, los valores de inicio son: Banco 1, Escena 1, Canal 1 y valor 0. Figura 8.4.

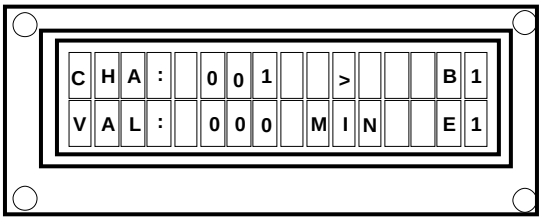


Figura 8.4. LCD en modo manual

MENSAJE MODO ONLINE: Visualiza la leyenda “ONLINE”, en este modo se enlaza con la PC. Figura 8.5.

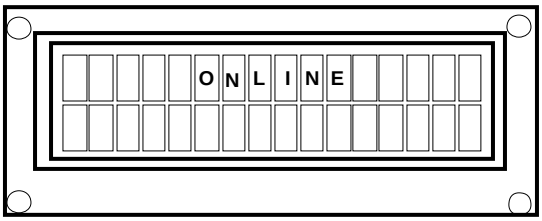


Figura 8.5. LCD en modo online

MENSAJE MODO RUN: Visualiza el tiempo y velocidad. En la primer línea 1 representa la velocidad de refresco de escena y en la línea 2 representa el tiempo de refresco del banco. Tanto la velocidad como el tiempo están expresados en segundos (s). Figura 8.6.

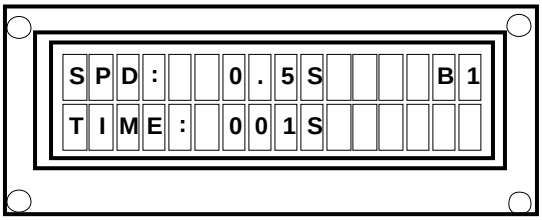


Figura 8.6. LCD en modo run

8.3 Operación Práctica

Para la operación práctica es necesario conocer cada uno de los componentes y la forma de realizar las conexiones entre los elementos, desde la conexión de energía hasta el cableado de los accesorios para poder realizar el ensamble de un espectáculo de iluminación. Los tipos de cables y conectores necesarios han quedado ya especificados en el capitulo 1 y 4.

8.3.1 Conexiones de Alimentación y Control para el Funcionamiento

8.3.1.1 Conexión de Fuente de Alimentación

Para suministrar con una fuente de voltaje a la CONSOLA, se empleara un adaptador de corriente de 120V AC – 5V DC con jack hembra como se muestra en Figura 8.7.

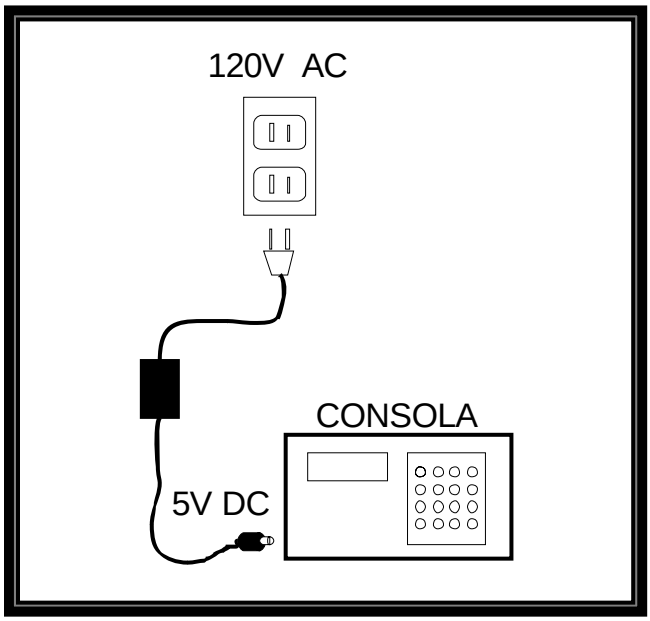


Figura 8.7. Conexión de Fuente de Alimentación

8.3.1.2 Conexión PC-CONSOLA

Se conectara la CONSOLA a través del puerto serial COM de la PC, por medio de cable serial db9 hembra-hembra. Figura 8.8.

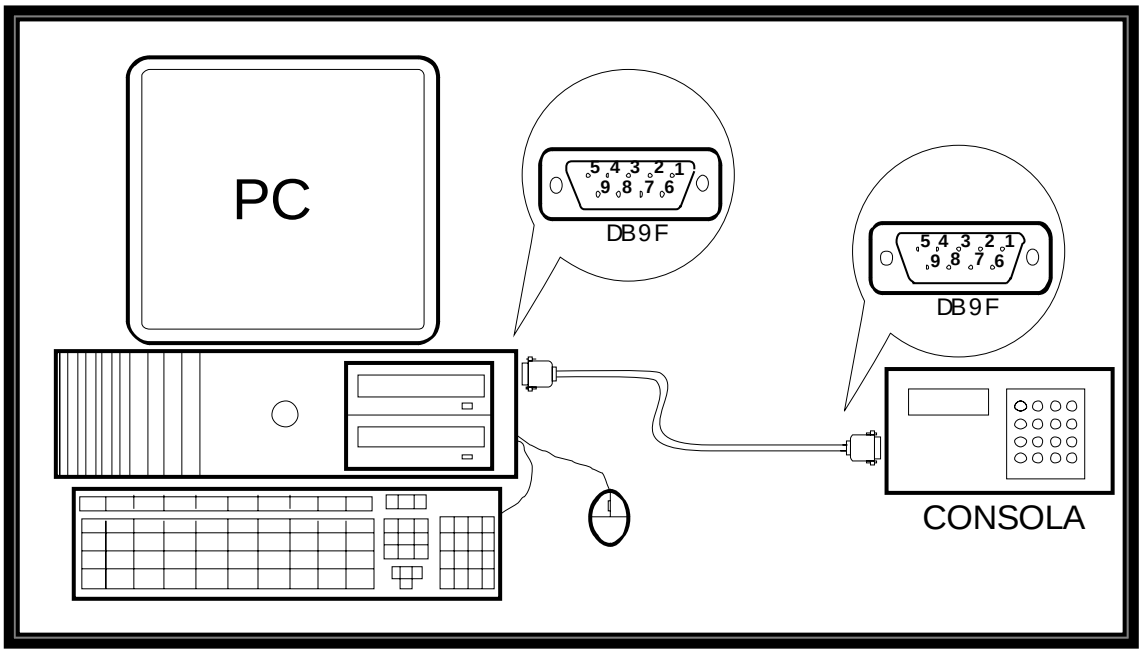


Figura 8.8. Conexión entre PC y Consola

Esta conexión se realiza cuando se requiere hacer el control por medio de la PC, en caso contrario es indistinto el realizar la conexión, ya que no se ve afectado el funcionamiento en modo manual.

8.3.1.3 Conexión CONSOLA-DISPOSITIVOS

La conexión entre la CONSOLA y los dispositivos, se realizara por medio de un cable par trenzado con sus respectivos conectores xlr hembra-macho. Esta conexión se efectúa siempre, no importando el tipo de modalidad que se elija, ya que es imprescindible para el control de los accesorios de iluminación. Figura 8.9.

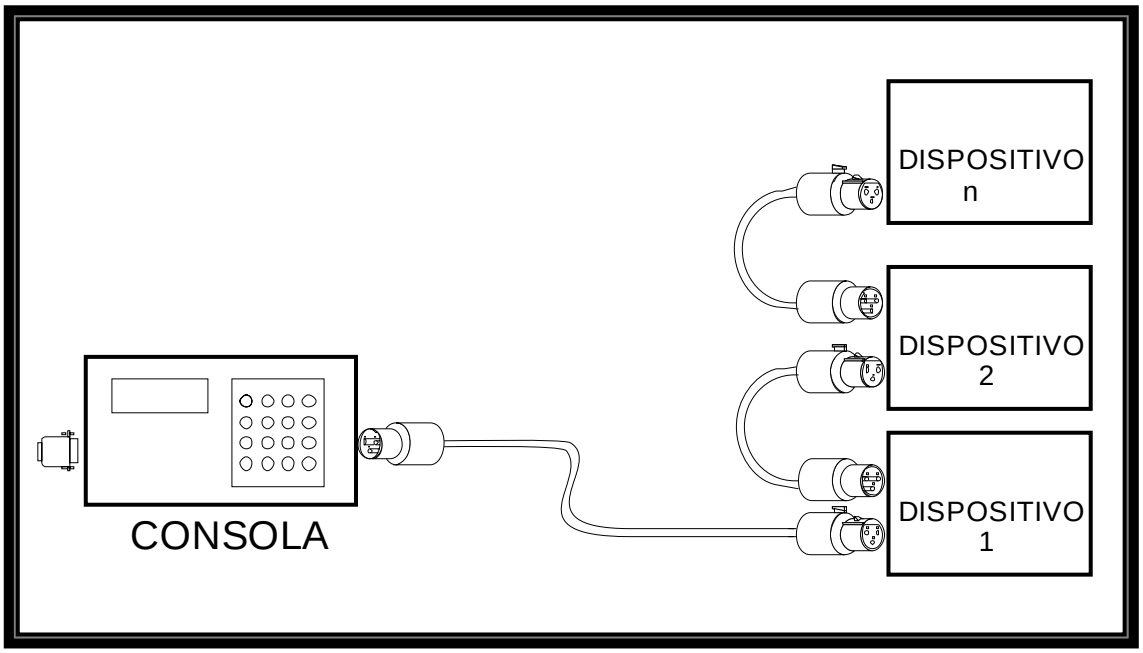


Figura 8.9. Conexión entre Consola y Dispositivos DMX512

8.3.2 Funcionamiento de la CONSOLA

8.3.2.1 Encendido

Una vez conectado a la fuente de alimentación, se presentara la pantalla de bienvenida. Figura 8.10.

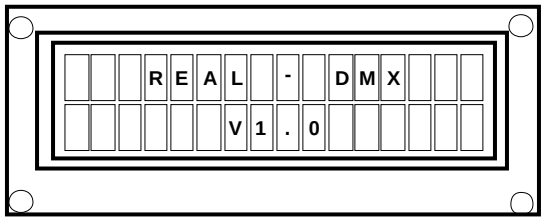


Figura 8.10. Pantalla de inicio

8.3.2.2 Funcionamiento Modo Manual

Para seleccionar el modo de operación se utilizara el botón ON LINE. Presionando una vez ingresa en Modo Manual. Con este modo se realiza el control por medio del teclado. Figura 8.11.

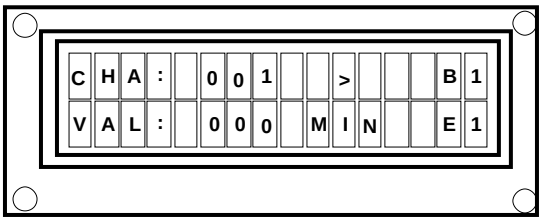


Figura 8.11. Pantalla Modo Manual

8.3.2.2.1 Selección de un Banco

Para seleccionar un banco, se necesita presionar SEL BANK con el cual este incrementa desde banco B1 hasta el B6, este se ubica en esquina superior derecha. Al seleccionar el banco con cual se trabajara, da la oportunidad de modificar cualquiera de sus escenas contenidas. Figura 8.12.

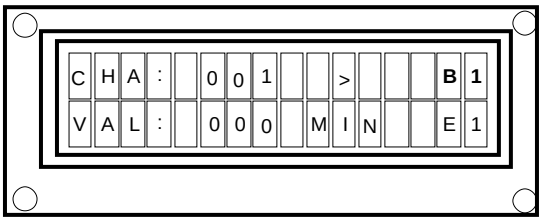


Figura 8.12. Ubicación del banco seleccionado

8.3.2.2.2 Carga de Banco

Una vez seleccionado el banco con el cual se desea trabajar, se usara LOAD para poder cargar en memoria SRAM y ser manipulable sus escenas. Finalizada la carga del banco, aparece a un lado del número de banco la letra “L”, indicando que ha terminado de hacer la carga. Figura 8.13

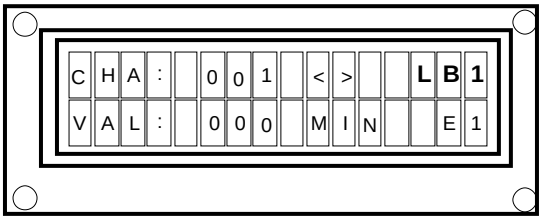


Figura 8.13. Indicación de carga de banco

8.3.2.2.3 Selección de Escena

Para la selección de escena se emplea SEL ESC, y al igual que selección de banco, se conmuta entre las 8 escenas existentes pertenecientes al banco precargado, desde E1 hasta E8. Se encuentra en la esquina inferior derecha. Figura 8.14.

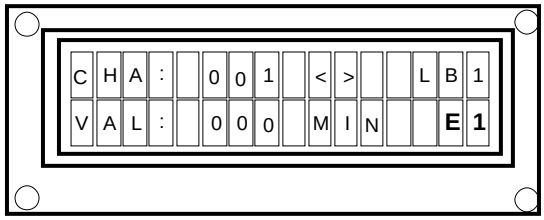


Figura 8.14. Ubicación de escena seleccionada

8.3.2.2.4 Edición de Escena

Para editar una escena se ocupa EDIT, con esto logramos entrar en la escena preselecciona que se desea modificar. Con el fin de visualizar los datos contenidos en la escena, si los hay. Se indica por medio de la letra "e" que los datos visualizados pertenecen a la escena seleccionada. Figura 8.15.

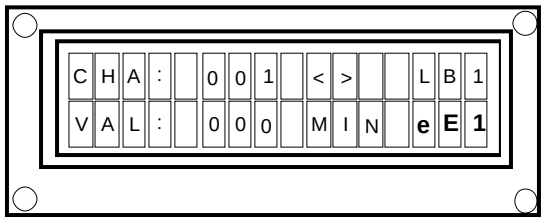


Figura 8.15 Indicación de Edición de escena

8.3.2.2.5 Guardar Escena

Después de editar una escena, se requiere guardar los cambios efectuados. Para esto se emplea SAVE. Se indicara a través de la letra "s" colocada a un lado de la escena que ha terminado de salvar los ajustes realizados. Figura 8.16.

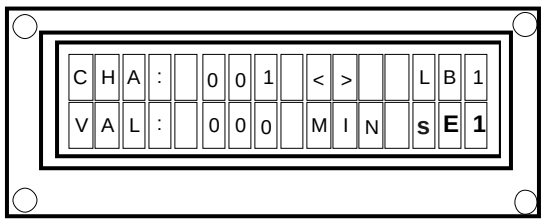


Figura 8.16. Indicación de Escena salvada

8.3.2.2.6 Navegación entre Canales

Para lograr desplazarnos entre los 512 canales se emplean [<<] , [>>] para el decremento e incremento respectivamente. Figura 8.17.

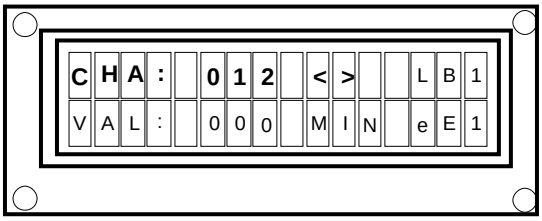


Figura 8.17. Ubicación de Canales DMX512

8.3.2.2.6 Selección de Canal

Se emplea el botón [↵] para entrar en el canal seleccionado y poder modificar el valor contenido. Se presentan "* *" enseguida del canal que indican la selección del mismo. Figura 8.18.

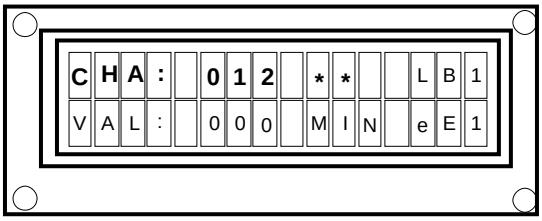


Figura 8.18. Selección de Canal DMX512

Para regresar a la selección de canales se ocupa nuevamente, [↵] una vez que se ha modificado el valor o permanezca igual sin cambios.

8.3.2.2.8 Modificación de Valor

El valor contenido en el canal seleccionado se puede modificar ocupando [Λ], [V] para aumentar y disminuir respectivamente el valor entre 0 – 255 decimal. Figura 8.19.

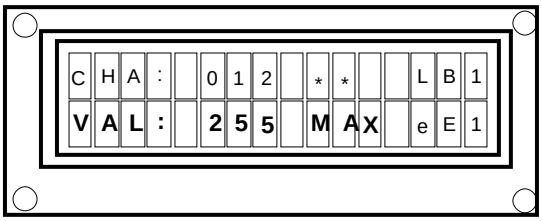


Figura 8.19. Ubicación del valor del Canal DMX512 seleccionado

8.3.2.3 Selección Modo RUN

En esta función se emplea RUN para generar una salida de información de un banco de forma automática, esta salida de información es variable y ajustable en tiempo y velocidad entre escenas. Figura 8.20.

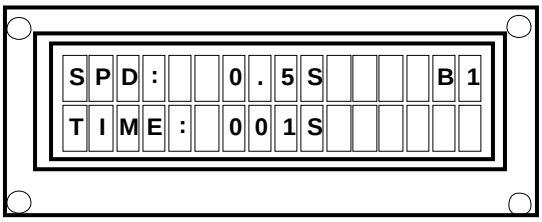


Figura 8.20 Pantalla en Modo Run

8.3.2.3.1 Variación de tiempo

En esta función se varía el tiempo por medio de las teclas Λ_S V_S. Estas sirven para incrementar y decrementar respectivamente, el tiempo que existe entre la repetición de un banco. Figura 8.21.

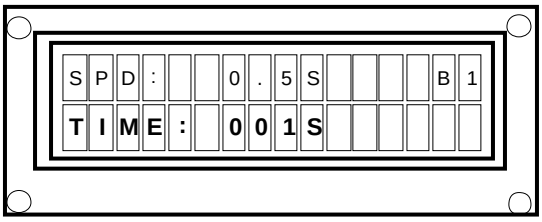


Figura 8.21. Ubicación de tiempo

8.3.2.3.2 Variación de Velocidad

En esta función se varía la velocidad por medio de las teclas Λ_T V_T. Estas sirven para incrementar y decrementar respectivamente, la velocidad con la que las escenas fluyen hacia afuera. Figura 8.22.

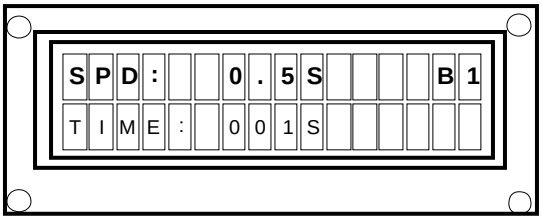


Figura 8.22 Ubicación de velocidad

8.3.2.4 **Modo Online**

Para seleccionar el modo de operación se utilizara ON LINE. Presionando dos veces. En este modo de operación el mando de control se cede a la PC, quedando así controlado a través de la aplicación desarrollada en el capitulo 7. La descripción de funcionamiento queda especificada en dicho capitulo. Recordando que el control se realiza por medio de interfaz grafica del usuario, este es apoyado por el sistema de control DMX512 funcionando como codificador entre la PC y los accesorios a controlar. Figura 8.23.

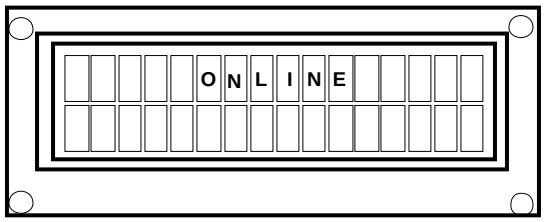


Figura 8.23. Pantalla en Modo Online

CONCLUSIONES

PRUEBAS

Conforme se fue diseñando e implementando el sistema, se realizaron las pruebas en cada etapa que conforman el sistema de control DMX512, como lo son: La respuesta de conexión del LCD y teclado al microcontrolador, en el que se pudo constatar el funcionamiento correcto de los 2 dispositivos de enlace, también se reviso la correcta comunicación entre el microcontrolador y las memorias a través de la introducción y almacenamiento de información en modo manual, se estableció la comunicación entre la PC y el sistema de control enviando un flujo de información por medio de la aplicación realizada en la PC, logrando realizar pruebas de control a dispositivos comerciales que emplean el protocolo DMX512 en forma manual y por PC, tales como: scanner LEADER de la marca ACME, scanner TECNOBEAM 1200 de la marca HIGH END, cabeza móvil EMPEROR de la marca ACME, cabeza móvil SOLO575 de la marca PR, estrobo de la marca HIGH END, PAR 64 controlados por dimmers, etc. Verificando su correcto funcionamiento para el cual fue diseñado.

RESULTADOS

Con las pruebas ya realizadas se establece que el sistema de control DMX512 implementado es reconocido por distintos dispositivos y accesorios que cuentan con el protocolo DMX512 cubriendo así con los requerimientos que establece la norma.

Este sistema es totalmente funcional, cuenta con las funciones básicas de un control DMX512 comercial. Aunque se puede adicionar mejoras en la programación del microcontrolador, para poder ser comercializado, ya que se tuvo la limitante en cuanto a la herramienta de desarrollo empleada para la programación del microcontrolador, por ser una versión de prueba.

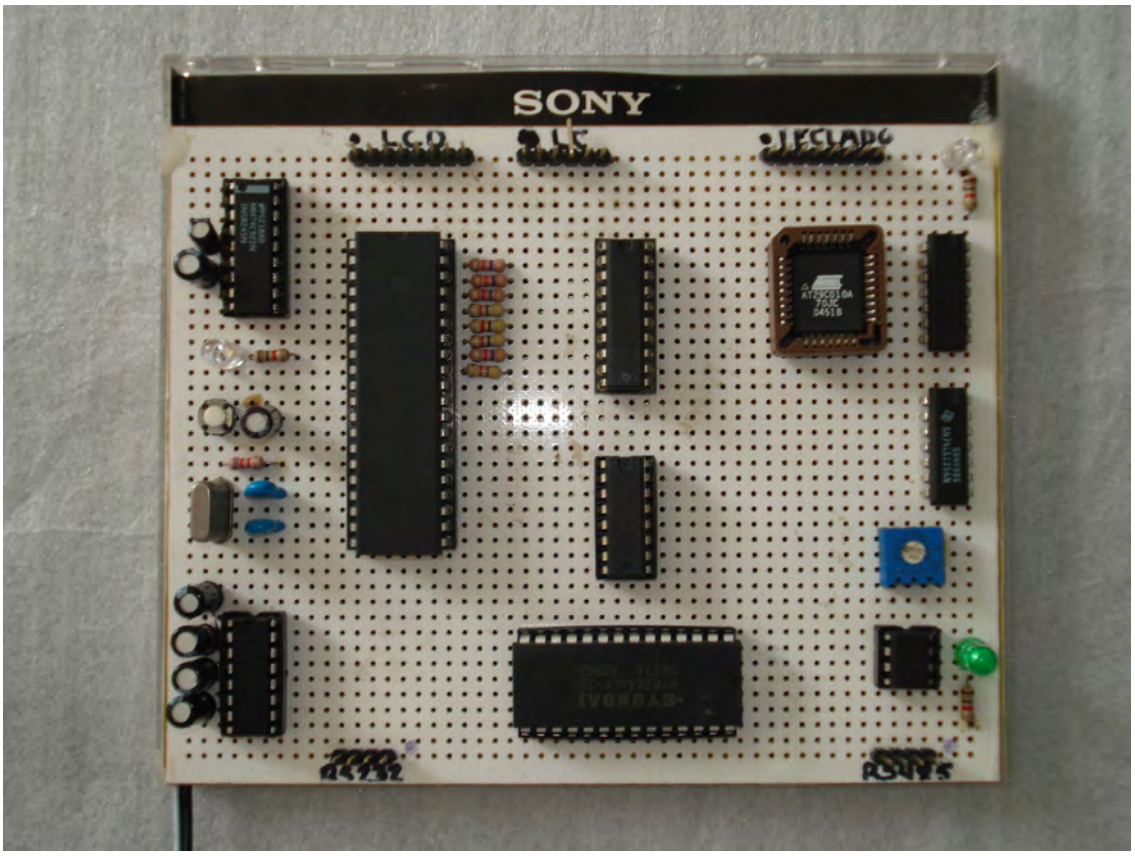
CONCLUSIONES

La consola DMX512 diseñada e implementada para el control de iluminación y accesorios DMX512 es útil para la realización de escenas de iluminación, en discotecas, bares y centro de espectáculos, etc... A través del control por medio de un PC, con una aplicación que simula una consola análoga, o si no se cuenta con una, se realiza el control manualmente, ya que este dispositivo cuenta con un teclado y un display que ayuda al usuario a interactuar con el control como lo haría con una consola comercial.

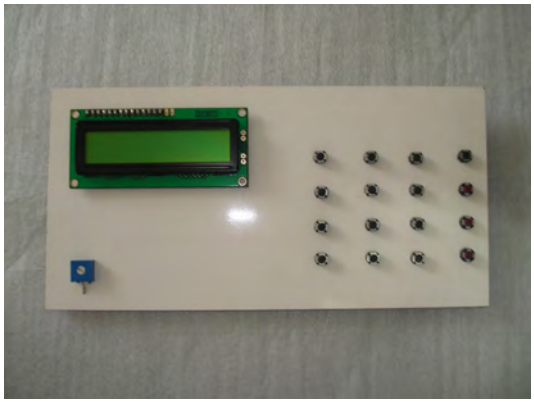
Este proyecto propone algo nuevo, ya que soluciona el problema de contar con una consola análoga para controlar manualmente los equipos y tener una interfaz para lograr el control por

medio de un PC. Ya que repercute en la inversión de comprar dos dispositivos cuando nuestro proyecto cubre ambas en un solo dispositivo.

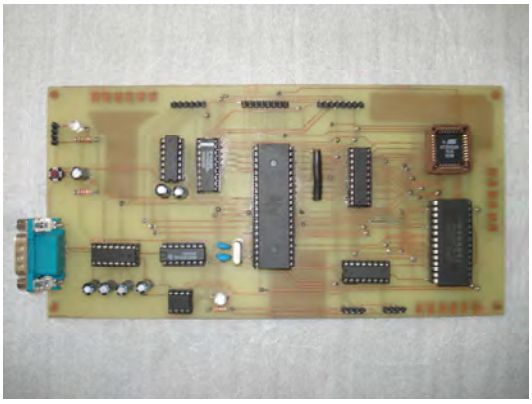
A continuación se muestra los elementos físicos que componen el proyecto ya finalizado y con el cual se realizaron las pruebas. Con esto concluimos el objetivo del proyecto.



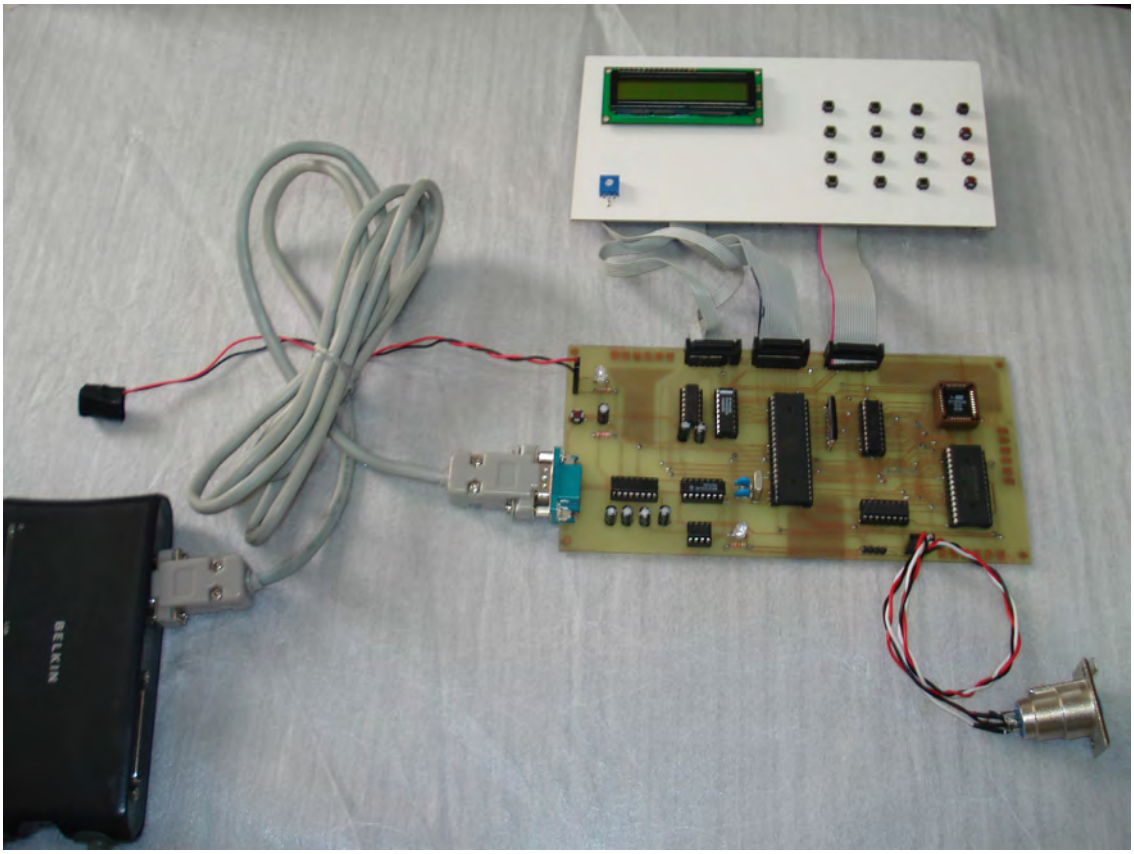
Circuito de pruebas, alambrado con wire-wrap



Teclado y Display



Tarjeta impresa del proyecto, consola DMX512



Conexión del proyecto

Conector XLR tipo cannon

Cable RS-232, Interfaz entre consola y pc

Cable par trenzado para DMX512
Interfaz entre consola y dispositivos

BIBLIOGRAFIA

BIBLIOGRAFIA

FRANCISCO JAVIER CEBALLOS, CURSO DE PROGRAMACION DE VISUAL BASIC, EDIT. ALFAOMEGA MEXICO D.F

FREDRICK M. CADY, MICROCONTROLLERS AND MICROCOMPUTERS, OXFORD UNIVERSITY PRESS, INC. NY 1997

HAN-WAY HUANG, USING MCS-51 MICROCONTROLLERS, OXFORD UNIVERSITY PRESS, INC. NY 2000

TEXAS INSTRUMENTS, INTERFACE CIRCUITS, TEXAS INSTRUMENTS, DALLAS 1991

WAYNE TOMASI, SISTEMAS DE COMUNICACIONES ELECTRÓNICAS, PEARSON EDUCACIÓN MEXICO 1996.

TOCCI, RONALD J. Y WIDMER, NEAL S. SISTEMAS DIGITALES, PEARSON EDUCACIÓN, MEXICO 2003.

http://es.wikipedia.org

http://www.bairesrobotics.com.ar

http://www.monacor-spain.net

http://www.tecnoprofile.com

http://www.usitt.org

http://www.esta.org

http://www.taringa.net

APENDICE A

Features

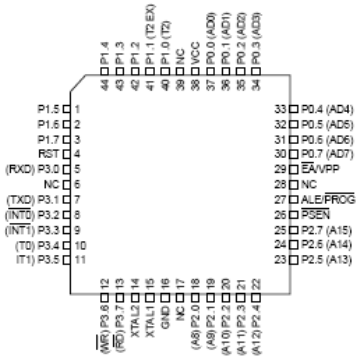
- Compatible with MCS-51™ Products
- 8K Bytes of In-System Reprogrammable Flash Memory
- Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Eight Interrupt Sources
- Programmable Serial Channel
- Low-power Idle and Power-down Modes

Description

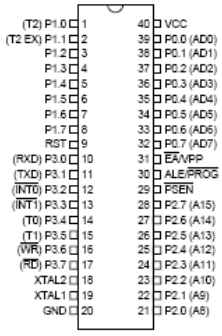
The AT89C52 is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 and 80C52 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C52 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

Pin Configurations

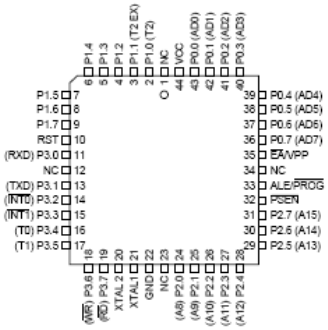
PQFP/TQFP



PDIP



PLCC



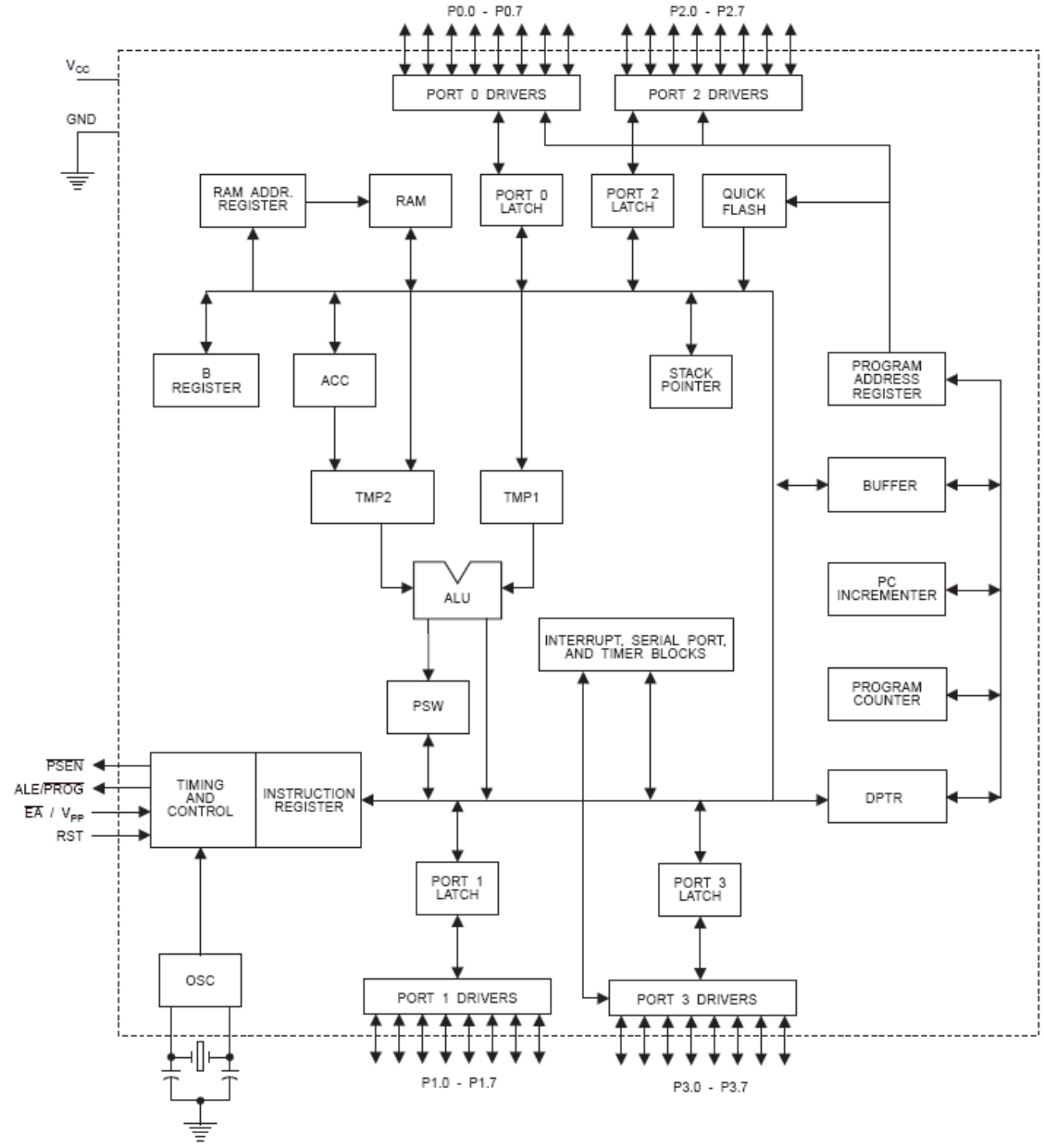
8-bit Microcontroller with 8K Bytes Flash

AT89C52

Not Recommended for New Designs. Use AT89S52.

Rev. 0313H–02/00

Block Diagram



The AT89C52 provides the following standard features: 8K bytes of Flash, 256 bytes of RAM, 32 I/O lines, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full-duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89C52 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power-down mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next hardware reset.

Pin Description

VCC
Supply voltage.

GND
Ground.

Port 0

Port 0 is an 8-bit open drain bi-directional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1

Port 1 is an 8-bit bi-directional I/O port with internal pullups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively, as shown in the following table.
Port 1 also receives the low-order address bytes during Flash programming and verification.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)

Port 2
Port 2 is an 8-bit bi-directional I/O port with internal pullups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3
Port 3 is an 8-bit bi-directional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C51, as shown in the following table.

Port 3 also receives some control signals for Flash programming and verification.

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST
Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$
Address Latch Enable is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external

timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

$\overline{\text{PSEN}}$

Program Store Enable is the read strobe to external program memory.

When the AT89C52 is executing code from external program memory, $\overline{\text{PSEN}}$ is activated twice each machine cycle, except that two $\overline{\text{PSEN}}$ activations are skipped during each access to external data memory.

$\overline{\text{EA}}$/VPP

External Access Enable. $\overline{\text{EA}}$ must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{\text{EA}}$ will be internally latched on reset.

$\overline{\text{EA}}$ should be strapped to V_{CC} for internal program executions.

This pin also receives the 12-volt programming enable voltage (V_{PP}) during Flash programming when 12-volt programming is selected.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier.

Table 1. AT89C52 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000								0AFH
0A0H	P2 11111111								0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H	P0 11111111	SP 00000111	DPL 00000000	DPH 00000000				PCON 0XXX0000	87H

Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in Table 1.

Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

User software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

Timer 2 Registers Control and status bits are contained in registers T2CON (shown in Table 2) and T2MOD (shown in Table 4) for Timer 2. The register pair (RCAP2H, RCAP2L) are the Capture/Reload registers for Timer 2 in 16-bit capture mode or 16-bit auto-reload mode.

Interrupt Registers The individual interrupt enable bits are in the IE register. Two priorities can be set for each of the six interrupt sources in the IP register.r

Table 2. T2CON – Timer/Counter 2 Control Registe

T2CON Address = 0C8H — Reset Value = 0000 0000B

Bit Addressable

Bit	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$
	7	6	5	4	3	2	1	0

Symbol	Function
TF2	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
RCLK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	Timer 2 external enable. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
C/$\overline{T2}$	Timer or counter select for Timer 2. C/$\overline{T2}$ = 0 for timer function. C/$\overline{T2}$ = 1 for external event counter (falling edge triggered).
CP/$\overline{RL2}$	Capture/Reload select. CP/$\overline{RL2}$ = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/$\overline{RL2}$ = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

Data Memory

The AT89C52 implements 256 bytes of on-chip RAM. The upper 128 bytes occupy a parallel address space to the Special Function Registers. That means the upper 128 bytes have the same addresses as the SFR space but are physically separate from SFR space.

When an instruction accesses an internal location above address 7FH, the address mode used in the instruction specifies whether the CPU accesses the upper 128 bytes of RAM or the SFR space. Instructions that use direct addressing access SFR space.

For example, the following direct addressing instruction accesses the SFR at location 0A0H (which is P2).

```
MOV 0A0H, #data
```

Instructions that use indirect addressing access the upper 128 bytes of RAM. For example, the following indirect addressing instruction, where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H).

```
MOV @R0, #data
```

Note that stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space.

Timer 0 and 1

Timer 0 and Timer 1 in the AT89C52 operate the same way as Timer 0 and Timer 1 in the AT89C51.

Timer 2

Timer 2 is a 16-bit Timer/Counter that can operate as either a timer or an event counter. The type of operation is selected by bit C/$\overline{T2}$ in the SFR T2CON (shown in Table 2). Timer 2 has three operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON, as shown in Table 3.

Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

Table 3. Timer 2 Operating Modes

RCLK +TCLK	CP/$\overline{RL2}$	TR2	MODE
0	0	1	16-bit Auto-reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(Off)

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since two machine cycles (24 oscillator periods) are required to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. To ensure that a given level is sampled at least once before it changes, the level should be held for at least one full machine cycle.

Capture Mode

In the capture mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 is a 16-bit timer or counter which upon overflow sets bit TF2 in T2CON. This bit can then be used to generate an interrupt. If EXEN2 = 1, Timer 2 performs the same operation, but a 1-to-0 transition at external input T2EX also causes the current value in TH2 and TL2 to be captured into RCAP2H and RCAP2L, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set. The EXF2 bit, like TF2, can generate an interrupt. The capture mode is illustrated in Figure 1.

Auto-reload (Up or Down Counter)

Timer 2 can be programmed to count up or down when configured in its 16-bit auto-reload mode. This feature is invoked by the DCEN (Down Counter Enable) bit located in the SFR T2MOD (see Table 4). Upon reset, the DCEN bit is set to 0 so that timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down, depending on the value of the T2EX pin.

Figure 1. Timer in Capture Mode

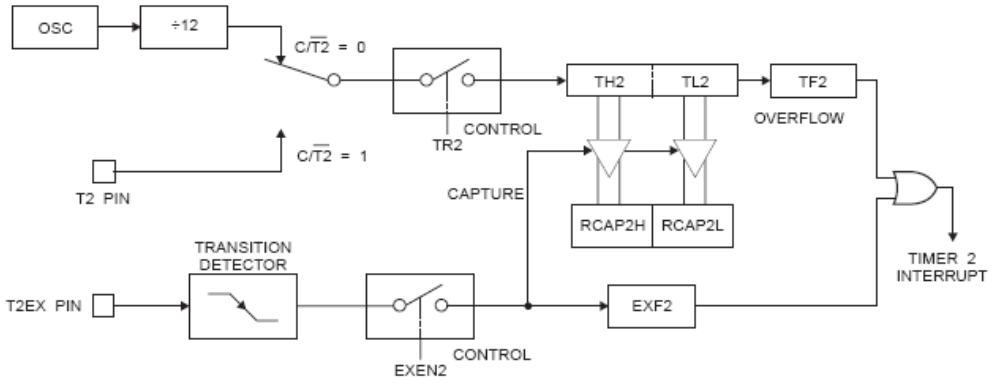


Figure 2 shows Timer 2 automatically counting up when DCEN = 0. In this mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H and RCAP2L. The values in Timer in Capture ModeRCAP2H and RCAP2L are preset by software. If EXEN2 = 1, a 16-bit reload can be triggered either by an overflow or by a 1-to-0 transition at external input T2EX. This transition also sets the EXF2 bit. Both the TF2 and EXF2 bits can generate an interrupt if enabled.

Setting the DCEN bit enables Timer 2 to count up or down, as shown in Figure 3. In this mode, the T2EX pin controls the direction of the count. A logic 1 at T2EX makes Timer 2 count up. The timer will overflow at 0FFFFH and set the TF2 bit. This overflow also causes the 16-bit value in RCAP2H and RCAP2L to be reloaded into the timer registers, TH2 and TL2, respectively.

A logic 0 at T2EX makes Timer 2 count down. The timer underflows when TH2 and TL2 equal the values stored in RCAP2H and RCAP2L. The underflow sets the TF2 bit and causes 0FFFFH to be reloaded into the timer registers.

The EXF2 bit toggles whenever Timer 2 overflows or underflows and can be used as a 17th bit of resolution. In this operating mode, EXF2 does not flag an interrupt.

Figure 2. Timer 2 Auto Reload Mode (DCEN = 0)

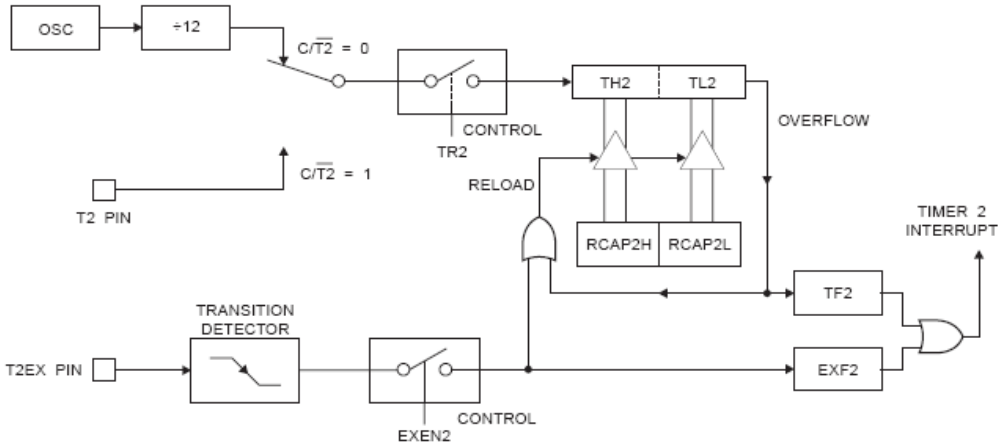


Table 4. T2MOD – Timer 2 Mode Control Register

T2MOD Address = 0C9H — Reset Value = XXXX XX00B

Not Bit Addressable

	–	–	–	–	–	–	T2OE	DCEN
Bit	7	6	5	4	3	2	1	0

Symbol	Function
–	Not implemented, reserved for future
T2OE	Timer 2 Output Enable bit.
DCEN	When set, this bit allows Timer 2 to be configured as an up/down counter.

Figure 3. Timer 2 Auto Reload Mode (DCEN = 1)

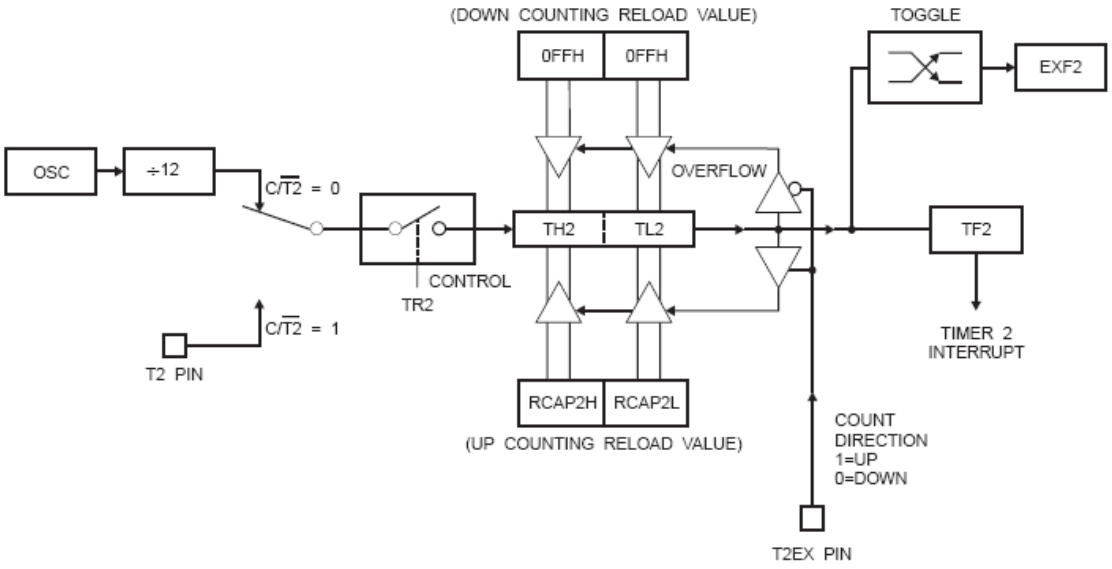
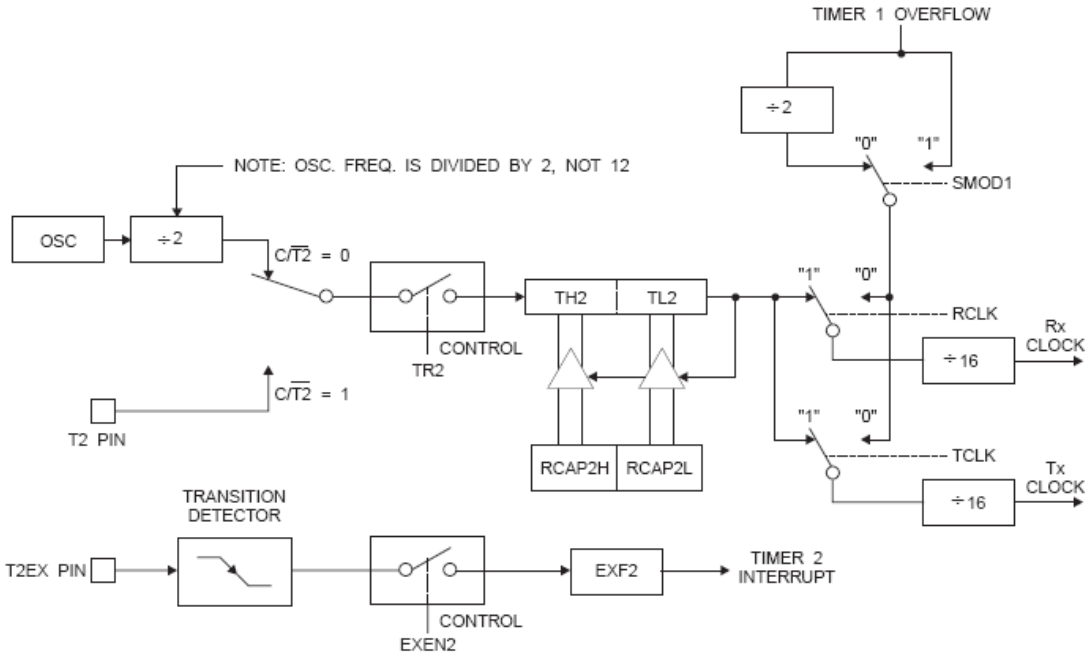


Figure 4. Timer 2 in Baud Rate Generator Mode



Baud Rate Generator

Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Table 2). Note that the baud rates for transmit and receive can be different if Timer 2 is used for the receiver or transmitter and Timer 1 is used for the other function. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 4.

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

The baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate according to the following equation.

$$\text{Modes 1 and 3 Baud Rates} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The Timer can be configured for either timer or counter operation. In most applications, it is configured for timer operation (CP/$\overline{T2}$ = 0). The timer operation is different for Timer 2 when it is used as a baud rate generator. Normally, as a timer, it increments every machine cycle (at 1/12 the oscillator frequency). As a baud rate generator, however, it increments every state time (at 1/2 the oscillator frequency). The baud rate formula is given below.

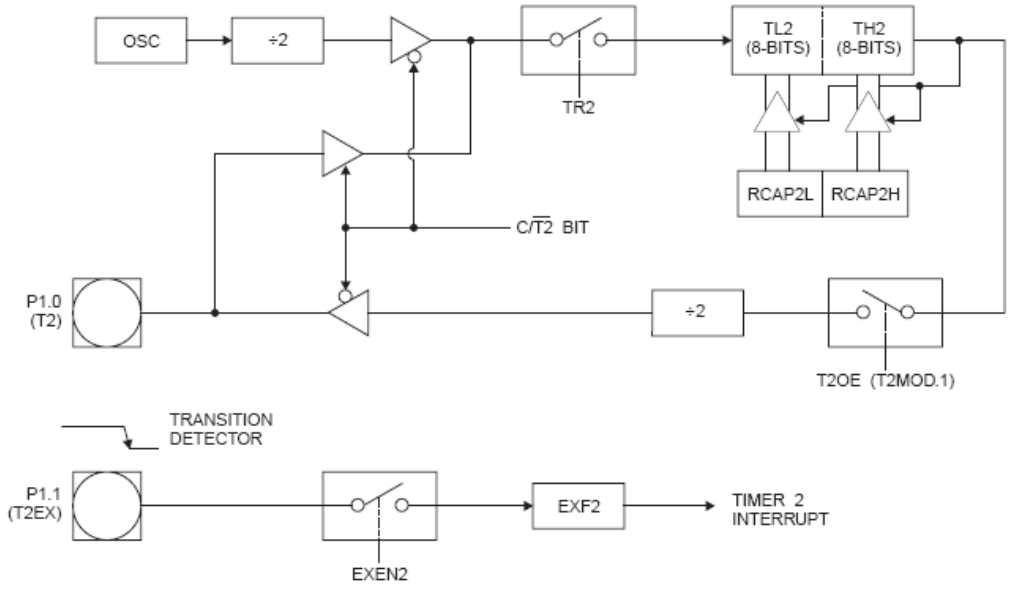
$$\frac{\text{Modes 1 and 3}}{\text{Baud Rate}} = \frac{\text{Oscillator Frequency}}{32 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as a baud rate generator is shown in Figure 4. This figure is valid only if RCLK or TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2 and will not generate an interrupt. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt.

Note that when Timer 2 is running (TR2 = 1) as a timer in the baud rate generator mode, TH2 or TL2 should not be read from or written to. Under these conditions, the Timer is incremented every state time, and the results of a read or write may not be accurate. The RCAP2 registers may be read but should not be written to, because a write might overlap a reload and cause write and/or reload errors. The timer should be turned off (clear TR2) before accessing the Timer 2 or RCAP2 registers.

Figure 5. Timer 2 in Clock-out Mode



Programmable Clock Out

A 50% duty cycle clock can be programmed to come out on P1.0, as shown in Figure 5. This pin, besides being a regular I/O pin, has two alternate functions. It can be programmed to input the external clock for Timer/Counter 2 or to output a 50% duty cycle clock ranging from 61 Hz to 4 MHz at a 16 MHz operating frequency.

To configure the Timer/Counter 2 as a clock generator, bit C/$\overline{T2}$ (T2CON.1) must be cleared and bit T2OE (T2MOD.1) must be set. Bit TR2 (T2CON.2) starts and stops the timer.

The clock-out frequency depends on the oscillator frequency and the reload value of Timer 2 capture registers (RCAP2H, RCAP2L), as shown in the following equation.

$$\text{Clock-Out Frequency} = \frac{\text{Oscillator Fequency}}{4 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

In the clock-out mode, Timer 2 roll-overs will not generate an interrupt. This behavior is similar to when Timer 2 is used as a baud-rate generator. It is possible to use Timer 2 as a baud-rate generator and a clock generator simultaneously. Note, however, that the baud-rate and clock-out frequencies cannot be determined independently from one another since they both use RCAP2H and RCAP2L.

UART

The UART in the AT89C52 operates the same way as the UART in the AT89C51.

Interrupts

The AT89C52 has a total of six interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (Timers 0, 1, and 2), and the serial port interrupt. These interrupts are all shown in Figure 6.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE. IE also contains a global disable bit, EA, which disables all interrupts at once.

Note that Table shows that bit position IE.6 is unimplemented. In the AT89C51, bit position IE.5 is also unimplemented. User software should not write 1s to these bit positions, since they may be used in future AT89 products.

Timer 2 interrupt is generated by the logical OR of bits TF2 and EXF2 in register T2CON. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and that bit will have to be cleared in software.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag, TF2, is set at S2P2 and is polled in the same cycle in which the timer overflows.

Table 5. Interrupt Enable (IE) Register

(MSB) (LSB)

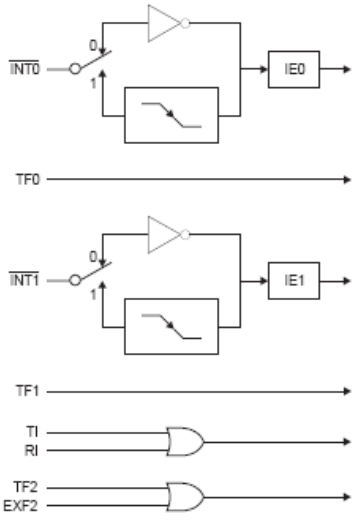
EA	–	ET2	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved.
ET2	IE.5	Timer 2 interrupt enable bit.
ES	IE.4	Serial Port interrupt enable bit.
ET1	IE.3	Timer 1 interrupt enable bit.
EX1	IE.2	External interrupt 1 enable bit.
ET0	IE.1	Timer 0 interrupt enable bit.
EX0	IE.0	External interrupt 0 enable bit.
User software should never write 1s to unimplemented bits, because they may be used in future AT89 products.		

Figure 6. Interrupt Sources



Oscillator Characteristics

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 7. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven, as shown in Figure 8. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Idle Mode

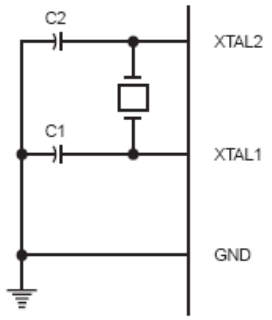
In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

Note that when idle mode is terminated by a hardware reset, the device normally resumes program execution from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when idle mode is terminated by a reset, the instruction following the one that invokes idle mode should not write to a port pin or to external memory.

Power-down Mode

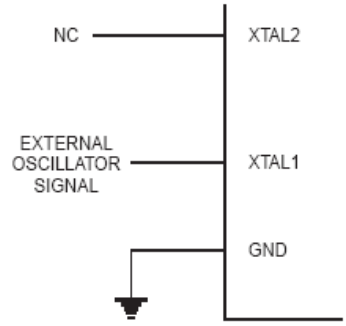
In the power-down mode, the oscillator is stopped, and the instruction that invokes power-down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power-down mode is terminated. The only exit from power-down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

Figure 7. Oscillator Connections



Note: C1, C2 = 30 pF ±10 pF for Crystals
= 40 pF ±10 pF for Ceramic Resonators

Figure 8. External Clock Drive Configuration



Status of External Pins During Idle and Powe-down Modes

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

1.11 Instruction Set Summary

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS				
ADD	A,R_n	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@R_i	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,R_n	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@R_i	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#data	Add immediate data to Acc with Carry	2	12
SUBB	A,R_n	Subtract Register from Acc with borrow	1	12
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB	A,@R_i	Subtract indirect RAM from ACC with borrow	1	12
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	R_n	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@R_i	Increment direct RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	R_n	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@R_i	Decrement indirect RAM	1	12
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12

Note: 1. All mnemonics copyrighted © Intel Corp., 1980.

Mnemonic		Description	Byte	Oscillator Period
LOGICAL OPERATIONS				
ANL	A,R_n	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@R_i	AND indirect RAM to Accumulator	1	12

Mnemonic		Description	Byte	Oscillator Period
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,R_n	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@R_i	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,R_n	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@R_i	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
LOGICAL OPERATIONS (continued)				
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,R_n	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@R_i	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	R_n,A	Move Accumulator to register	1	12
MOV	R_n,direct	Move direct byte to register	2	24
MOV	R_n,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,R_n	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@R_i	Move indirect RAM to direct byte	2	24

The 8051 Instruction Set

Mnemonic		Description	Byte	Oscillator Period
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@R_i,A	Move Accumulator to indirect RAM	1	12
MOV	@R_i,direct	Move direct byte to indirect RAM	2	24
MOV	@R_i,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@R_i	Move External RAM (8-bit addr) to Acc	1	24
DATA TRANSFER (continued)				
MOVX	A,@DPTR	Move Exernal RAM (16-bit addr) to Acc	1	24
MOVX	@R_i,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,R_n	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@R_i	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@R_i	Exchange low-order Digit indirect RAM with Acc	1	12
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24

Mnemonic		Description	Byte	Oscillator Period
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24
CJNE	R_n,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@R_i,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	R_n,rel	Decrement register and Jump if Not Zero	2	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

1.12 Instructions That Affect Flag Settings

Table 1-13. Instructions that affect Flag Settings

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	O		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	O	X		ANL C,/bit	X		
DIV	O	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

Note: Operations on SFR byte address 208 or bit addresses 209-215 (that is, the PSW or bits in the PSW) also affect flag settings.

AT29C010A

Features

- Fast Read Access Time - 70 ns
- 5-Volt-Only Reprogramming
- Sector Program Operation
  - Single Cycle Reprogram (Erase and Program)
  - 1024 Sectors (128 bytes/sector)
  - Internal Address and Data Latches for 128-Bytes
- Two 8 KB Boot Blocks with Lockout
- Internal Program Control and Timer
- Hardware and Software Data Protection
- Fast Sector Program Cycle Time - 10 ms
- DATA Polling for End of Program Detection
- Low Power Dissipation
  - 50 mA Active Current
  - 100 µA CMOS Standby Current
- Typical Endurance > 10,000 Cycles
- Single 5V ±10% Supply
- CMOS and TTL Compatible Inputs and Outputs
- Commercial and Industrial Temperature Ranges

1 Megabit (128K x 8) 5-volt Only CMOS Flash Memory

Description

The AT29C010A is a 5-volt-only in-system Flash programmable and erasable read only memory (PEROM). Its 1 megabit of memory is organized as 131,072 words by 8 bits. Manufactured with Atmel's advanced nonvolatile CMOS technology, the device offers access times to 70 ns with power dissipation of just 275 mW over the commercial temperature range. When the device is deselected, the CMOS standby current is less than 100 µA. The device endurance is such that any sector can typically be written to in excess of 10,000 times.

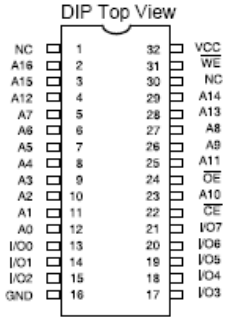
To allow for simple in-system reprogrammability, the AT29C010A does not require high input voltages for programming. Five-volt-only commands determine the opera-

(continued)

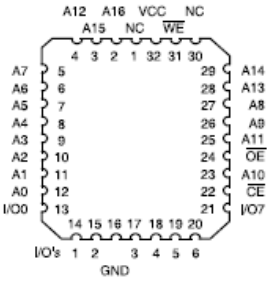
Pin Configurations

Pin Name	Function
A0 - A16	Addresses
$\overline{CE}$	Chip Enable
$\overline{OE}$	Output Enable
$\overline{WE}$	Write Enable
I/O0 - I/O7	Data Inputs/Outputs
NC	No Connect

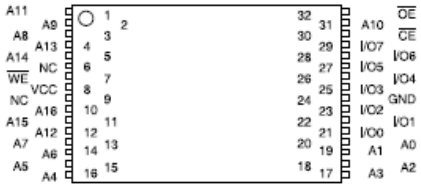
DIP Top View



PLCC Top View



TSOP Top View
Type 1



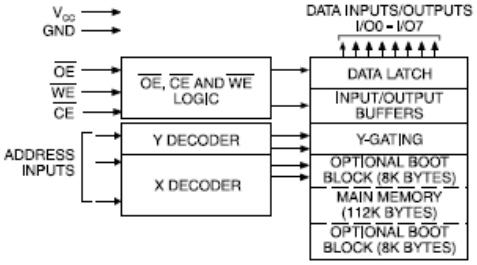
0394B

Description (Continued)

tion of the device. Reading data out of the device is similar to reading from an EPROM. Reprogramming the AT29C010A is performed on a sector basis; 128-bytes of data are loaded into the device and then simultaneously programmed.

During a reprogram cycle, the address locations and 128-bytes of data are internally latched, freeing the address and data bus for other operations. Following the initiation of a program cycle, the device will automatically erase the sector and then program the latched data using an internal control timer. The end of a program cycle can be detected by DATA polling of I/O7. Once the end of a program cycle has been detected, a new access for a read or program can begin.

Block Diagram



Device Operation

READ: The AT29C010A is accessed like an EPROM. When $\overline{CE}$ and $\overline{OE}$ are low and $\overline{WE}$ is high, the data stored at the memory location determined by the address pins is asserted on the outputs. The outputs are put in the high impedance state whenever $\overline{CE}$ or $\overline{OE}$ is high. This dual-line control gives designers flexibility in preventing bus contention.

BYTE LOAD: Byte loads are used to enter the 128-bytes of a sector to be programmed or the software codes for data protection. A byte load is performed by applying a low pulse on the $\overline{WE}$ or $\overline{CE}$ input with $\overline{CE}$ or $\overline{WE}$ low (respectively) and $\overline{OE}$ high. The address is latched on the falling edge of $\overline{CE}$ or $\overline{WE}$, whichever occurs last. The data is latched by the first rising edge of $\overline{CE}$ or $\overline{WE}$.

PROGRAM: The device is reprogrammed on a sector basis. If a byte of data within a sector is to be changed, data for the entire sector must be loaded into the device. The data in any byte that is not loaded during the programming of its sector will be indeterminate. Once the bytes of a sector are loaded into the device, they are simultaneously programmed during the internal programming period. After the first data byte has been loaded into the device, successive bytes are entered in the same manner. Each new byte to be programmed must have its high to low transition on $\overline{WE}$ (or $\overline{CE}$) within 150 µs of the low to high transition of $\overline{WE}$ (or $\overline{CE}$) of the preceding byte. If a high to low transition is not detected within 150 µs of the last low to high transition, the load period will end and the internal programming period will start. A7 to A16 specify the sector address. The sector address must be valid during each high to low transition of $\overline{WE}$ (or $\overline{CE}$). A0 to A6 specify the byte address within the sector. The bytes may be loaded in any order; sequential loading is not required. Once a programming operation has been initiated, and for the duration of t_{WC}, a read operation will effectively be a polling operation.

SOFTWARE DATA PROTECTION: A software controlled data protection feature is available on the AT29C010A. Once the software protection is enabled a software algorithm must be issued to the device before a program may be performed. The software protection feature may be enabled or disabled by the user; when shipped from Atmel, the software data protection feature is disabled. To enable the software data protection, a series of three program commands to specific addresses with specific data must be performed. After the software data protection is enabled the same three program commands must begin each program cycle in order for the programs to occur. All software program commands must obey the sector program timing specifications. Once set, the software data protection feature remains active unless its disable command is issued. Power transitions will not reset the software data protection feature, however the software feature will guard against inadvertent program cycles during power transitions.

Once set, software data protection will remain active unless the disable command sequence is issued.

After setting SDP, any attempt to write to the device without the 3-byte command sequence will start the internal write timers. No data will be written to the device; however, for the duration of t_{WC}, a read operation will effectively be a polling operation.

(continued)

Device Operation (Continued)

After the software data protection's 3-byte command code is given, a byte load is performed by applying a low pulse on the $\overline{WE}$ or $\overline{CE}$ input with $\overline{CE}$ or $\overline{WE}$ low (respectively) and $\overline{OE}$ high. The address is latched on the falling edge of $\overline{CE}$ or $\overline{WE}$, whichever occurs last. The data is latched by the first rising edge of $\overline{CE}$ or $\overline{WE}$. The 128-bytes of data must be loaded into each sector by the same procedure as outlined in the program section under device operation.

HARDWARE DATA PROTECTION: Hardware features protect against inadvertent programs to the AT29C010A in the following ways: (a) V_{CC} sense— if V_{CC} is below 3.8V (typical), the program function is inhibited. (b) V_{CC} power on delay— once V_{CC} has reached the V_{CC} sense level, the device will automatically time out 5 ms (typical) before programming. (c) Program inhibit— holding any one of $\overline{OE}$ low, $\overline{CE}$ high or $\overline{WE}$ high inhibits program cycles. (d) Noise filter— pulses of less than 15 ns (typical) on the $\overline{WE}$ or $\overline{CE}$ inputs will not initiate a program cycle.

PRODUCT IDENTIFICATION: The product identification mode identifies the device and manufacturer as Atmel. It may be accessed by hardware or software operation. The hardware operation mode can be used by an external programmer to identify the correct programming algorithm for the Atmel product. In addition, users may wish to use the software product identification mode to identify the part (i.e. using the device code), and have the system software use the appropriate sector size for program operations. In this manner, the user can have a common board design for 256K to 4-megabit densities and, with each density's sector size in a memory map, have the system software apply the appropriate sector size.

For details, see Operating Modes (for hardware operation) or Software Product Identification. The manufacturer and device code is the same for both modes.

$\overline{DATA}$ POLLING: The AT29C010A features $\overline{DATA}$ polling to indicate the end of a program cycle. During a program cycle an attempted read of the last byte loaded will result in the complement of the loaded data on I/O7. Once the program cycle has been completed, true data is valid on all outputs and the next cycle may begin. DATA polling may begin at any time during the program cycle.

TOGGLE BIT: In addition to $\overline{DATA}$ polling the AT29C010A provides another method for determining the end of a program or erase cycle. During a program or erase operation, successive attempts to read data from the device will result in I/O6 toggling between one and zero. Once the program cycle has completed, I/O6 will stop toggling and valid data will be read. Examining the toggle bit may begin at any time during a program cycle.

OPTIONAL CHIP ERASE MODE: The entire device can be erased by using a 6-byte software code. Please see Software Chip Erase application note for details.

BOOT BLOCK PROGRAMMING LOCKOUT: The AT29C010A has two designated memory blocks that have a programming lockout feature. This feature prevents programming of data in the designated block once the feature has been enabled. Each of these blocks consists of 8K bytes; the programming lockout feature can be set independently for either block. While the lockout feature does not have to be activated, it can be activated for either or both blocks.

These two 8K memory sections are referred to as *boot blocks*. Secure code which will bring up a system can be contained in a boot block. The AT29C010A blocks are located in the first 8K bytes of memory and the last 8K bytes of memory. The boot block programming lockout feature can therefore support systems that boot from the lower addresses of memory or the higher addresses. Once the programming lockout feature has been activated, the data in that block can no longer be erased or programmed; data in other memory locations can still be changed through the regular programming methods. To activate the lockout feature, a series of seven program commands to specific addresses with specific data must be performed. Please see Boot Block Lockout Feature Enable Algorithm.

If the boot block lockout feature has been activated on either block, the chip erase function will be disabled.

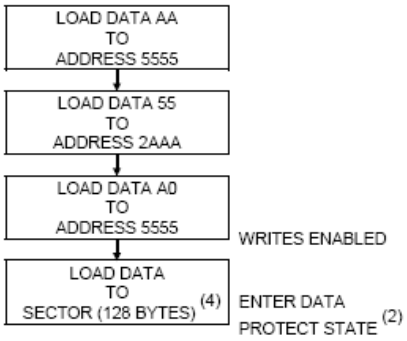
(continued)

Absolute Maximum Ratings*

Temperature Under Bias	-55°C to +125°C
Storage Temperature	-65°C to +150°C
All Input Voltages (including NC Pins) with Respect to Ground	-0.6V to +6.25V
All Output Voltages with Respect to Ground	-0.6V to V_{CC} + 0.6V
Voltage on $\overline{OE}$ with Respect to Ground	-0.6V to +13.5V

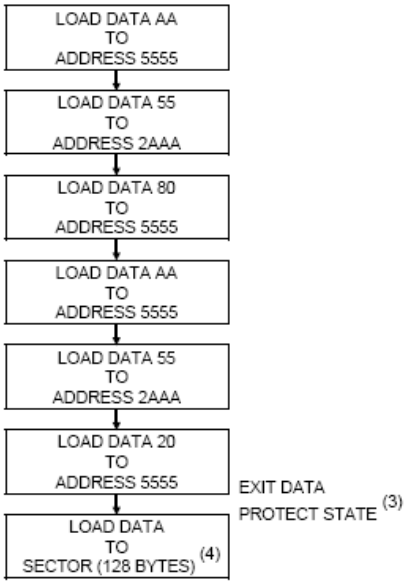
*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Software Data Protection Enable Algorithm (1)

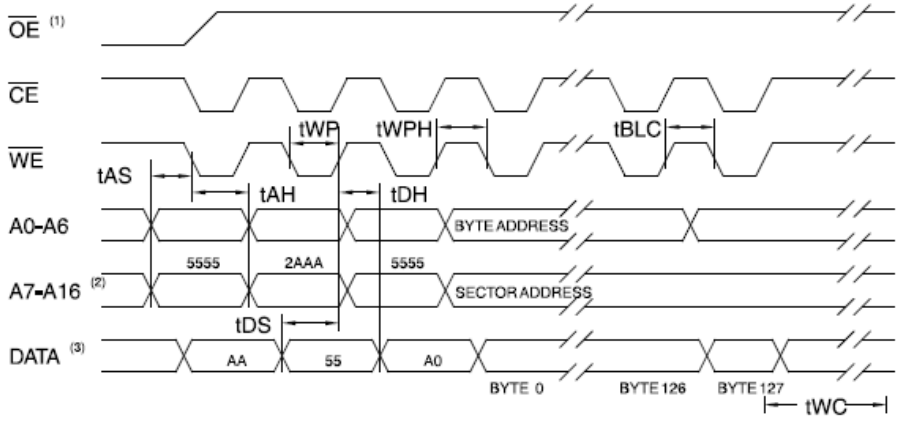


Notes for software program code:
1. Data Format: I/O7 - I/O0 (Hex);
   Address Format: A14 - A0 (Hex).
2. Data Protect state will be activated at end of program cycle.
3. Data Protect state will be deactivated at end of program period.
4. 128-bytes of data MUST BE loaded.

Software Data Protection Disable Algorithm (1)



Software Protected Program Cycle Waveform (1, 2, 3)



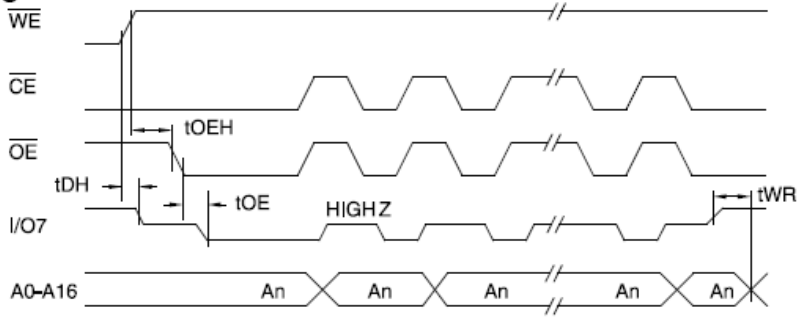
Notes: 1. A7 through A16 must specify the sector address during each high to low transition of $\overline{WE}$ (or $\overline{CE}$) after the software code has been entered.
2. $\overline{OE}$ must be high when $\overline{WE}$ and $\overline{CE}$ are both low.
3. **All bytes that are not loaded within the sector being programmed will be indeterminate.**

$\overline{\text{Data}}$ Polling Characteristics (1)

Symbol	Parameter	Min	Typ	Max	Units
t_{DH}	Data Hold Time	10			ns
t_{OEH}	$\overline{OE}$ Hold Time	10			ns
t_{OE}	$\overline{OE}$ to Output Delay (2)				ns
t_{WR}	Write Recovery Time	0			ns

Notes: 1. These parameters are characterized and not 100% tested.
2. See t_{OE} spec in AC Read Characteristics.

$\overline{\text{Data}}$ Polling Waveforms

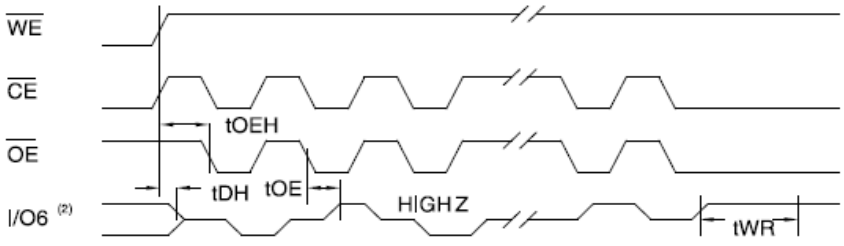


Toggle Bit Characteristics (1)

Symbol	Parameter	Min	Typ	Max	Units
t_{DH}	Data Hold Time	10			ns
t_{OEH}	$\overline{OE}$ Hold Time	10			ns
t_{OE}	$\overline{OE}$ to Output Delay (2)				ns
t_{OEHP}	$\overline{OE}$ High Pulse	150			ns
t_{WR}	Write Recovery Time	0			ns

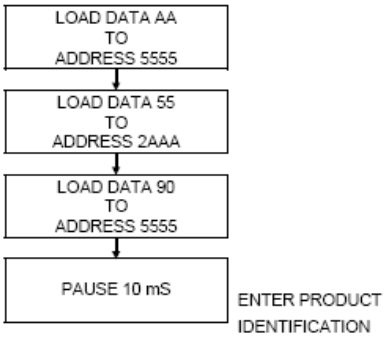
Notes: 1. These parameters are characterized and not 100% tested.
2. See t_{OE} spec in AC Read Characteristics.

Toggle Bit Waveforms (1, 2, 3)

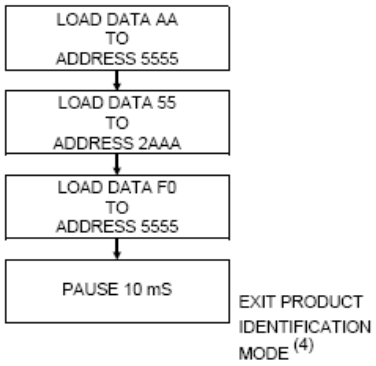


Notes: 1. Toggling either $\overline{OE}$ or $\overline{CE}$ or both $\overline{OE}$ and $\overline{CE}$ will operate toggle bit.
2. Beginning and ending state of I/O6 will vary.
3. Any address location may be used but the address should not vary.

Software Product Identification Entry (1)

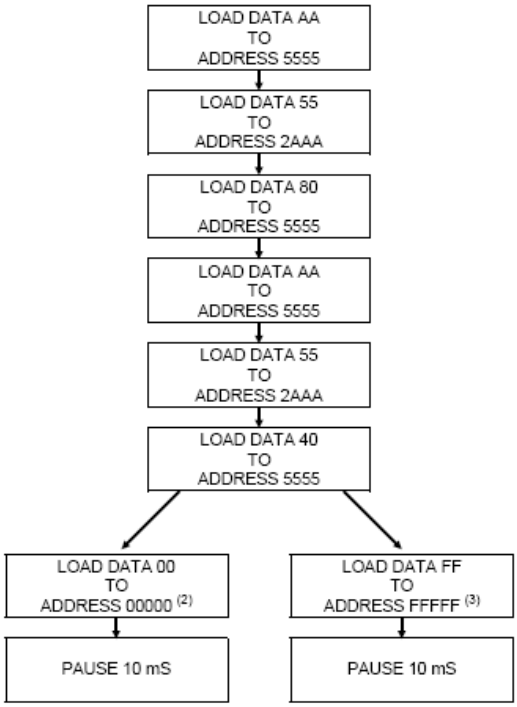


Software Product Identification Exit (1)



Notes for software product identification:
1. Data Format: I/O7 - I/O0 (Hex);
   Address Format: A14 - A0 (Hex).
2. A1 - A16 = V_{IL}.
   Manufacture Code is read for A0 = V_{IL};
   Device Code is read for A0 = V_{IH}.
3. The device does not remain in identification mode if powered down.
4. The device returns to standard operation mode.
5. Manufacturer Code: 1F
   Device Code: D5

Boot Block Lockout Feature Enable Algorithm (1)



Notes for boot block lockout feature enable:
1. Data Format: I/O7 - I/O0 (Hex);
   Address Format: A14 - A0 (Hex).
2. Lockout feature set on lower address boot block.
3. Lockout feature set on higher address boot block.

HYUNDAI

HY6264A-(I) Series
8Kx8bit CMOS SRAM

DESCRIPTION

The HY6264A/ HY6264A-I is a high-speed, low power and 8,192x8-bits CMOS static RAM fabricated using Hyundai's high performance twin tub CMOS process technology. This high reliability process coupled with innovative circuit design techniques, yields maximum access time of 70ns. The HY6264A/HY6264A-I has a data retention mode that guarantees data to remain valid at the minimum power supply voltage of 2.0 volt. Using the CMOS technology, supply voltage from 2.0 to 5.5 volt has little effect on supply current in the data retention mode. Reducing the supply voltage to minimize current drain is unnecessary for the HY6264A/ HY6264A-I Series.

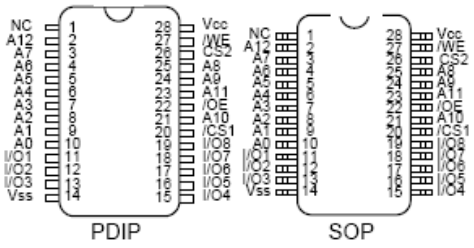
FEATURES

- Fully static operation and Tri-state outputs
- TTL compatible inputs and outputs
- Low power consumption
- Battery backup(L/LL-part)
  -2.0V(min.) data retention
- Standard pin configuration
  -28 pin 600 mil PDIP
  -28 pin 330 mil SOP

Product No.	Voltage (V)	Speed (ns)	Operation Current(mA)	Standby Current(uA)			Temperature (••)
					L	LL	
HY6264A	5.0	70/85/100	50	1mA	100	10	0~70(Normal)
HY6264A-I	5.0	70/85/100	50	-	100	30	-40~85(E.T.)

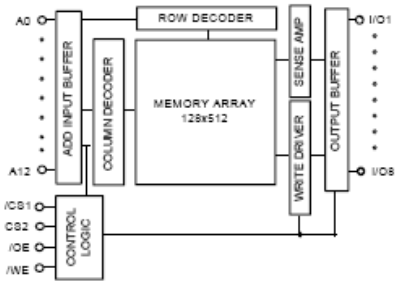
Note 1. E.T. : Extended Temperature, Normal : Normal Temperature
2. Current value is max.

PIN CONNECTION



PDIP SOP

BLOCK DIAGRAM



PIN DESCRIPTION

Pin Name	Pin Function	Pin Name	Pin Function
/CS1	Chip Select 1	I/O1-I/O8	Data Input/Output
CS2	Chip Select 2	Vcc	Power(+5V)
/WE	Write Enable	Vss	Ground
/OE	Output Enable	NC	No Connect
	Address Inputs		

This document is a general product description and is subject to change without notice. Hyundai Electronics does not assume any responsibility for use of circuits described. No patent licenses are implied.
Rev.01 /Jul.96 Hyundai Semiconductor

ORDERING INFORMATION

PART NO.	SPEED	POWER	TEMP.	PACKAGE
HY6264AP	70/85/100			PDIP
HY6264ALP	70/85/100	L-part		PDIP
HY6264ALLP	70/85/100	LL-part		PDIP
HY6264AJ	70/85/100			SOP
HY6264ALJ	70/85/100	L-part		SOP
HY6264ALLJ	70/85/100	LL-part		SOP
HY6264ALP-I	70/85/100	L-part	E.T.	PDIP
HY6264ALLP-I	70/85/100	LL-part	E.T.	PDIP
HY6264ALJ-I	70/85/100	L-part	E.T.	SOP
HY6264ALLJ-I	70/85/100	LL-part	E.T.	SOP

ABSOLUTE MAXIMUM RATING (1)

Symbol	Parameter	Rating	Unit	Remark
V_{CC}, V_{IN}, V_{OUT}	Power Supply, Input/Output Voltage	-0.5 to 7.0	V	
T_A	Operating Temperature	0 to 70	••	HY6264A
		-40 to 85	••	HY6264A-I
T_{STG}	Storage Temperature	-65 to 125	••	
P_D	Power Dissipation	1.0	W	
I_{OUT}	Data Output Current	50	mA	
T_{SOLDER}	Lead Soldering Temperature & Time	260•10	••• sec	

Note
1. Stresses greater than those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent damage to the device. This is stress rating only and the functional operation of the device under these or any other conditions above those indicated in the operation of this specification is not implied. Exposure to the absolute maximum rating conditions for an extended period may affect reliability.

RECOMMENDED DC OPERATING CONDITIONS

T_A=0°C TO 70°C /-40°C TO 85°C

Symbol	Parameter	Min.	Typ.	Max.	Unit
V_{CC}	Supply Voltage	4.5	5.0	5.5	V
V_{IH}	Input High Voltage	2.2	-	Vcc+0.5	V
V_{IL}	Input Low Voltage	-0.5(1)	-	0.8	V

Note
1.V_{IL} = -3.0V for pulse width less than 50ns

TRUTH TABLE

/CS1	CS2	/WE	/OE	MODE	I/O OPERATION
H	X	X	X	Standby	High-Z
X	L	X	X		High-Z
L	H	H	H	Output Disabled	High-Z
L	H	H	L	Read	Data Out
L	H	L	X	Write	Data In

Note
1. H=V_{IH}, L=V_{IL}, X=Don't Care

DC ELECTRICAL CHARACTERISTICS

Vcc = 5.0V•• 10%, TA = 0•• to 70••(Normal)/ -40•• to 85••(E.T.) unless otherwise specified

Symbol	Parameter		Test Condition		Min	Typ	Max	Unit
ILI	Input Leakage Current		Vss •• VIN •• Vcc		-1	-	1	uA
ILO	Output Leakage Current		Vss •• VOUT •• Vcc /CS1=VIH or CS2=VIL or /OE = VIH or/ WE = VIL		-1	-	1	uA
Icc	Operating Power Supply Current		/CS1 = VIL, CS2=VIH, VIN = VIH or VIL, II/O = 0mA		-	30	50	mA
ICC1	Average Operating Current		/CS1 = VIL, CS2=VIH Min. Duty Cycle = 100%, II/O = 0mA		-	30	50	mA
ISB	TTL Standby Current (TTL Input)		/CS1 = VIH or CS2=VIL		-	0.4	2	mA
ISB1	CMOS Standby Current (CMOS Input)	HY6264A	/CS1 •• Vcc - 0.2V, CS2 •• 0.2V,or CS2 •• Vcc-0.2V		-	-	1	mA
				L	-	2	100	uA
				LL	-	1	10	uA
		HY6264A-I		L	-	2	100	uA
				LL	-	1	30	uA
VOL	Output Low Voltage		IOL = 2.1mA		-	-	0.4	V
VOH	Output High Voltage		IOH = -1.0mA		2.4	-	-	V

Note : Typical values are at Vcc = 5.0V, TA = 25••

AC CHARACTERISTICS

Vcc = 5.0V•• 10%, TA = 0•• to 70••(Normal) / -40•• to 85••(E.T.), unless otherwise noted

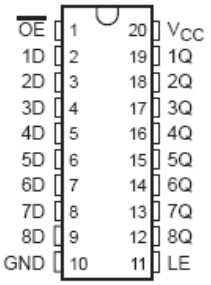
#	Symbol	Parameter	-70		-85		-10		Unit
			Min	Max	Min	Max	Min	Max	
	READ CYCLE								
1	tRC	Read Cycle Time	70	-	85	-	100	-	ns
2	tAA	Address Access Time	-	70	-	85	-	100	ns
3	tACS	Chip Select Access Time	-	70	-	85	-	100	ns
4	tOE	Output Enable to Output Valid	-	45	-	50	-	55	ns
5	tCLZ	Chip Select to Output in Low Z	10	-	10	-	10	-	ns
6	tOLZ	Output Enable to Outputin Low Z	5	-	5	-	5	-	ns
7	tCHZ	Chip Deselection to Output in High Z	0	30	0	35	0	35	ns
8	tOHZ	Out Disable to Output in High Z	0	30	0	35	0	35	ns
9	tOH	Output Hold from Address Change	5	-	5	-	10	-	ns
	WRITE CYCLE								
10	tWC	Write Cycle Time	70	-	85	-	100	-	ns
11	tCW	Chip Selection to End of Write	55	-	60	-	70	-	ns
12	tAW	Address Valid to End of Write	55	-	60	-	70	-	ns
13	tAS	Address Set-up Time	0	-	0	-	0	-	ns
14	tWP	Write Pulse Width	50	-	55	-	60	-	ns
15	tWR	Write Recovery Time	0	-	0	-	0	-	ns
16	tWHZ	Write to Output in High Z	0	30	0	35	0	35	ns
17	tDW	Data to Write Time Overlap	35	-	35	-	40	-	ns
18	tDH	Data Hold from Write Time	0	-	0	-	0	-	ns
19	tOW	Output Active from End of Write	5	-	5	-	5	-	ns

SN54HC573A, SN74HC573A
OCTAL TRANSPARENT D-TYPE LATCHES WITH 3-STATE OUTPUTS

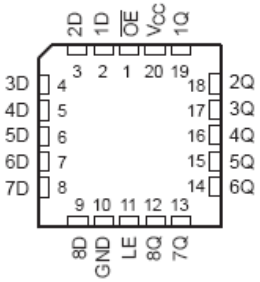
SCLS147E – DECEMBER 1982 – REVISED SEPTEMBER 2003

- Wide Operating Voltage Range of 2 V to 6 V
- High-Current 3-State Outputs Drive Bus Lines Directly or Up To 15 LSTTL Loads
- Low Power Consumption, 80-μA Max I_{CC}
- Typical t_{pd} = 21 ns
- ±6-mA Output Drive at 5 V
- Low Input Current of 1 μA Max
- Bus-Structured Pinout

SN54HC573A . . . J OR W PACKAGE
SN74HC573A . . . DB, DW, N, OR PW PACKAGE
(TOP VIEW)



SN54HC573A . . . FK PACKAGE
(TOP VIEW)



description/ordering information

These octal transparent D-type latches feature 3-state outputs designed specifically for driving highly capacitive or relatively low-impedance loads. They are particularly suitable for implementing buffer registers, I/O ports, bidirectional bus drivers, and working registers.

While the latch-enable (LE) input is high, the Q outputs respond to the data (D) inputs. When LE is low, the outputs are latched to retain the data that was set up.

A buffered output-enable ($\overline{OE}$) input can be used to place the eight outputs in either a normal logic state (high or low logic levels) or the high-impedance state. In the high-impedance state, the outputs neither load nor drive the bus lines significantly. The high-impedance state and increased drive provide the capability to drive bus lines without interface or pullup components.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
−40°C to 85°C	PDIP – N	Tube of 25	SN74HC573AN	SN74HC573AN
	SOIC – DW	Tube of 40	SN74HC573ADW	HC573A
		Reel of 2500	SN74HC573ADWR	
	SSOP – DB	Reel of 2000	SN74HC573ADBR	HC573A
	TSSOP – PW	Reel of 2000	SN74HC573APWR	HC573A
		Reel of 250	SN74HC573APWT	
−55°C to 125°C	CDIP – J	Tube of 25	SNJ54HC573AJ	SNJ54HC573AJ
	CFP – W	Tube of 150	SNJ54HC573AW	SNJ54HC573AW
	LCCC – FK	Tube of 55	SNJ54HC573AFK	SNJ54HC573AFK

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 2003, Texas Instruments Incorporated
On products compliant to MIL-PRF-38535, all parameters are tested unless otherwise noted. On all other products, production processing does not necessarily include testing of all parameters.

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

description/ordering information (continued)

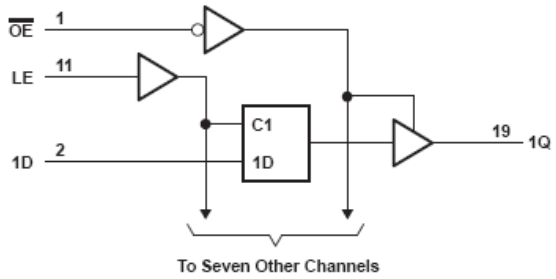
To ensure the high-impedance state during power up or power down, $\overline{OE}$ should be tied to V_{CC} through a pullup resistor; the minimum value of the resistor is determined by the current-sinking capability of the driver.

$\overline{OE}$ does not affect the internal operations of the latches. Old data can be retained or new data can be entered while the outputs are in the high-impedance state.

FUNCTION TABLE
(each latch)

INPUTS			OUTPUT
$\overline{OE}$	LE	D	Q
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

logic diagram (positive logic)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage range, V_{CC} . . . . . −0.5 V to 7 V
Input clamp current, I_{IK} ($V_I < 0$ or $V_I > V_{CC}$) (see Note 1) . . . . . ±20 mA
Output clamp current, I_{OK} ($V_O < 0$ or $V_O > V_{CC}$) (see Note 1) . . . . . ±20 mA
Continuous output current, I_O (V_O = 0 to V_{CC}) . . . . . ±35 mA
Continuous current through V_{CC} or GND . . . . . ±70 mA
Package thermal impedance, θ_{JA} (see Note 2): DB package . . . . . 70°C/W
DW package . . . . . 58°C/W
N package . . . . . 69°C/W
PW package . . . . . 83°C/W
Storage temperature range, T_{stg} . . . . . −65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. The input and output voltage ratings may be exceeded if the input and output current ratings are observed.
2. The package thermal impedance is calculated in accordance with JESD 51-7.

recommended operating conditions (see Note 3)

			SN54HC573A			SN74HC573A			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage		2	5	6	2	5	6	V
V_{IH}	High-level input voltage	V_{CC} = 2 V	1.5			1.5			V
		V_{CC} = 4.5 V	3.15			3.15			
		V_{CC} = 6 V	4.2			4.2			
V_{IL}	Low-level input voltage	V_{CC} = 2 V			0.5			0.5	V
		V_{CC} = 4.5 V			1.35			1.35	
		V_{CC} = 6 V			1.8			1.8	
V_I	Input voltage		0		V_{CC}	0		V_{CC}	V
V_O	Output voltage		0		V_{CC}	0		V_{CC}	V
t_t	Input transition (rise and fall) time	V_{CC} = 2 V			1000			1000	ns
		V_{CC} = 4.5 V			500			500	
		V_{CC} = 6 V			400			400	
T_A	Operating free-air temperature		–55		125	–40		85	°C

NOTE 3: All unused inputs of the device must be held at V_{CC} or GND to ensure proper device operation. Refer to the TI application report, *Implications of Slow or Floating CMOS Inputs*, literature number SCBA004.

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS		V_{CC}	T_A = 25°C			SN54HC573A		SN74HC573A		UNIT
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
V_{OH}	V_I = V_{IH} or V_{IL}	I_{OH} = –20 µA	2 V	1.9	1.998		1.9		1.9		V
			4.5 V	4.4	4.499		4.4		4.4		
			6 V	5.9	5.999		5.9		5.9		
		I_{OH} = –6 mA	4.5 V	3.98	4.3		3.7		3.84		
		I_{OH} = –7.8 mA	6 V	5.48	5.8		5.2		5.34		
V_{OL}	V_I = V_{IH} or V_{IL}	I_{OL} = 20 µA	2 V		0.002	0.1		0.1		0.1	V
			4.5 V		0.001	0.1		0.1		0.1	
			6 V		0.001	0.1		0.1		0.1	
		I_{OL} = 6 mA	4.5 V		0.17	0.26		0.4		0.33	
		I_{OL} = 7.8 mA	6 V		0.15	0.26		0.4		0.33	
I_I	V_I = V_{CC} or 0		6 V			±0.1	±100		±1000		±1000	nA
I_{OZ}	V_O = V_{CC} or 0		6 V			±0.01	±0.5		±10		±5	µA
I_{CC}	V_I = V_{CC} or 0,	I_O = 0	6 V			8		160		80	µA
C_i			2 V to 6 V		3	10		10		10	pF

FAIRCHILD SEMICONDUCTOR™

October 1987
Revised April 2001

MM74C922 • MM74C923
16-Key Encoder • 20-Key Encoder

General Description

The MM74C922 and MM74C923 CMOS key encoders provide all the necessary logic to fully encode an array of SPST switches. The keyboard scan can be implemented by either an external clock or external capacitor. These encoders also have on-chip pull-up devices which permit switches with up to 50 kΩ on resistance to be used. No diodes in the switch array are needed to eliminate ghost switches. The internal debounce circuit needs only a single external capacitor and can be defeated by omitting the capacitor. A Data Available output goes to a high level when a valid keyboard entry has been made. The Data Available output returns to a low level when the entered key is released, even if another key is depressed. The Data Available will return high to indicate acceptance of the new key after a normal debounce period; this two-key roll-over is provided between any two switches.

An internal register remembers the last key pressed even after the key is released. The 3-STATE outputs provide for easy expansion and bus operation and are LPTTL compatible.

Features

- 50 kΩ maximum switch on resistance
- On or off chip clock
- On-chip row pull-up devices
- 2 key roll-over
- Keybounce elimination with single capacitor
- Last key register at outputs
- 3-STATE output LPTTL compatible
- Wide supply range: 3V to 15V
- Low power consumption

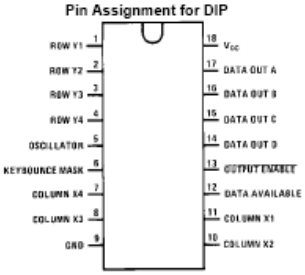
Ordering Code:

Order Number	Package Number	Package Description
MM74C922WM	M20B	20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300" Wide
MM74C922N	N18B	18-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide
MM74C923WM	M20B	20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300" Wide
MM74C923N	N20A	20-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Device also available in Tape and Reel. Specify by appending suffix letter "X" to the ordering code.

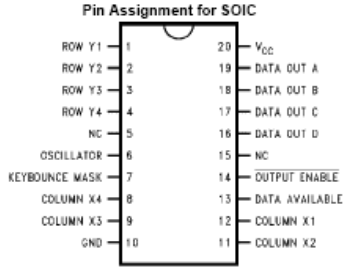
Connection Diagrams

Pin Assignment for DIP



Top View
MM74C922

Pin Assignment for SOIC

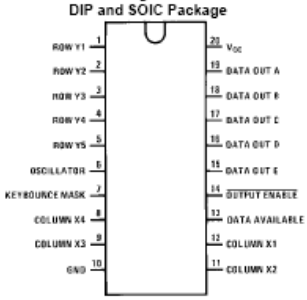


Top View
MM74C922

© 2001 Fairchild Semiconductor Corporation DS006037 www.fairchildsemi.com

Connection Diagrams (Continued)

Pin Assignment for
DIP and SOIC Package



Top View
MM74C923

Truth Tables

(Pins 0 through 11)

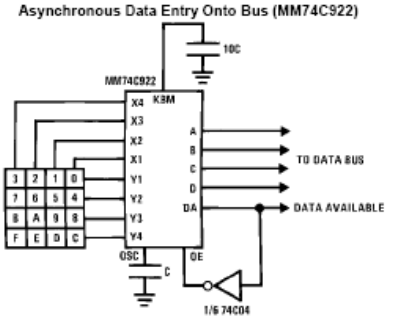
Switch Position		0 Y1, X1	1 Y1, X2	2 Y1, X3	3 Y1, X4	4 Y2, X1	5 Y2, X2	6 Y2, X3	7 Y2, X4	8 Y3, X1	9 Y3, X2	10 Y3, X3	11 Y3, X4
DATA	A	0	1	0	1	0	1	0	1	0	1	0	1
	B	0	0	1	1	0	0	1	1	0	0	1	1
	C	0	0	0	0	1	1	1	1	0	0	0	0
	D	0	0	0	0	0	0	0	0	1	1	1	1
	E (Note 1)	0	0	0	0	0	0	0	0	0	0	0	0

(Pins 12 through 19)

Switch Position		12 Y4, X1	13 Y4, X2	14 Y4, X3	15 Y4, X4	16 Y5(Note 1), X1	17 Y5 (Note 1), X2	18 Y5 (Note 1), X3	19 Y5 (Note 1), X4
DATA	A	0	1	0	1	0	1	0	1
	B	0	0	1	1	0	0	1	1
	C	1	1	1	1	0	0	0	0
	D	1	1	1	1	0	0	0	0
	E (Note 1)	0	0	0	0	1	1	1	1

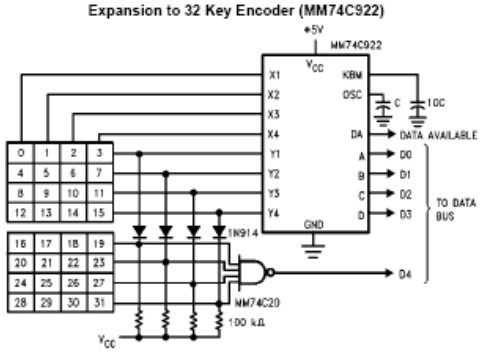
Note 1: Omit for MM74C922

Asynchronous Data Entry Onto Bus (MM74C922)



Outputs are in 3-STATE until key is pressed, then data is placed on bus. When key is released, outputs return to 3-STATE.

Expansion to 32 Key Encoder (MM74C922)



Theory of Operation

The MM74C922/MM74C923 Keyboard Encoders implement all the logic necessary to interface a 16 or 20 SPST key switch matrix to a digital system. The encoder will convert a key switch closer to a 4(MM74C922) or 5(MM74C923) bit nibble. The designer can control both the keyboard scan rate and the key debounce period by altering the oscillator capacitor, C_{OSE}, and the key bounce mask capacitor, C_{MSK}. Thus, the MM74C922/MM74C923's performance can be optimized for many keyboards.

The keyboard encoders connect to a switch matrix that is 4 rows by 4 columns (MM74C922) or 5 rows by 4 columns (MM74C923). When no keys are depressed, the row inputs are pulled high by internal pull-ups and the column outputs sequentially output a logic "0". These outputs are open drain and are therefore low for 25% of the time and otherwise off. The column scan rate is controlled by the oscillator input, which consists of a Schmitt trigger oscillator, a 2-bit counter, and a 2–4-bit decoder.

When a key is depressed, key 0, for example, nothing will happen when the X1 input is off, since Y1 will remain high. When the X1 column is scanned, X1 goes low and Y1 will go low. This disables the counter and keeps X1 low. Y1 going low also initiates the key bounce circuit timing and locks out the other Y inputs. The key code to be output is a combination of the frozen counter value and the decoded Y inputs. Once the key bounce circuit times out, the data is latched, and the Data Available (DAV) output goes high.

If, during the key closure the switch bounces, Y1 input will go high again, restarting the scan and resetting the key bounce circuitry. The key may bounce several times, but as soon as the switch stays low for a debounce period, the closure is assumed valid and the data is latched.

A key may also bounce when it is released. To ensure that the encoder does not recognize this bounce as another key closure, the debounce circuit must time out before another closure is recognized.

The two-key roll-over feature can be illustrated by assuming a key is depressed, and then a second key is depressed. Since all scanning has stopped, and all other Y inputs are disabled, the second key is not recognized until the first key is lifted and the key bounce circuitry has reset.

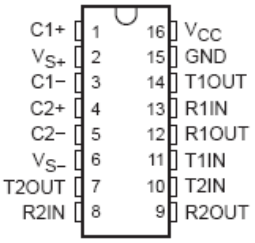
The output latches feed 3-STATE, which is enabled when the Output Enable ($\overline{OE}$) input is taken low.

MAX232, MAX232I
DUAL EIA-232 DRIVERS/RECEIVERS

SLLS047L – FEBRUARY 1989 – REVISED MARCH 2004

- Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
- Operates From a Single 5-V Power Supply With 1.0-μF Charge-Pump Capacitors
- Operates Up To 120 kbit/s
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current . . . 8 mA Typical
- ESD Protection Exceeds JESD 22
  - 2000-V Human-Body Model (A114-A)
- Upgrade With Improved ESD (15-kV HBM) and 0.1-μF Charge-Pump Capacitors is Available With the MAX202
- Applications
  - TIA/EIA-232-F, Battery-Powered Systems, Terminals, Modems, and Computers

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)



description/ordering information

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube of 25	MAX232N	MAX232N
	SOIC (D)	Tube of 40	MAX232D	MAX232
		Reel of 2500	MAX232DR	
	SOIC (DW)	Tube of 40	MAX232DW	MAX232
		Reel of 2000	MAX232DWR	
	SOP (NS)	Reel of 2000	MAX232NSR	MAX232
−40°C to 85°C	PDIP (N)	Tube of 25	MAX232IN	MAX232IN
	SOIC (D)	Tube of 40	MAX232ID	MAX232I
		Reel of 2500	MAX232IDR	
	SOIC (DW)	Tube of 40	MAX232IDW	MAX232I
		Reel of 2000	MAX232IDWR	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

LinASIC is a trademark of Texas Instruments.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 2004, Texas Instruments Incorporated

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

Function Tables

EACH DRIVER

INPUT TIN	OUTPUT TOUT
L	H
H	L

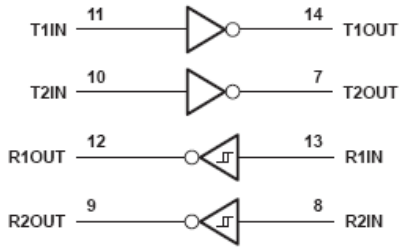
H = high level, L = low level

EACH RECEIVER

INPUT RIN	OUTPUT ROUT
L	H
H	L

H = high level, L = low level

logic diagram (positive logic)



TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Input supply voltage range, V_{CC} (see Note 1) . . . . . −0.3 V to 6 V
Positive output supply voltage range, V_{S+} . . . . . V_{CC} − 0.3 V to 15 V
Negative output supply voltage range, V_{S-} . . . . . −0.3 V to −15 V
Input voltage range, V_I: Driver . . . . . −0.3 V to V_{CC} + 0.3 V
Receiver . . . . . ±30 V
Output voltage range, V_O: T1OUT, T2OUT . . . . . V_{S-} − 0.3 V to V_{S+} + 0.3 V
R1OUT, R2OUT . . . . . −0.3 V to V_{CC} + 0.3 V
Short-circuit duration: T1OUT, T2OUT . . . . . Unlimited
Package thermal impedance, θ_{JA} (see Notes 2 and 3): D package . . . . . 73°C/W
DW package . . . . . 57°C/W
N package . . . . . 67°C/W
NS package . . . . . 64°C/W
Operating virtual junction temperature, T_J . . . . . 150°C
Storage temperature range, T_{stg} . . . . . −65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. All voltages are with respect to network GND.
2. Maximum power dissipation is a function of $T_J(max)$, θ_{JA}, and T_A. The maximum allowable power dissipation at any allowable ambient temperature is $P_D = (T_J(max) - T_A)/\theta_{JA}$. Operating at the absolute maximum T_J of 150°C can affect reliability.
3. The package thermal impedance is calculated in accordance with JESD 51-7.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	V
V_{IH}	High-level input voltage (T1IN,T2IN)		2			V
V_{IL}	Low-level input voltage (T1IN, T2IN)				0.8	V
R1IN, R2IN	Receiver input voltage				±30	V
T_A	Operating free-air temperature	MAX232	0		70	°C
		MAX232I	−40		85	

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (see Note 4 and Figure 4)

PARAMETER		TEST CONDITIONS	MIN	TYP‡	MAX	UNIT
I_{CC}	Supply current	V_{CC} = 5.5 V, All outputs open, T_A = 25°C		8	10	mA

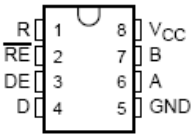
‡ All typical values are at V_{CC} = 5 V and T_A = 25°C.
NOTE 4: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

SN65176B, SN75176B
DIFFERENTIAL BUS TRANSCEIVERS

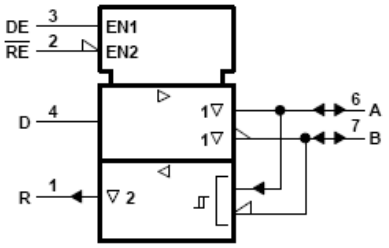
SLLS101A – JULY 1985 – REVISED MAY 1995

- Bidirectional Transceivers
- Meet or Exceed the Requirements of ANSI Standards EIA/TIA-422-B and RS-485 and ITU Recommendations V.11 and X.27
- Designed for Multipoint Transmission on Long Bus Lines in Noisy Environments
- 3-State Driver and Receiver Outputs
- Individual Driver and Receiver Enables
- Wide Positive and Negative Input/Output Bus Voltage Ranges
- Driver Output Capability . . . ±60 mA Max
- Thermal Shutdown Protection
- Driver Positive and Negative Current Limiting
- Receiver Input Impedance . . . 12 kΩ Min
- Receiver Input Sensitivity . . . ±200 mV
- Receiver Input Hysteresis . . . 50 mV Typ
- Operate From Single 5-V Supply

D OR P PACKAGE
(TOP VIEW)

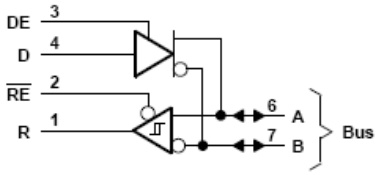


logic symbol†



† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

logic diagram (positive logic)



description

The SN65176B and SN75176B differential bus transceivers are monolithic integrated circuits designed for bidirectional data communication on multipoint bus transmission lines. They are designed for balanced transmission lines and meet ANSI Standards EIA/TIA-422-B and RS-485 and ITU Recommendations V.11 and X.27.

The SN65176B and SN75176B combine a 3-state differential line driver and a differential input line receiver, both of which operate from a single 5-V power supply. The driver and receiver have active-high and active-low enables, respectively, that can be externally connected together to function as a direction control. The driver differential outputs and the receiver differential inputs are connected internally to form differential input/output (I/O) bus ports that are designed to offer minimum loading to the bus whenever the driver is disabled or V_{CC} = 0. These ports feature wide positive and negative common-mode voltage ranges making the device suitable for party-line applications.

Function Tables

DRIVER

INPUT D	ENABLE DE	OUTPUTS A	OUTPUTS B
H	H	H	L
L	H	L	H
X	L	Z	Z

RECEIVER

DIFFERENTIAL INPUTS A – B	ENABLE $\overline{RE}$	OUTPUT R
$V_{ID} \geq 0.2$ V	L	H
-0.2 V < V_{ID} < 0.2 V	L	?
$V_{ID} \leq -0.2$ V	L	L
X	H	Z
Open	L	H

H = high level, L = low level, ? = indeterminate, X = irrelevant, Z = high impedance (off)

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 1995, Texas Instruments Incorporated

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

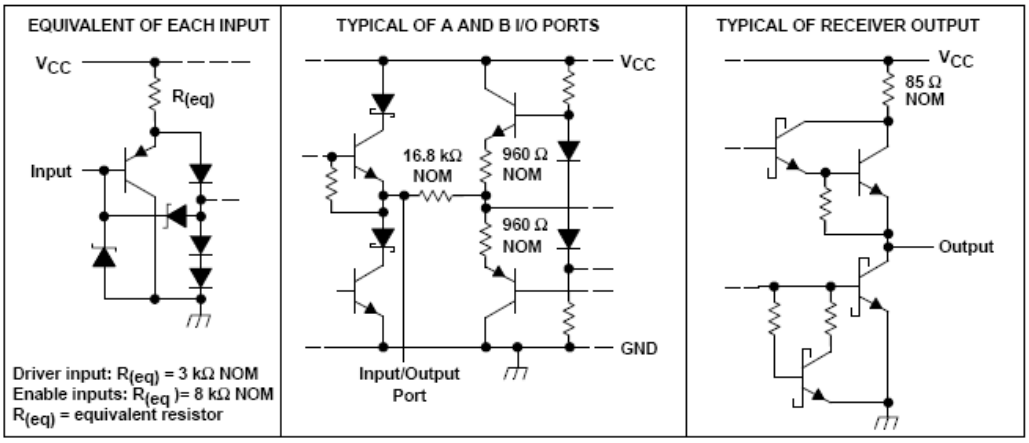
description (continued)

The driver is designed for up to 60 mA of sink or source current. The driver features positive- and negative-current limiting and thermal shutdown for protection from line-fault conditions. Thermal shutdown is designed to occur at a junction temperature of approximately 150°C. The receiver features a minimum input impedance of 12 kΩ, an input sensitivity of ±200 mV, and a typical input hysteresis of 50 mV.

The SN65176B and SN75176B can be used in transmission line applications employing the SN75172 and SN75174 quadruple differential line drivers and SN75173 and SN75175 quadruple differential line receivers.

The SN65176B is characterized for operation from –40°C to 105°C and the SN75176B is characterized for operation from 0°C to 70°C.

schematics of inputs and outputs



TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage, V_{CC} (see Note 1) . . . . . 7 V
Voltage range at any bus terminal . . . . . –10 V to 15 V
Enable input voltage, V_I . . . . . 5.5 V
Continuous total power dissipation . . . . . See Dissipation Rating Table
Operating free-air temperature range, T_A: SN65176B . . . . . –40°C to 105°C
SN75176B . . . . . 0°C to 70°C
Storage temperature range, T_{stg} . . . . . –65°C to 150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds . . . . . 260°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: All voltage values, except differential input/output bus voltage, are with respect to network ground terminal.

DISSIPATION RATING TABLE

PACKAGE	$T_A \le 25°C$ POWER RATING	DERATING FACTOR ABOVE $T_A = 25°C$	$T_A = 70°C$ POWER RATING	$T_A = 105°C$ POWER RATING
D	725 mW	5.8 mW/°C	464 mW	261 mW
P	1100 mW	8.8 mW/°C	704 mW	396 mW

recommended operating conditions

		MIN	TYP	MAX	UNIT
Supply voltage, V_{CC}		4.75	5	5.25	V
Voltage at any bus terminal (separately or common mode), V_I or V_{IC}				12	V
				–7	
High-level input voltage, V_{IH}	D, DE, and $\overline{RE}$	2			V
Low-level input voltage, V_{IL}	D, DE, and $\overline{RE}$			0.8	V
Differential input voltage, V_{ID} (see Note 2)				±12	V
High-level output current, I_{OH}	Driver			–60	mA
	Receiver			–400	µA
Low-level output current, I_{OL}	Driver			60	mA
	Receiver			8	
Operating free-air temperature, T_A	SN65176B	–40		105	°C
	SN75176B	0		70	

NOTE 2: Differential-input/output bus voltage is measured at the noninverting terminal A with respect to the inverting terminal B.

FAIRCHILD
SEMICONDUCTOR™

August 1986
Revised March 2000

DM74LS04
Hex Inverting Gates

General Description

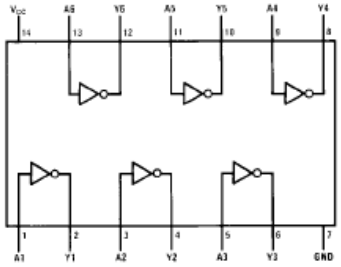
This device contains six independent gates each of which performs the logic INVERT function.

Ordering Code:

Order Number	Package Number	Package Description
DM74LS04M	M14A	14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-120, 0.150 Narrow
DM74LS04SJ	M14D	14-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS04N	N14A	14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

Connection Diagram



Function Table

$Y = \overline{A}$

Input	Output
A	Y
L	H
H	L

H = HIGH Logic Level
L = LOW Logic Level

© 2000 Fairchild Semiconductor Corporation DS006345 www.fairchildsemi.com

Absolute Maximum Ratings(Note 1)

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note 1: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V_{CC}	Supply Voltage	4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage	2			V
V_{IL}	LOW Level Input Voltage			0.8	V
I_{OH}	HIGH Level Output Current			−0.4	mA
I_{OL}	LOW Level Output Current			8	mA
T_A	Free Air Operating Temperature	0		70	°C

Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 2)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max	2.7	3.4		V
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IH} = Min		0.35	0.5	V
		I_{OL} = 4 mA, V_{CC} = Min		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V			0.1	mA
I_{IH}	HIGH Level Input Current	V_{CC} = Max, V_I = 2.7V			20	µA
I_{IL}	LOW Level Input Current	V_{CC} = Max, V_I = 0.4V			−0.36	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 3)	−20		−100	mA
I_{CCH}	Supply Current with Outputs HIGH	V_{CC} = Max		1.2	2.4	mA
I_{CCL}	Supply Current with Outputs LOW	V_{CC} = Max		3.6	6.6	mA

Note 2: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 3: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

Symbol	Parameter	R_L = 2 kΩ				Units
		C_L = 15 pF		C_L = 50 pF		
		Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	3	10	4	15	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	3	10	4	15	ns

FAIRCHILD SEMICONDUCTOR™

August 1986
Revised March 2000

DM74LS125A
Quad 3-STATE Buffer

General Description

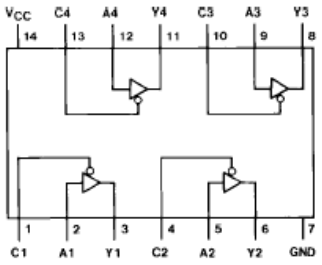
This device contains four independent gates each of which performs a non-inverting buffer function. The outputs have the 3-STATE feature. When enabled, the outputs exhibit the low impedance characteristics of a standard LS output with additional drive capability to permit the driving of bus lines without external resistors. When disabled, both the output transistors are turned off presenting a high-impedance state to the bus line. Thus the output will act neither as a significant load nor as a driver. To minimize the possibility that two outputs will attempt to take a common bus to opposite logic levels, the disable time is shorter than the enable time of the outputs.

Ordering Code:

Order Number	Package Number	Package Description
DM74LS125AM	M14A	14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-120, 0.150 Narrow
DM74LS125ASJ	M14D	14-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS125AN	N14A	14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

Connection Diagram



Function Table

Y = A

Inputs		Output
A	C	Y
L	L	L
H	L	H
X	H	Hi-Z

H = HIGH Logic Level
L = LOW Logic Level
X = Either LOW or HIGH Logic Level
Hi-Z = 3-STATE (Outputs are disabled)

© 2000 Fairchild Semiconductor Corporation DS006387 www.fairchildsemi.com

DM74LS125A

Absolute Maximum Ratings(Note 1)

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note 1: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V_{CC}	Supply Voltage	4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage	2			V
V_{IL}	LOW Level Input Voltage			0.8	V
I_{OH}	HIGH Level Output Current			−2.6	mA
I_{OL}	LOW Level Output Current			24	mA
T_A	Free Air Operating Temperature	0		70	°C

Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 2)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	2.4	3.4		V
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max		0.35	0.5	V
		I_{OL} = 12 mA, V_{CC} = Min		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V			0.1	mA
I_{IH}	HIGH Level Input Current	V_{CC} = Max, V_I = 2.7V			20	µA
I_{IL}	LOW Level Input Current	V_{CC} = Max, V_I = 0.4V			−0.4	mA
I_{OZH}	Off-State Output Current with HIGH Level Output Voltage Applied	V_{CC} = Max, V_O = 2.4V, V_{IH} = Min, V_{IL} = Max			20	µA
I_{OZL}	Off-State Output Current with LOW Level Output Voltage Applied	V_{CC} = Max, V_O = 0.4V, V_{IH} = Min, V_{IL} = Max			−20	µA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 3)	−20		−100	mA
I_{CC}	Supply Current	V_{CC} = Max (Note 4)		11	20	mA

Note 2: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 3: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Note 4: I_{CC} is measured with the data control (C) inputs at 4.5V and the data inputs grounded.

Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

Symbol	Parameter	R_L = 667Ω, C_L = 50 pF		R_L = 667Ω, C_L = 150 pF		Units
		Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output		15		21	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output		18		22	ns
t_{PZH}	Output Enable Time to HIGH Level Output		25		35	ns
t_{PZL}	Output Enable Time to LOW Level Output		25		40	ns
t_{PHZ}	Output Disable Time from HIGH Level Output (Note 5)		20			ns
t_{PLZ}	Output Disable Time from LOW Level Output (Note 5)		20			ns

Note 5: C_L = 5pF.

FAIRCHILD
SEMICONDUCTOR™

August 1986
Revised March 2000

DM74LS138 • DM74LS139
Decoder/Demultiplexer

General Description

These Schottky-clamped circuits are designed to be used in high-performance memory-decoding or data-routing applications, requiring very short propagation delay times. In high-performance memory systems these decoders can be used to minimize the effects of system decoding. When used with high-speed memories, the delay times of these decoders are usually less than the typical access time of the memory. This means that the effective system delay introduced by the decoder is negligible.

The DM74LS138 decodes one-of-eight lines, based upon the conditions at the three binary select inputs and the three enable inputs. Two active-low and one active-high enable inputs reduce the need for external gates or inverters when expanding. A 24-line decoder can be implemented with no external inverters, and a 32-line decoder requires only one inverter. An enable input can be used as a data input for demultiplexing applications.

The DM74LS139 comprises two separate two-line-to-four-line decoders in a single package. The active-low enable input can be used as a data line in demultiplexing applications.

All of these decoders/demultiplexers feature fully buffered inputs, presenting only one normalized load to its driving circuit. All inputs are clamped with high-performance Schottky diodes to suppress line-ringing and simplify system design.

Features

- Designed specifically for high speed:
  - Memory decoders
  - Data transmission systems
- DM74LS138 3-to-8-line decoders incorporates 3 enable inputs to simplify cascading and/or data reception
- DM74LS139 contains two fully independent 2-to-4-line decoders/demultiplexers
- Schottky clamped for high performance
- Typical propagation delay (3 levels of logic)
  - DM74LS138 21 ns
  - DM74LS139 21 ns
- Typical power dissipation
  - DM74LS138 32 mW
  - DM74LS139 34 mW

Ordering Code:

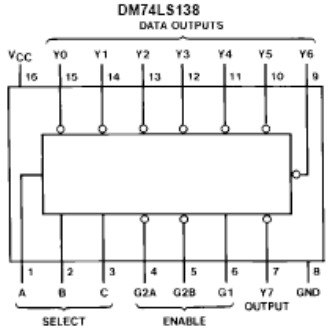
Order Number	Package Number	Package Description
DM74LS138M	M16A	16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150 Narrow
DM74LS138SJ	M16D	16-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS138N	N16E	16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide
DM74LS139M	M16A	16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150 Narrow
DM74LS139SJ	M16D	16-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS139N	N16E	16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

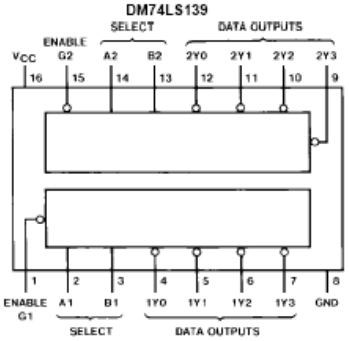
© 2000 Fairchild Semiconductor Corporation DS006391

Connection Diagrams

DM74LS138



DM74LS139



Function Tables

DM74LS138

Inputs					Outputs							
Enable		Select										
G1	G2 (Note 1)	C	B	A	YO	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

DM74LS139

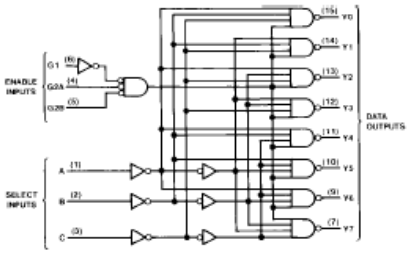
Inputs			Outputs			
Enable	Select					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = HIGH Level
L = LOW Level
X = Don't Care

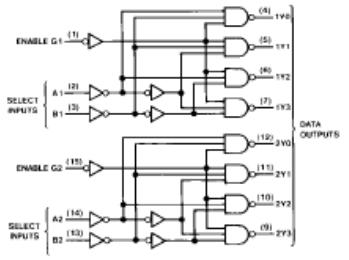
Note 1: G2 = G2A + G2B

Logic Diagrams

DM74LS138



DM74LS139



Absolute Maximum Ratings(Note 2)

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note 2: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

DM74LS138 Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V_{CC}	Supply Voltage	4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage	2			V
V_{IL}	LOW Level Input Voltage			0.8	V
I_{OH}	HIGH Level Output Current			−0.4	mA
I_{OL}	LOW Level Output Current			8	mA
T_A	Free Air Operating Temperature	0		70	°C

DM74LS138 Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 3)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	2.7	3.4		V
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min		0.35	0.5	V
		I_{OL} = 4 mA, V_{CC} = Min		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V			0.1	mA
I_{IH}	HIGH Level Input Current	V_{CC} = Max, V_I = 2.7V			20	µA
I_{IL}	LOW Level Input Current	V_{CC} = Max, V_I = 0.4V			−0.36	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 4)	−20		−100	mA
I_{CC}	Supply Current	V_{CC} = Max (Note 5)		6.3	10	mA

Note 3: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 4: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Note 5: I_{CC} is measured with all outputs enabled and OPEN.

DM74LS138 Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

Symbol	Parameter	From (Input) To (Output)	Levels of Delay	R_L = 2 kΩ				Units
				C_L = 15 pF		C_L = 50 pF		
				Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Select to Output	2		18		27	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Select to Output	2		27		40	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Select to Output	3		18		27	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Select to Output	3		27		40	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Enable to Output	2		18		27	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Enable to Output	2		24		40	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Enable to Output	3		18		27	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Enable to Output	3		28		40	ns

APENDICE B

Programa realizado en Keil μ Vision 3, Codigo Fuente para el Microcontrolador AT89C52

;---------- **DECLARACION DE DIRECTIVAS** ----------

;---

```
$NOMOD51          ; disable predefined 8051 registers
#include <reg52.h>  // include CPU definition file (for example, 8052)
;Programa para poder ver y almacenar los valores a transmitir
```

;---------- **DECLARACION DE VARIABLES** ----------

;---

```
reta1       EQU   030H      ;variables para la recarga de retardos
reta2       EQU   031H
ch1         EQU   040h      ;valores de canal en linea uno
ch2         EQU   041h
ch3         EQU   042h
val1        EQU   043h      ;valores de datos
val2        EQU   044h
val3        EQU   045h
vald        EQU   046h
backval     EQU   047h
tempshow    EQU   048H
tm1         EQU   049H
tm2         EQU   04AH
tm3         EQU   04BH
conttm      EQU   04CH
ddram       EQU   04DH
savea       EQU   050H      ;variables para manejo y respaldo de datos
datos       EQU   051H
banco       EQU   052H
escena      EQU   053H
escenav2    EQU   054H
contesc     EQU   055H
limite      EQU   056H
limitev     EQU   057H
limitev2    EQU   058H
```

```
indice      EQU   059H
temp        EQU   05AH
flashdph    EQU   060H          ;variables para el manejo y respaldo del dptr
flashdpl    EQU   061H
sramdph     EQU   062H
sramdpl     EQU   063H
editramh    EQU   064H
editraml    EQU   065H
sdph        EQU   066H
sdpl        EQU   067H
c1          EQU   070H          ;variables para el manejo de los valores de los timer´s
c2          EQU   071H
c3          EQU   072H
c33         EQU    073H
s1          EQU   074H
s11         EQU   075H
s2          EQU   076H
tc          EQU   077H
td          EQU   078H
sd          EQU   079H
escenax     EQU   07ah


;---------- ASIGNACION A PINES ----------
;----------------------------------------------------------------------------------------------------------------------------

a138        bit   p1.6          ;a138 = 0 habilita sram     a138 = 1 habilita flash
d           bit   p1.7          ;d = 1 habilita transmision en 485          d = 0    habilita transmicion en 232
bussy       bit   p2.7    ;(bussy (bussy=1      lcd ocupado en operacion interna, bussy=0 lcd disponible para recibir inst.)
e           bit   p3.3          ;Enable (E=0 modulo desactivado, E=1 modulo activado)
rw          bit   p3.4          ;R/W (R/W=0 LCD modo escritura, R/W=1 LCD modo lectura)
rs          bit   p3.5          ;RS (RS=0 registro de control, RS=1 registro de datos)
```

```
;---------------------------------- DECLARACION DE VECTORES DE INTERRUPCION ----------------------------
;-------------------------------------------------------------------------------------------------------------------------------

org  00h
ajmp inicio

org         03h
ajmp key

org  0bh
ajmp t0_speed

org 00dh
online:   db        '    ON-LINE',0

org  1bh
ajmp t1_time

org  23h
ajmp serie

;---------------------------------------- CUERPO DE PROGRAMA PRINCIPAL ----------------------------------------
;-------------------------------------------------------------------------------------------------------------------------------

org 025h
inicio:             mov p0,#00h
                    mov p2,#00h
                    mov r0,#00h             ;Limpieza del registro, almacena el dato obtenido de la subrutina key
                    mov banco, #00h
                    mov escena, #00h        ;Se inicializa banco y escena en el 1ero.
                    mov escenax, #00h
                    mov contesc, #'1'
                    mov ddram, #00h
                    clr d                   ;Inhabilita la transmision rs485  y habilita la transmicion en rs232
                    clr a138                ;Habilita control de memorias
                    clr e                   ;Habilita la memoria SRAM
                    acall clrram_ext        ;Limpia 512 bytes de memoria
```

```
        acall setup_lcd          ;Configura LCD
        setb    ea
        setb    ex0
        setb    et0
        setb    et1
        setb    px0
        setb    pt0
        setb    pt1
        setb    it0              ;Habilitacion general de interrupciones, habilitacion y prioridad a interrupciones
        mov tmod, #011h
        clr tf0
        clr tf1
        clr tr0
        clr tr1                  ;Configuracion de los TIMER´S
        acall ddram1
        mov dptr,#mensini1
        acall show
        acall ddram2
        mov dptr,#mensini2
        acall show               ;Despliega mensaje de INICIO
        mov editramh, escena
        mov editraml, #00h
        mov limitev, #01h
start1: cjne r0,#'a',start1      ;En espera de ingresar en modo MANUAL
        mov r0,#00h
        acall check_bussy
        mov p2,#01h
        acall pulso              ;Borra display
        acall ddram1
        mov dptr,#mens1
        acall show
        acall ddram2
        mov dptr,#mens2
        acall show               ;Despliga mensajes cha: y val:
        mov s1, #03h
        mov s11, #03h
        mov s2, #05h
```

```
                mov sd, #05h
                setb tr0            ;Valores iniciales de timer 0, y ACTIVACION
                mov ch1,#30h
                mov ch2,#30h
                mov ch3,#31h        ;Valores iniciales para contador de CANALES
                mov a,#00h
                mov dptr,#0000h     ;inicializa datapointer en 0000h ubicacion de mem. ext.  SRAM (a138 = 0 y e = 0)
canal:          mov dph, editramh
                mov dpl, editraml
                cjne r0,#00h,incch  ; R0, compara la entrada de valores en el teclado, r0=00 y refleja en display
                ajmp refleja1
incch:          cjne r0,#'6',decch  ; Tecla [ 6 ] *** INCREMENTA POSICION DE CANAL ***
                mov a,dpl
                cjne a,#0ffh,coninc
                mov a,dph
                cjne a,limitev,coninc
                ajmp refleja1
coninc:         inc dptr
                inc ch3
                mov r2,ch3
                cjne r2,#3ah,ve1
                inc ch2
                mov ch3,#30h
                mov r2,ch2
                cjne r2,#3ah,ve1
                inc ch1
                mov ch2,#30h
ve1:            ajmp refleja1
decch:          cjne r0,#'4',selch  ; Tecla [ 4 ] *** DECREMENTA POSICION DE CANAL ***
                mov a,dpl
                cjne a,#00h,condec
                mov  a, dph
                cjne a, limitev, conrev
                mov dph, escena
                ajmp condec
conrev:         cjne a, escena, condec
                ajmp refleja1
```

```
condec:     dec dpl
            dec ch3
            mov r2,ch3
            cjne r2,#2fh,ve2
            dec ch2
            mov ch3,#39h
            mov r2,ch2
            cjne r2,#2fh,ve2
            dec ch1
            mov ch2,#39h
ve2:        ajmp refleja1
selch:      cjne r0,#'5',enlinea
            mov r0,#00h          ; Tecla [ 5 ] *** SELECCIONA CANAL E INGRESA EN SU VALOR ***
            acall ddram14
            mov a,#'*'
            acall visualizar
            acall visualizar
            mov vald,#00h
            mov val1,#30h
            mov val2,#30h
            mov val3,#30h
            movx a,@dptr
            mov vald,a
            mov a,#00h
repval:     cjne a,vald,incvald
            acall valor
            ajmp canal
incvald:    inc a
            inc val3
            mov r3,val3
            cjne r3,#3ah,repval
            inc val2
            mov val3,#30h
            mov r3,val2
            cjne r3,#3ah,repval
            inc val1
            mov val2,#30h
```

```
                ajmp repval
enlinea:        cjne r0,#'a',r_banco      ; Tecla [ a ] *** ENTRA EN MODO ON-LINE Y DESHABILITA EL TECLADO *
                mov r0,#00h
                clr tr0
                acall check_bussy
                mov p2,#01h
                acall pulso               ;Instruccion para borrar display
                acall ddram1
                mov dptr,#online
                acall show                ;Despliega mensaje ON-LINE
                clr     ex0
                clr     px0               ;Deshabilita teclado
                clr d
                acall uart25pc            ;Configura el baud rate al del PC
                mov dptr, #0000h
                mov a,#'r'
                mov sbuf,a                ;Envio de palabra de inicio "ready"
                nop
                mov escena, #00h
wait:           cjne r0,#'w',wait_1       ;Espera de recepcion de 512 datos, y codigo de finalizado "w"
                mov r0, #00h              ;Termino de recibir
                clr tr2
                mov dptr,#0000h
                acall uartdmx             ;Saca datos recibidos por DMX
                mov dptr, #0000h
                acall uart25pc
wait_1:         cjne r0,#'a', wait_2
                clr tr2
                ajmp  inicio
wait_2:         ajmp wait
r_banco:        cjne r0, #'1', r_escena   ; Tecla [ 1 ] *** SELECCIONA UN BANCO ***
                mov r0, #00h
                acall incbanco
r_escena:       cjne r0, #'3', r_edit     ; Tecla [ 3 ] *** SELECCIONA UNA ESCENA ***
                mov r0, #00h
                acall incescena
r_edit:         cjne r0, #'b', r_load     ; Tecla [ b ] ***  SELECCIONA UNA ESCENA PARA EDITAR ***
```

```
                mov r0, #00h
                mov a, dph
                cjne a, escena, suma1
                mov editramh, escenax
                ajmp salsum
suma1:          mov a, escenax
                add a, #01
                mov editramh, a
salsum:         mov  escena, escenax
                mov editraml, dpl
                mov a, escena
                add a,#01h
                mov limitev,a
                acall ddrams
                mov a,#'e'
                acall visualizar
r_load:         cjne r0, #'c', r_save     ; Tecla [ c ] *** CARGA UN BANCO DESDE LA FLASH A SRAM ***
                mov r0, #00h
                clr tr0
                acall load
                mov s1, #03h
                mov s2, #05h
                setb tr0
r_save:         cjne r0, #'d', r_run      ; Tecla [ d ] *** GUARDA UNA ESCENA DESDE LA SRAM A LA FLASH
                mov r0, #00h
                clr tr0
                acall save
                mov s1, #03h
                mov s2, #05h
                setb tr0
r_run:          cjne r0, #'0', send       ; Tecla [ 0 ] *** SACA UN  BANCO COMPLETO, E X E ATRAVES DE DMX,
                mov r0, #00h              ; CARGADO PREVIAMENTE  EN LA SRAM  ***
                clr tr0
                ajmp run
send:           cjne r0, #'s', salir      ; Codigo de control generado por el timer 0, al finalizar el tiempo y envia escena
                mov r0,#00h
                acall uartdmx
```

```
                mov s1, #03h
                mov s2, #05h
                setb tr0
                ajmp canal
refleja1:       mov r0,#00h          ; Refleja los desplazamientos atraves de los canales
                mov editramh, dph
                mov editraml, dpl
                acall ddram11
                mov a,ch1
                acall visualizar
                mov a,ch2
                acall visualizar
                mov a,ch3
                acall visualizar
                acall ddram14
                mov a,#'<'
                acall visualizar
                mov a,#'>'
                acall visualizar
                mov a,dpl
                cjne a,#00h,incdec
                mov a,dph
                cjne a,escena,incdec
                acall ddram14
                mov a,#' '
                acall visualizar
                mov a,#'>'
                acall visualizar
incdec:         mov a,dpl
                cjne a,#0ffh,salir
                mov a,dph
                cjne a,limitev,salir
                acall ddram14
                mov a,#'<'
                acall visualizar
                mov a,#' '
                acall visualizar
```

```
salir:      ajmp canal
run:        mov dptr, #0000h        ; Subrutina para enviar un banco de escenas por medio de DMX,
            acall check_bussy       ; escena por escena y tiempo variable
            mov p2,#01h
            acall pulso             ; Borra display
            acall ddram1
            mov dptr,#speed
            acall show
            acall ddram2
            mov dptr,#time
            acall show              ; DespliEga mensajes spd: 0.5 s  y clk: 003s
            mov tm1, #'0'
            mov tm2, #'0'
            mov tm3, #'1'
            mov conttm, #00h        ; Valores iniciales para el conteo de tiempo TIME
            mov s1, #03h
            mov s11, #03h
            mov s2, #05h
            mov sd, #05h
            mov c33, #01eh
            mov c3, #01eh
            mov c2, #01h
            mov td,  #01h
            mov c1, #01h
            mov tc,  #01h
            setb tr0
inctime:    cjne r0,#'9',dectime    ; Tecla [ 9 ] *** INCREMENTO DE TIEMPO ENTRE ENVIO DE ESCENAS ***
             mov r0,#00h
             mov a, tc
             cjne a,#06h, inct
             ajmp  reflejat
inct:       inc tm3
             mov r3,tm3
             cjne r3,#3ah,reptm
             inc tm2
             mov tm3,#30h
             mov r3,tm2
```

```
                cjne r3,#3ah,reptm
                inc tm1
                mov tm2,#30h
reptm:          inc td
                mov a,td
                cjne a,#064h,exitinc
                mov td,#00h
                inc tc
exitinc:        ajmp reflejat
dectime:        cjne r0,#'#', incspeed      ; Tecla [ # ] *** DECREMENTO DE TIEMPO ENTRE  ENVIO DE ESCENAS
                mov r0,#00h
                mov a, tc
                cjne a, #06h, revspeed
                mov tc, #05h
                mov td, #64h
revspeed:       mov a, td
                cjne a,#01h,dectd
                mov td,#064h
                mov a, tc
                cjne a,#01h,dectc
                mov td,#01h
                ajmp  reflejat
dectd:          dec td
                dec tm3
                mov r2,tm3
                cjne r2,#2fh,vetm
                dec tm2
                mov tm3,#39h
                mov r2,tm2
                cjne r2,#2fh,vetm
                dec tm1
                mov tm2,#39h
vetm:           ajmp reflejat
dectc:          dec tc
                dec tm3
                mov r2,tm3
                cjne r2,#2fh,vetm2
```

```
                dec tm2
                mov tm3,#39h
                mov r2,tm2
                cjne r2,#2fh,vetm2
                dec tm1
                mov tm2,#39h
vetm2:          ajmp reflejat
reflejat:       acall ddram21
                 mov a, tm1
                 acall visualizar
                 mov a, tm2
                 acall visualizar
                 mov a, tm3
                 acall visualizar
incspeed:       cjne r0,#'7', decspeed      ; Tecla [ 7 ] *** INCREMENTO DE TIEMPO ENTRE REPETICION  DE BANCO
                 mov r0,#00h
                 inc sd
                 mov a, sd
                 cjne a, #0bh, preg
                 mov sd, #0ah
                 mov a, #0ah
preg:            cjne a,#0ah,reflejas
                 acall ddram11
                 mov a,#'1'
                 acall visualizar
                 mov a, #'.'
                 acall visualizar
                 mov a,#'0'
                 acall visualizar
decspeed:       cjne r0,#'*',exitrun        ; Tecla [ * ] *** DECREMENTO DE TIEMPO ENTRE REPETICION DE BANCO
                 mov r0,#00h
                 dec sd
                 mov a, sd
                 cjne a,#00h,reflejas
                 mov sd, #01h
reflejas:        acall ddram11
                 mov a,#'0'
```

```
                acall visualizar
                mov a, #'.'
                acall visualizar
                mov a, sd
                add a,#'0'
                acall visualizar
exitrun:        cjne r0,#'0', continua   ; Tecla [ 0 ] *** SALE DE ENVIO DE BANCO Y SALE A MODO MANUAL ***
                mov r0,#'a'
                clr tr1
                mov banco, #00h
                mov editramh, escena
                mov editraml, #00h
                mov limitev, #01h
                ajmp start1
continua:       cjne r0,#'s',continua2   ; Codigo de control generado por el timer 1, al finalizar el tiempo
                mov r0, #00h             ; y repite el envio de banco
                mov escenav2,#00h
                setb tr1
continua2:cjne r0, #'t', sales
                mov r0, #00h
sendesc:        mov a,escenav2           ;escenav2, entra con valor #00h para inciar desde la posicion 0000 de la SRAM
                add a,#01h               ;Se realiza la suma en una unidad a escenav2,  para obtener el limite variable 2
                mov limitev2, a          ;el resultado se almacena en la variable respectiva.
num2:           acall uartdmx3
                mov dph, escenav2
                mov dpl, #00h            ;Se recargan los valores de inico de escena en el DPTR
tren2:          setb tb8
                movx a,@dptr             ;Se obtiene el valor almacenado en la memoria SRAM
                mov sbuf,a               ;y se manda a traves de la uart a 250Kbps
                acall retardo1           ;retardo de espera entre envio de canales
                mov a,dpl
                cjne a,#0ffh, refe
                mov a,dph
                cjne a,limitev2, refe    ;Realiza la comprobacion del DPTR, con los valores limites,
                clr d                    ; para no pasarnos de la escena sacada al momento
                mov a,escenav2           ; Si termino de mandar la escena, deshabilita la transmision rs485
                add a,#02h
```

```
                mov escenav2, a
                cjne a,#010h,nextesc
                setb tr0
                ajmp inctime
refe:           inc dptr
                ajmp tren2
nextesc:        setb tr1
sales:          ajmp inctime
```

;---------- **MODIFICA VALOR DE CANAL** ----------

;---

```
org 3bfh

valor:          cjne r0,#00h,safeval      ; Subrutina para la modificacion del valor en un canal seleccionado
                ajmp refleja2
safeval:        cjne r0,#'5',incval       ; Tecla [ 5 ]  *** GUARDA EL VALOR MODIFICADO Y SALE DEL CANAL
                mov r0,#00h               ; SELECCIONADO ***
                mov a,vald
                movx @dptr,a
                ret
incval:         cjne r0,#'2',decval       ; Tecla [ 2 ]        *** INCREMENTA EL VALOR EN UN CANAL ***
                mov r3,vald
                cjne r3,#0ffh,siginc
                ajmp refleja2
siginc:         inc vald
                inc val3
                mov r3,val3
                cjne r3,#03ah,refleja2
                inc val2
                mov val3,#30h
                mov r3,val2
                cjne r3,#03ah,refleja2
                inc val1
                mov val2,#30h
```

```
                ajmp refleja2
decval:         cjne r0,#'8',send2      ; Tecla [ 8 ]  *** DECREMENTA EL VALOR EN UN CANAL  ***
                mov r3,vald
                cjne r3,#00h,sigdec
                ajmp refleja2
sigdec:         dec vald
                dec val3
                mov r3,val3
                cjne r3,#2fh,refleja2
                dec val2
                mov val3,#39h
                mov r3,val2
                cjne r3,#2fh,refleja2
                dec val1
                mov val2,#39h
                ajmp refleja2
send2:          cjne r0, #'s', salid    ; Codigo de control generado por el timer 0, al finalizar el tiempo y envia escena
                mov r0,#00h
                acall uartdmx
                mov s1, #03h
                mov s2, #05h
                setb tr0
                ajmp valor
refleja2:       mov r0,#00h
                mov a,vald
                movx @dptr,a
                acall ddram21
                mov a,val1
                acall visualizar
                mov a,val2
                acall visualizar
                mov a,val3
                acall visualizar
                acall ddram24
                mov a,#' '
                acall visualizar
                acall visualizar
```

```
                acall visualizar
                mov r4,vald
                cjne r4,#00h,sigue2
                acall ddram24
                mov a,#'M'
                acall visualizar
                mov a,#'I'
                acall visualizar
                mov a,#'N'
                acall visualizar
                mov r4,vald
sigue2:         cjne r4,#0ffh,salid
                acall ddram24
                mov a,#'M'
                acall visualizar
                mov a,#'A'
                acall visualizar
                mov a,#'X'
                acall visualizar
salid:          ajmp valor


;------------------------------------------------------------ RAM EXTERNA ------------------------------------------------------
;-------------------------------------------------------------------------------------------------------------------------------

org 46bh

clrram_ext:     mov dptr,#0000h
                mov a,#00h
ciclo1:         movx @dptr,a
                nop
                mov r4,dpl
                cjne r4,#0ffh, rec1
                mov r4,dph
                cjne r4,#01h, rec1
                mov dptr,#0000h
                ret
```

```
rec1:   inc dptr
        ajmp ciclo1
load:   clr tr0
        mov flashdph,banco
        mov flashdpl,#00h
        mov sramdph,#00h
        mov sramdpl,#00h
        mov a, banco
        add a,#10h
        mov limite, a
nextl:  mov a, flashdph
        cjne a,limite,readf
        acall ddramx
        mov a,#'l'
        acall visualizar
        mov s1, #03h
        mov s2, #05h
        setb tr0
        ret
readf:  mov dph, flashdph
        mov dpl, flashdpl
        setb a138
        movx a, @dptr
        inc dptr
        mov temp, a
        mov flashdph, dph
        mov flashdpl, dpl
        clr a138
        mov dph, sramdph
        mov dpl, sramdpl
        mov a, temp
        movx @dptr, a
        inc dptr
        mov sramdph, dph
        mov sramdpl, dpl
        ajmp nextl
```

```
save:   clr tr0
        mov datos, #80h        ;contabiliza 128 datos por meter a la flash
        mov a, banco
        add a, escena
        mov flashdph, a
        mov flashdpl, #00h
        mov sramdph, escena
        mov sramdpl, #00h
        mov a, escena
        add a, #02h
        mov limite,a
        acall sdp
nextd:  mov a,sramdph
        cjne a, limite, save1
        acall ddrams
        mov a,#'s'
        acall visualizar
        mov s1, #03h
        mov s2, #05h
        setb tr0
        ret
save1:  clr a138               ;habilita SRAM
        mov dph, sramdph
        mov dpl, sramdpl       ;Carga la ubicacion en el DPTR
        movx a, @dptr          ;Lee el dato de la SRAM y lo guarda en acumulador
        mov temp, a            ;Respalda el dato en la variable TEMP
        inc dptr
        mov sramdph, dph
        mov sramdpl, dpl       ;respalda la siguiente ubiccacion del DPTR, para la sram en la proxima lectura
        setb a138              ;deshabilita SRAM y activa FLASH
        mov dph, flashdph
        mov dpl, flashdpl      ;Carga la ubicacion en FLASH en la que se va a guardar el dato obtenido de la SRAM
        mov a, temp            ;recupera el dato por guardar
        movx @dptr, a          ;guarda el dato en la ubicacion de la FLASH
        inc dptr
        mov flashdph, dph
        mov flashdpl, dpl      ;Guarda la siguiente ubicacion de la flash
```

```
                djnz datos, nextdd          ;Decrementa y salta si aun DATOS no es cero
                acall retardo10
                mov datos, #80h
                acall sdp
nextdd:         ajmp nextd
incbanco:       mov a, banco
                add a, #10h
                mov banco, a
                cjne a, #60h, reflejab
                mov banco, #00h
                mov a, #00h
reflejab:       acall ddramx
                mov a,#' '
                acall visualizar
                mov a, #'B'
                acall visualizar
                mov a, banco
                swap a
                add a, #'1'             ; ajuste numero de banco a ASCII    para poder reflejarlo en lcd
                acall visualizar
                ret
incescena:      mov a, escenax
                add a, #02h
                mov escenax, a
                inc contesc
                cjne a, #10h, reflejae
                mov escenax, #00h
                mov contesc, #'1'
reflejae:       acall ddrams
                mov a, #' '
                acall visualizar
                mov a, #'E'
                acall visualizar
                mov a, contesc
                acall visualizar
                ret
sdp:            clr e
```

```
                setb a138
                mov dptr,#5555h
                mov a,#0aah
                movx @dptr,a
                nop
                clr a138
                nop
                setb a138
                mov dptr,#2aaah
                mov a,#055h
                movx @dptr,a
                nop
                clr a138
                nop
                setb a138
                mov dptr,#5555h
                mov a,#0a0h
                movx @dptr,a
                nop
                clr a138
                ret
```

;---------- **CONFIGURACION DE LCD** ----------

;----------

```
setup_lcd:      acall check_bussy
                mov p2,#38h          ;subrutina de configuracion del lcd, se coloca el bus para recibir 8bits,
                acall pulso          que presente los datos
                acall check_bussy    ;en pantalla en la primer linea y que los datos tengan el tamaño de 5x7 puntos
                mov p2,#01h          ;subrutina para limpiar la memoria y el display
                acall pulso
                acall check_bussy
                mov p2,#03h          ;subrutina para inicilalizar el cursor desde el origen, posicion 00h de la DDRAM
                acall pulso
                acall check_bussy
```

```
                mov p2,#0ch          ;subrutina para activar el display, activar cursor y que se encuentre parpadeando
                acall pulso
                acall check_bussy
                mov p2,#06h          ;subrutina para colocar el desplazamientode la inf. en el display durante
                acall pulso          ; lectura o escritura
                acall check_bussy    ;se incrementa el cursor a la der. y no desplaza la visualizacion
                mov p2,#14h          ;subrutina del sentido de desplazamiento de los caracteres en el display
                acall pulso          ;se desplaza el cursor hacia la derecha  y se mantiene lo que tiene la DDRAM
                ret
visualizar:     acall check_bussy
                acall enviar
                mov p2,a
                acall pulso
                ret
show:           mov a,#00h
                mov tempshow,a
rep1:           movc a, @a+dptr
                cjne a,#00h,sal1
                ret
sal1:           acall check_bussy
                acall enviar
                mov p2,a
                acall pulso
                inc tempshow
                mov a,tempshow
                ajmp rep1
check_bussy:    clr rs
                setb rw
                setb e
                setb bussy
L_BUSY:         jb bussy, L_BUSY
                clr e
                clr     rw
                ret
pulso:          setb e
                nop
                nop
```

```
                clr e
                ret
enviar:         setb rs
                clr     rw
                ret

;************************************** UBICACIONES DDRAM ******************************
;*****************************************************************************************

;LINEA1          00|01|02|03|04|05|06|07|08|09|0A|0B|0C|0D|0E|0F
;LINEA2          40|41|42|43|44|45|46|47|48|49|4A|4B|4C|4D|4E|4F

org 5Feh

ddram1:         mov ddram, #80h
                acall setddram
                ret
ddram11:        mov ddram,#085h
                acall setddram
                ret
ddram14:        mov ddram,#089h
                acall setddram
                ret
ddram2:         mov ddram,#0c0h
                acall setddram
                ret
ddram21:        mov ddram,#0c5h
                acall setddram
                ret
ddram24:        mov ddram,#0c9h
                acall setddram
                ret
ddramx:         mov ddram,#8dh
                acall setddram
                ret
ddrams:         mov ddram,#0cdh
                acall setddram
```

```
                ret
ddramb:         mov ddram,#8eh
                acall setddram
                ret
ddrame:         mov ddram,#0ceh
                acall setddram
                ret
setddram:       clr rw
                 clr        rs
                 acall check_bussy
                 mov p2,ddram
                 acall pulso
                 ret

;************************************ PROTOCOLO DMX 512 **********************************
;*****************************************************************************************

uart25pc:                       ; /* UART en modo 1 (10 bits) -bit start/8 bits datos/bit stop
                                ; /* RCLK = 1; TCLK=1; */
                                ; /* VALOR INICIAL */
                                ; /* VALOR INICIAL*/
                                ; /* VALOR DE RECARGA, 125000 Bps a 24MHz */
                                ; /* VALOR DE RECARGA, 125000 Bps a 24MHz*/
                                ; /* HABILITA INTERRUPCIONES GRAL.*/
                                ; /* HABILITA INTERRUPCION
                                ; /* HABILITA LA PRIORIDAD DE INTERRUPCION SERIE */
                                ; /* INICIA TIMER 2 */
                ret

uartdmx:
tren:                           ; Se envian los 512 canales ubicados en memoria sram

                                ;TIEMPO DE ESPERA APROX. ARRIBA DE 8 MICROSEG.
                 ret
rec2:
                 ajmp tren
```

```
uartdmx3:                ;habilita transmision rs485
                         /* UART en modo 3 (11 bits) -bit start/8 bits datos/tb8 programable/bit stop
                        /* RCLK = 0; TCLK=1; */
                        /* VALOR INICIAL */
                         /* VALOR INICIAL*/
                         /* VALOR DE RECARGA, 125000 Bps a 24MHz */
                        /* VALOR DE RECARGA, 125000 Bps a 24MHz*/
                        /* HABILITA INTERRUPCIONES GRAL.*/
                        /* HABILITA  INTERRUPCION SERIE */
                        /* HABILITA LA PRIORIDAD DE INTERRUPCION SERIE */
                        /* INICIA TIMER 2 */
                         ;GENERA LA SEÑAL DE BREAK
                         ;apaga timer 2
                         /* VALOR INICIAL */
                         /* VALOR INICIAL*/
                         /* VALOR DE RECARGA, 250000 Bps a 24MHz */

                         ;enciende timer 2
                         ;genera el  MAB  mayor a 8 us.
                         ;Cargo el START CODE
                         ;Lo envio
                         ;HABILITA LA SRAM
                        Ret

;---------- SUBRUTINAS DE INTERRUPCIONES ----------
;-------------------------------------------------------------------------------------------------------------------------

org 6d9h

key:        cjne r0, #'s', serchkey
            reti
serchkey:   clr tr0
            clr     ea
            mov savea,a
            mov sdpl,dpl
            mov sdph,dph
            mov dptr,#teclas
```

```
                mov     a,P1
                anl     a,#0Fh
                movc a,@a+dptr
                mov r0,a
                mov a,savea
                mov dpl,sdpl
                mov dph,sdph
                setb ea
                mov s2, #05h
                setb tr0
                reti
;*****************************************************************************************************

t0_speed:       clr tr0
                clr tf0
                mov savea,a
                mov sdpl,dpl
                mov sdph,dph
                djnz s1, sal_t0
                mov s1,s11
                djnz s2, sal_t0
                mov s2,sd
                mov r0, #'s'
                mov a,savea
                mov dpl,sdpl
                mov dph,sdph
                reti
sal_t0:         mov a,savea
                mov dpl,sdpl
                mov dph,sdph
                setb tr0
                reti
t1_time:        clr tr1
                clr tf1
                mov savea,a
                mov sdpl,dpl
                mov sdph,dph
```

```
                djnz c3, sal_t1
                mov c3,c33
                djnz c2, sal_t1
                mov c2,td
                djnz c1, sal_t1
                mov c1,tc
                mov r0, #'t'
                mov a,savea
                mov dpl,sdpl
                mov dph,sdph
                reti
sal_t1:         mov a,savea
                mov dpl,sdpl
                mov dph,sdph
                setb tr1
                reti
serie:          jbc ti, fin
                mov savea, a
                clr ri
                clr a138
                clr e
                mov a, sbuf
                movx @dptr,a
                nop
                mov r7, dpl
                cjne r7,#00h,rec4
                mov r7, dph
                cjne r7,#02h,rec4
                mov r0,a        ;Se almacena el codigo de finalizado, """ #12h """ Direccion 0200h - 512d
rec4:           inc dptr
                mov a,savea
fin:            reti
```

```
;------------------------------------------------------- RETARDOS ---------------------------------------------------
;--------------------------------------------------------------------------------------------------------------------

retardo1:       mov reta1,#01h
ret01:          mov reta2,#0ffh
ret02:          djnz reta2,ret02
                djnz reta1,ret01
                ret

retardo10:      mov reta1,#028h
ret10:          mov reta2,#0ffh
ret20:          djnz reta2,ret20
                djnz reta1,ret10
                ret

;------------------------------------------------------- LCD  MENSAJES ----------------------------------------------
;--------------------------------------------------------------------------------------------------------------------

org 079dh
teclas:         db '123a456b789c*0#d',0
org 07aeh
mens1:          db 'CHA: 001  >   B1',0
org 07bfh
mens2:          db 'VAL: 000 MIN  E1',0
org 07d0h
mensini1:       db      '***RPV_DMX512***',0
org 07e1h
mensini2:       db      '  -PROTOTIPO-',0
org 07f0h
speed:          db      'SPD: 0.5 s',0
org 07fbh
time:           db      'CLK: 001 s',0

END
```

APENDICE C

Programa realizado en Visual Basic 6.0 Aplicación " CONSOLA DMX512"

' Declaracion de variables grobales

```
Public INDICE0 As String
Public IDESCENA As Integer
Public faltan As Integer
Public IDBANCO  As Integer
Public interv2 As Integer
Public INI As Boolean
Public flagg As Boolean
Public bloque As Integer
Dim DMX512(512) As Integer
Dim POSICION As Integer
```

```
Private Sub Form_Load()                 ' Parametros iniciales
Form3.Show
see.Visible = False
send.Value = True
OFFLINE.Visible = False
ONLINE.Visible = True
SELBANCO.ListIndex = 0
Dim b As Integer                        ' carga la seleccion de bancos 1 - 6
For b = 1 To 6 Step 1
SELBANCO.List(b) = "BANCO" & b
Adodc1(b).Visible = False
Next b
For b = 0 To 8 Step 1
ESCENA(b).Visible = False
Next b
esc(1).Value = True                     ' inicializa la seleccion de escena 1, queda marcada
For POSICION = 1 To 512 Step 1
DMX512(POSICION) = 0
Next POSICION                           ' Limpia la matriz DMX512
MSComm1.CommPort = 4                    ' Designa el puerto para abrir
com4.Checked = True                     ' queda seleccionado en el menu
```

```
INI = False                          ' inicializa variable
Timer4.Interval = 500                ' carga valor predeterminado 0.5s
flagg = False                        ' inicializa variable
Slider1.Enabled = False              ' deshabilita control para timer
Slider2.Enabled = False              ' deshabilita control para timer
Timer3.Interval = 1000               ' carga valor predeterminado a 1s
valspeed.Caption = "0.5 s"           ' coloca etiquetas iniciales
valtime.Caption = "0.5 s"            ' coloca etiquetas iniciales
StatusBar1.Panels(5).Text = Time & "  " & Date    ' inicializa la barra de estado, fecha y hora
INDICE0 = ""                         'inicializa indice con dato nulo
MSComm1.Settings = "128000,n,8,1"    'Configura la UART  a 128Kbps sin paridad con 8 bits de datos y 1 bit
de stop
MSComm1.InBufferSize = 1             ' Tamaño del bufer de entrada o recepcion es uno
MSComm1.OutBufferSize = 1            ' Tamaño del bufer de salida o transmision es uno
MSComm1.RThreshold = 1
MSComm1.SThreshold = 1
MSComm1.InputLen = 0                 ' lee todo el bufer de entrada duando esta lleno
OFFLINE.BackColor = &HFF&            ' inicializa en boton de offline en rojo
End Sub

Private Sub acercade_Click()
form2.Show vbModal
End Sub


Private Sub deslizable_Change(Index As Integer)
VALOR(Index).Caption = Format(deslizable(Index).Value)
POSICION = Val(CANAL(Index).Caption)
DMX512(POSICION) = VALOR(Index).Caption
End Sub

Private Sub deslizable_Scroll(Index As Integer)
VALOR(Index).Caption = Format(deslizable(Index).Value)
POSICION = Val(CANAL(Index).Caption)
DMX512(POSICION) = VALOR(Index).Caption
End Sub
Private Sub bancos_Click(Index As Integer)
```

```
bloque = Index
Dim I As Integer
For I = 1 To 16 Step 1
CANAL(I).Caption = (I + (bloque) * (16))
POSICION = Val(CANAL(I).Caption)
deslizable(I).Value = DMX512(POSICION)
VALOR(I).Caption = DMX512(POSICION)
Next I
StatusBar1.Panels(1).Text = "    S e l e c c i o n  d e  b l o q u e :  " & bloque
End Sub

Private Sub CARGA_Click()                    ' carga el banco seleccionado
Dim n As Integer
IDBANCO = SELBANCO.ListIndex
If IDBANCO = 0 Then Exit Sub
For n = 0 To 8 Step 1
Set ESCENA(n).DataSource = Adodc1(IDBANCO)
Next n
Dim z As Integer
For z = 1 To 8 Step 1
If esc(z).Value = False Then Exit Sub
Adodc1(IDBANCO).Recordset.MoveFirst
IDESCENA = z
For n = 1 To 512 Step 1
DMX512(n) = ESCENA(IDESCENA).Text
Adodc1(IDBANCO).Recordset.MoveNext
If Adodc1(IDBANCO).Recordset.EOF Then
Adodc1(IDBANCO).Recordset.MoveFirst
End If
Next n
Dim I As Integer
For I = 1 To 16 Step 1
CANAL(I).Caption = (I + (bloque) * (16))
POSICION = Val(CANAL(I).Caption)
deslizable(I).Value = DMX512(POSICION)
VALOR(I).Caption = DMX512(POSICION)
Next I
```

```
Next z
End Sub

Private Sub GUARDA_Click()
If IDESCENA = 0 Then Exit Sub
Text1.Text = IDESCENA
Adodc1(IDBANCO).Recordset.MoveFirst
For n = 1 To 512 Step 1
ESCENA(IDESCENA).Text = DMX512(n)          ' Guarda la escena editada
Adodc1(IDBANCO).Recordset.MoveNext
If Adodc1(IDBANCO).Recordset.EOF Then
Adodc1(IDBANCO).Recordset.MoveFirst
End If
Next n
Adodc1(IDBANCO).Recordset.Update
End Sub

Private Sub ESC_Click(Index As Integer)
Dim z As Integer                           ' Seleccion de escena para editar
For z = 1 To 8 Step 1
If esc(z).Value = True Then                ' identifica escena seleccionada
If IDBANCO = 0 Then Exit Sub               ' comprueba que banco sea valido
Adodc1(IDBANCO).Recordset.MoveFirst
IDESCENA = z
For n = 1 To 512 Step 1
DMX512(n) = ESCENA(IDESCENA).Text          ' Descarga la escena de la tabla a la matriz dmx para
trabajarla
Adodc1(IDBANCO).Recordset.MoveNext
If Adodc1(IDBANCO).Recordset.EOF Then
Adodc1(IDBANCO).Recordset.MoveFirst
End If
Next n
Adodc1(IDBANCO).Recordset.Update           ' actualiza la escena
Dim I As Integer
For I = 1 To 16 Step 1
CANAL(I).Caption = (I + (bloque) * (16))   ' actualiza los deslizables con los valores de la escena seleccionada
POSICION = Val(CANAL(I).Caption)
```

```
deslizable(I).Value = DMX512(POSICION)
VALOR(I).Caption = DMX512(POSICION)
Next I
End If
Next z
End Sub

Private Sub online_Click()
If MSComm1.PortOpen = False Then
OFFLINE.BackColor = &H8000000F
MSComm1.PortOpen = True
ONLINE.BackColor = &HFF00&                    'Procedimiento para abrir el puerto seleccionado
ONLINE.Visible = False
OFFLINE.Visible = True
StatusBar1.Panels(4).Text = "com " & MSComm1.CommPort
Else
MsgBox "com_4 esta abierto", , "com_4"
End If
End Sub

Private Sub offline_Click()
If MSComm1.PortOpen = True Then
ONLINE.BackColor = &H8000000F
MSComm1.PortOpen = False
OFFLINE.BackColor = &HFF&                     'Procedimiento para cerrar el puerto seleccionado
ONLINE.Visible = True
OFFLINE.Visible = False
StatusBar1.Panels(4).Text = "com " & MSComm1.CommPort
Else
MsgBox "com_4 esta cerrado", , "com_4"
End If
End Sub
Private Sub RESET_Click()
For POSICION = 1 To 512 Step 1
DMX512(POSICION) = 0
Next POSICION
Dim I As Integer                    ' Procedimiento para resetear los deslizables y matriz dmx a ceros
```

```
For I = 1 To 16 Step 1
CANAL(I).Caption = (I + (bloque) * (16))
POSICION = Val(CANAL(I).Caption)
deslizable(I).Value = DMX512(POSICION)
VALOR(I).Caption = DMX512(POSICION)
Next I
End Sub

Private Sub com1_Click()
MSComm1.CommPort = 1                    ' Seleccion de puerto serie  com 1 para abrir
com1.Checked = True
com4.Checked = False
End Sub

Private Sub com4_Click()
MSComm1.CommPort = 4                    ' Seleccion de puerto serie  com 4 para abrir
com4.Checked = True
com1.Checked = False
End Sub

Private Sub MSComm1_OnComm()
If MSComm1.CommEvent = comEvReceive Then
INDICE0 = MSComm1.Input
see.Text = INDICE0
If INDICE0 = "r" Then
enlaza.Caption = "CONECTADO"
Else
End If
End If
End Sub
Private Sub RESETT_Click()
If INI = False Then                     ' Resetea los timer, restablece los tiempos a los predeterminados
Exit Sub
Else
Timer1.Enabled = False
Slider1.Value = 500
Timer2.Enabled = True
```

```
Slider2.Value = 500
Timer2.Interval = 500
interv2 = 1
End If
End Sub

Private Sub salir_Click()
Unload Form1                    ' Cierra el la aplicacion
End
End Sub

Private Sub Slider1_Change()
Dim l As Variant
l = ((Slider1.Value) / 1000)
valtime.Caption = l & " s"      ' slider1 lleva el control de tiempo TIME, altera el valor de recarga
Timer1.Interval = l
End Sub

Private Sub Slider1_Click()
Dim l As Variant
l = ((Slider1.Value) / 1000)
valtime.Caption = l & " s"      ' slider1 lleva el control de tiempo TIME, altera el valor de recarga
Timer1.Interval = l
End Sub

Private Sub Slider1_KeyDown(KeyCode As Integer, Shift As Integer)
Dim l As Variant
l = ((Slider1.Value) / 1000)
valtime.Caption = l & " s"      ' slider1 lleva el control de tiempo TIME, altera el valor de recarga
Timer1.Interval = l
End Sub

Private Sub Slider1_KeyUp(KeyCode As Integer, Shift As Integer)
Dim l As Variant
l = ((Slider1.Value) / 1000)
valtime.Caption = l & " s"      ' slider1 lleva el control de tiempo TIME, altera el valor de recarga
Timer1.Interval = l
```

```
End Sub

Private Sub Slider2_Change()
Dim ll As Variant
ll = ((Slider2.Value) / 1000)          ' slider2 lleva el control de tiempo SPEED, altera el valor de recarga
valspeed.Caption = ll & " s"
Timer2.Interval = ll
End Sub

Private Sub Slider2_Click()
Dim ll As Variant
ll = ((Slider2.Value) / 1000)          ' slider2 lleva el control de tiempo SPEED, altera el valor de recarga
valspeed.Caption = ll & " s"
Timer2.Interval = ll
End Sub

Private Sub Slider2_KeyDown(KeyCode As Integer, Shift As Integer)
Dim ll As Variant
ll = ((Slider2.Value) / 1000)
valspeed.Caption = ll & " s"           ' slider2 lleva el control de tiempo SPEED, altera el valor de recarga
Timer2.Interval = ll
End Sub

Private Sub Slider2_KeyUp(KeyCode As Integer, Shift As Integer)
Dim ll As Variant
ll = ((Slider2.Value) / 1000)
valspeed.Caption = ll & " s"           ' slider2 lleva el control de tiempo SPEED, altera el valor de recarga
Timer2.Interval = ll
End Sub
Private Sub start_Click()
If INI = False Then                    ' Procedimiento destinado a habilitar los timer 1 y timer 2,
Timer4.Enabled = False                 ' slider1 y slider2 y se puedan utilizar
SELECTOR.Enabled = False
Slider1.Enabled = True
Slider2.Enabled = True                 ' Los cuales llevan el control de tiempo, tiempo que transcurre
Timer1.Interval = 0                    ' en eviar una escena, terminada otra y el tiempo que tarda
Timer1.Enabled = False                 ' en volver a enviar las 8 escenas
```

```
interv2 = 0
Timer2.Enabled = True
Timer2.Interval = 500
start.Caption = "STOP"
INI = True
Else
INI = False
Timer4.Enabled = True
SELECTOR.Enabled = True
Slider1.Enabled = False
Slider2.Enabled = False
start.Caption = "RUN"
RESETT_Click
Timer1.Enabled = False
Timer2.Enabled = False
End If
End Sub

Private Sub Timer1_Timer()
Timer2.Enabled = True                ' Procedimiento destinado al timer 1, tiempo que trasncurre
Timer1.Enabled = False               ' entre envio de banco
Timer2.Interval = Slider2.Value
interv2 = 0
End Sub

Private Sub Timer2_Timer()
Dim k As Integer
For k = 1 To 8 Step 1                ' Hace visible las etiquetas de las escenas
esce(k).Visible = True
Next k
interv2 = interv2 + 1                ' lleva el conteo de la escena actual
If interv2 = 9 Then                  ' Si se han enviado mas de 8 escenas, deshabilita el timer 2,
Timer2.Enabled = False               ' y habilita el timer 1, recargandolo del slider1.value
Timer1.Enabled = True
Timer1.Interval = Slider1.Value
Exit Sub
End If
```

```
Adodc1(IDBANCO).Recordset.MoveFirst          ' Se inicia el indice en la primer posicion
For n = 1 To 512 Step 1
DMX512(n) = ESCENA(interv2).Text             ' Guarda los datos de la escena en la matriz dmx512
Adodc1(IDBANCO).Recordset.MoveNext           ' Avanza al siguiente dato
If Adodc1(IDBANCO).Recordset.EOF Then        ' Revisa que no ha llegado al final
Adodc1(IDBANCO).Recordset.MoveFirst          ' Devuelve el indice al inicio
End If
Next n
Dim I As Integer                             ' Muestra la informacion actual en los deslizables
For I = 1 To 16 Step 1
CANAL(I).Caption = (I + (bloque) * (16))
POSICION = Val(CANAL(I).Caption)
deslizable(I).Value = DMX512(POSICION)
VALOR(I).Caption = DMX512(POSICION)
Next I
For n = 1 To 512 Step 1
MSComm1.Output = Chr(DMX512(n))
Next n
If send.Value = True Then
MSComm1.Output = "w"                         ' sale de  online y vuelve a modo manual
Else
MSComm1.Output = "a"                         ' avisa que ha terminado de enviar los 512 canales
End If
esce(interv2).Visible = False                ' Muestra la escena en curso saliendo
Timer2.Interval = Slider2.Value              ' Recarga el timer 2, del slider2.value
End Sub

Private Sub Timer3_Timer()
StatusBar1.Panels(5).Text = Time & "  " & Date   ' Timer 3, refresca la hora y la fecha cada segundo
Timer3.Interval = 1000
End Sub

Private Sub Timer4_Timer()
Dim k As Integer
Dim n As Integer
Dim m As Integer
For k = 1 To 8 Step 1                ' Hace visibles las etiquetas de escena seleccionables
```

```
esce(k).Visible = True
Next k
For k = 1 To 8 Step 1                    ' Busca la escena que esta seleccionada
If esc(k).Value = True Then
If IDBANCO = 0 Then Exit Sub             ' Revisa que la escena seleccionada, pertenezca a algun banco
For n = 1 To 512 Step 1
MSComm1.Output = Chr(DMX512(n))
Next n
Dim I As Integer
For I = 1 To 16 Step 1                   ' Actualiza la informacion que esta saliendo, dependiendo del banco seleccionado
CANAL(I).Caption = (I + (bloque) * (16))
POSICION = Val(CANAL(I).Caption)
deslizable(I).Value = DMX512(POSICION)
VALOR(I).Caption = DMX512(POSICION)
Next I
If send.Value = True Then
MSComm1.Output = "w"                     ' sale de   online y vuelve a modo manual
Else
MSComm1.Output = "a"                     ' avisa que ha terminado de enviar los 512 canales
End If
If flagg = False Then                    ' Apaga y enciende la etiqueta de la escena que se esta editando
flagg = True
esce(k).Visible = False
Else
flagg = False
esce(k).Visible = True
End If
Adodc1(IDBANCO).Recordset.Update         ' Actualiza la escena selecionada
End If
Next k
Timer4.Interval = 500                    ' Recarga el timer 4 nuevamente para que comienze nuevamente
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
Select Case Button.Key
Case "btsalir"
Unload Form1
```

```
End                    ' Procedimiento destinado a las funciones de la barra de herramientas
Case "btconexion"
online_Click
Case "bt actualizar"
End Select
End Sub
```