24
Tejeu

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

FALLA DE ORIGEN

DISEÑO Y CONSTRUCCION DE UN SISTEMA AUTOMATICO DE REGISTRO DE ASISTENCIAS PARA EL PERSONAL ADMINISTRATIVO DE LA FACULTAD DE INGENIERIA

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

A R E A D E E L E C T R O N I C A

P R E S E N T A N :

MARIA DEL CARMEN ARENAS SANCHEZ

SAUL DAVID SANTOS JAUREGUI

DIRECTOR DE TESIS:
M. EN I. LAURO SANTIAGO CRUZ

CIUDAD UNIVERSITARIA, JULIO DE 1995

FALLA DE ORIGEN

Agradecemos

Al personal de la Secretaría Administrativa de la Facultad de Ingeniería, en especial al M en I. Bernardo Frontana de la Cruz, al Ing. Luis Jiménez Escobar, a la Sra. Laura Villarán Toscano y al Sr. José Antonio Reyes Rodríguez, por las facilidades prestadas para la elaboración de esta Tesis.

Al Instituto de Ingeniería y en especial, a la Coordinación de Instrumentación, así como a todos los que en ella laboran y que de alguna forma contribuyeron a la elaboración de este trabajo.

A la Coordinación de Sistemas de Cómputo del Instituto de Ingeniería, en especial a Ricardo, Gustavo, Alva, Artemia, Claudia, Alejandro y Mauricio, por su valiosa cooperación.

A la D. I. Leticia Gaytán por su importante intervensión en el desarrollo de este trabajo.

Al Dr. Baltazar Mena Iniesta y al Ing. Carlos B. Motta Miranda por su apoyo en la elaboración de esta Tesis.

En especial gracias al M. en I. Lauro Santiago Cruz por su tiempo, apoyo y amistad, como profesor en la carrera y como director de esta Tesis.

A la Facultad de Ingeniería por apoyar este proyecto y muy especialmente, por ser el elemento principal en nuestra formación profesional.

A la Universidad Nacional Autónoma de México, por todas las oportunidades que nos brindó y que fueron la causa de este objetivo.

Y a todas las personas que faltan por mencionar y no por ello menos importantes, sino que al contrario, sin ellas no hubiésemos podido llegar hasta aquí.

María del Carmen Arenas Sánchez
Saúl David Santos Jáuregui

*A **Dios**, le agradezco todo, porque alimenta mi espíritu, porque fortalece mi ser.*

A mis padres,
A mis abuelos Ana María y Faus
A mis hermanos Mara, Francisco y Abril.
Porque su amor y su apoyo son pilares de mi vida.

A todos los que fueron parte importante en mi formación, porque me legaron algo de ellos, su tiempo, su amor, su saber.

A mis amigos, porque simplemente forman parte de los mejores momentos de mi vida.

A Saúl, por su franqueza y sencillez, por ser especial.

María del Carmen.

You Only Live Once,
But if You Live Right,
Once is Enough.

A mis padres por que sin su apoyo y comprensión jamas hubiera alcanzado esta meta en especial a mi madre.

A mis hermanos Efraín, Irma, Sandra y Armando por sus consejos y apoyo.

A mi director de tesis M. en I. Lauro Santiago Cruz por brindarme su amistad y apoyo incondicional.

A mis profesores de la Facultad de Ingeniería por su paciencia en mi aprendisaje.

A mis amigos que siempre me apoyarón en especial Hector Benitez, Carlos Meza, Guillermo Mercado, Leonardo Freyssinier, Elsa Barón, Daniel Ortega y demas compañeros de la Facultad con quienes viví grandes momentos y experiencias.

A mi compañera de tesis María del Carmen por que este trabajo me ha permitido conocerla y admirarla profundamente.

A Dios que me ha permitido terminar este trabajo.

Saúl David Santos Jáuregui

Indice

INTRODUCCION 1

CAPITULO 1
Generalidades sobre Códigos de Barras 1

Código de barras y cinta magnética 1
Código de barras 3
Conceptos básicos 5
Simbologías 9
Escogiendo una simbología 17
Eficiencia de una simbología 17
Seguridad en los datos 19
Calidad de impresión 25
Estándares en códigos de barras 26
Lectura óptica 28
Equipos de lectura 28
Rendimiento 33
Impresión de códigos de barras 33

CAPITULO 2
Elementos del Sistema **35**

Selección de elementos **37**
Bloque de identificación y almacenamiento **38**
Memoria de sólo lectura **39**
Memoria de acceso aleatorio **40**
El micro-controlador **41**
El reloj de tiempo real **48**
Memoria IC **56**
El *SCANNER* **59**
Pantalla de cristal líquido **62**
Comunicación entre una PC y el sistema de identificación **64**

CAPITULO 3
Arquitectura del Sistema **68**

Estructura del Sistema **68**
Métodos para el direccionamiento memoria **69**
Asignación de direcciones a los elementos del sistema **73**
La ROM **76**
La RAM **76**
El RTR **76**
El LCD **78**
Comunicación serial **81**
La tarjeta IC **83**
Fuente de alimentación **85**

CAPITULO 4
Programación del Sistema y la PC **91**

Decodificación **92**
Comunicación del sistema de identificación con una PC **101**
Manejo de archivos en la computadora personal **104**

RESULTADOS Y CONCLUSIONES **110**

BIBLIOGRAFIA

APENDICES

A: *Hojas de Especificaciones*

B: *Listado de Programa*

Introducción

Dentro de las condiciones generales de trabajo se estipula que "las instituciones o particulares" que celebran un convenio de trabajo, tienen la obligación de remunerar el mismo a su personal. Para tal efecto, ambas partes deben contar con un medio de comprobación oficial, esto es, un registro confiable que contenga el nombre del trabajador, la fecha, la hora de llegada y la hora de salida, lo cual permite calcular la remuneración del trabajador. Entre los métodos más comunes para realizar dicho registro se encuentran:

- **Sistema de firmas**, esto es, se elabora una relación de los trabajadores que entran a una hora establecida. El trabajador debe llegar y registrar la hora en que llega y su firma. Después de un lapso de tiempo la relación de firmas es retirada y llevada al lugar donde se efectúa el control de asistencia.

- **Reloj checador,** consta de un mecanismo adaptado a un reloj (que puede ser mecánico o electrónico). El mecanismo imprime la hora y fecha actual en la tarjeta del trabajador. Una vez que se ha pasado el tiempo de tolerancia para la entrada, se recogen las tarjetas y se almacena su información. Elaborando así la relación de asistencias del personal.

Actualmente muchos de los sistemas de registro han sido modernizados de manera que se simplifican los pasos que realiza el personal encargado del control de asistencias. Estos sistemas en su mayoría son electrónicos y con la identificación del trabajador registran todos sus datos, los que van directamente a una base de datos y el control se hace de manera automática.

Objetivo

El presente trabajo tiene como objetivo el desarrollo de un sistema de control de asistencias que atiende a las necesidades de la **Direccion de Personal Administrativo de la Facultad de Ingenieria**.

Las características principales de este sistema son el agilizar la captura de la información de las tarjetas de los trabajadores y proporcionar además una lista de ocurrencias, incidencias, retardos y tiempos extras y con ello disminuir el tiempo, el trabajo y los errores, como consecuencia del proceso de recuento de las tarjetas.

El sistema ha desarrollar estará controlado por un microprocesador, se basará en un módulo óptico de lectura de código de barras. El código contendrá el R.F.C. del trabajador, éste junto con la fecha y hora de checado serán almacenados en el sistema que interactuarán con una Computadora Personal (PC) para el manejo de la información.

Aunque ya existen en el mercado relojes comerciales basados en lectura óptica, la innovación que se presenta en este trabajo es la de incluir dentro del sistema una tarjeta de memoria TC-RAM (Thin Card), cuyas características son: una gran capacidad de almacenamiento, y la facilidad de transportarse a una PC con el fin de tener acceso a la información sin necesidad del sistema.

El trabajo de tesis que se presenta está estructurado de la siguiente manera:

En el capítulo 1 se hablará de las generalidades de los códigos de barras, sus conceptos básicos, tipos de simbologías, los diferentes tipos de lectores ópticos, además de una explicación de como se realiza una lectura de código de barras y algunos criterios y parámetros sobre la evaluación de un sistema basado en una tecnología de código de barras.

En el capítulo 2 se hablará de los elementos que deben constituir al sistema de control de asistencias, explicando la forma en se deben programar así como sus principales característica por las que fueron seleccionados.

En el capítulo 3 se hablará como se estructura la arquitectura para el funcionamiento del sistema de control de asistencias y la posición de cada elemento dentro de el mapa de memoria.

En el capítulo 4 se hablará de cómo fueron desarrollados los programas para el funcionamiento del sistema de control de asistencias, explicando las partes fundamentales de los programas de la parte electrónica como el programa de la computadora personal y la forma en que la computadora personal procesa la información.

Finalmente se expondrán los resultados y conclusiones obtenidos del desarrollo de este trabajo de tesis, así como los listados de los programas y las hojas de especificaciones de los elementos más importantes del sistema.

Definición del problema

Actualmente, en la Facultad de Ingeniería de la UNAM, para realizar el control de personal, se cuenta con un reloj checador y un sistema de tarjetas, en las cuales se registran la fecha, hora de entrada, hora de salida y horas extras (en el caso de que éstas existan), de cada trabajador.

Para llevar el registro de ocurrencias de los trabajadores, es necesario que cada semana una persona realice manualmente el recuento de todas las tarjetas y capture esta información para el sistema de nómina, obteniendo así las faltas, retardos, horas extras, incentivos, etc., de cada trabajador. El proceso anterior puede causar errores como por ejemplo, la mala captura de la información, un procesamiento erróneo al calcular las horas de trabajo, retraso de tiempo al revisar las tarjetas de los trabajadores y la posible pérdida de información para el caso en que la tarjeta se extravíe o se destruya.

Finalmente, se elabora un formato específico por medio de un programa para PC, proporcionado por la Dirección General de Personal (DGP). Este programa sólo permite la entrada de datos por medio del teclado es decir, hay que capturarlos. Una vez obtenido el formato, éste se entrega a la dependencia antes mencionada, para la elaboración de los pagos de los trabajadores.

Solución Propuesta

La solución propuesta fue el diseño de un sistema capaz de almacenar la información de los trabajadores, para después ser procesada y generar los reportes de asistencias, faltas, tiempo extra, etc.

Capítulo 1

Generalidades sobre Códigos de Barras

Los sistemas automáticos actuales para efectuar el reconocimiento del trabajador operan básicamente bajo dos tecnologías: cinta magnética y código de barras. Existen otros tipos de tecnologías como OCR (*Optical Character Recognition*) que consiste en reconocer los caracteres que pueden ser identificados visualmente por el hombre pero presenta algunos inconvenientes, ya que tiene una menor eficiencia que el código de barras o la cinta magnética. Los caracteres pueden ser impresos en una hoja de papel al igual que un código de barras, pero es mucho más sensible a la calidad de impresión y al uso que ésta reciba.

Código de Barras y Cinta Magnética

La cinta magnética y el código de barras son similares, por lo que una comparación entre ambos no resulta sencilla. Sin embargo, al tomar en cuenta factores de aplicación y de seguridad, hay diferencias considerables.

Para elaborar una tarjeta magnética se utilizan tres o cuatro capas de PBC laminadas, enseguida va la cinta magnética, con un número de identificación grabado en ella. Finalmente se recubre con una capa obscura. Este último recubrimiento resulta indispensable para proteger la información de la cinta.

Una de las aplicaciones de la cinta magnética está en la telefonía, las tarjetas telefónicas son grabadas magnéticamente con el crédito disponible. Otra de sus aplicaciones está en las tarjetas bancarias, en ésta se graba el número de cuenta del usuario. La razón por la que los bancos empezaron con esta modalidad es porque hace treinta años, sólo la cinta magnética podía ser recubierta por una tira obscura protegiendo así la información. El código de barras en ese entonces no podía ser protegido, resultando muy riesgoso dejarlo al descubierto, pues era muy fácil de reproducir. Actualmente el código de barras se puede recubrir y su aplicación se ha difundido aún más que la de la cinta magnética.

Las aplicaciones de la cinta magnética han disminuido, mientras que las del código de barrras cada vez encuentran mayor aplicación. La cinta magnética ha resultado ser un medio no muy seguro para guardar información ya que es sensible a los campos magnéticos, se ha demostrado que con frecuencia son adulteradas dando lugar a fraudes en los sistemas bancarios, e incluso en los telefónicos. Actualmente las tarjetas magnéticas telefónicas están siendo sustituidas por las tarjetas con *chip* integrado y los bancos por su parte, no tardarán mucho en hacer lo mismo.

Los lectores *SCANNERS* tanto óptico como magnético son parecidos en precio. Actualmente un lector magnético y uno óptico cuestan respectivamente $136.00 y $265.00 dólares estadounidenses. Hasta aquí ambas opciones son muy parecidas. La gran diferencia está en los medios que se utilizan para grabar tanto la cinta magnética como el código de barras. Un grabador magnético cuesta aproximadamente $4,500.00 ó $5,000.00 dólares estadounidenses, mientras para el código de barras, una impresora cuesta alrededor de $1,600.00 dólares estadounidenses. Cabe mencionar que el código de barras puede también ser impreso en la mayoría de las impresoras que se tienen conectadas a una computadora personal, ya sea en la oficina o en la propia casa, independientemente de sus otras utilidades, lo que representa una gran ventaja con respecto al grabador magnético, cuyo uso es muy específico.

Aunque la impresión del código de barras es accesible en muchos de los casos, esto no impide que la confiabilidad en los sistemas de seguridad que lo emplean disminuya, pues como se mencionó anteriormente, puede ser leído si lleva un recubrimiento, ya que la nueva tecnología de *SCANNERS* ópticos usa sensores infrarrojos. Las tarjetas con código de barras generalmente están enmicadas y la parte de la mica que cubre al código se encuentra pintada con una tinta vegetal obscura, de tal manera que el código no es visible para el ojo humano y esto garantiza mayor confiabilidad y seguridad. El tipo de impresión es un factor importante para que el código sea leído correctamente, ya que deben estar perfectamente definidas las barras y los espacios.

Es muy probable que la cinta magnética desaparezca en corto tiempo, pues sus aplicaciones con respecto a las del código de barras representan más desventajas y menor seguridad. Por estas razones el medio de identificación de los trabajadores que se eligió fue el código de barras.

Código de Barras

Historia del código de barras

Wallace Flint, hijo de un vendedor de una tienda de comestibles, realizó en 1932 un sistema de venta automática de productos. Wallance Flint utilizó para este efecto despachadores automáticos y tarjetas perforadas para proporcionar automáticamente productos a los consumidores. El costo de este proyecto no permitió su comercialización. Esta fue la primera vez que se mostraban los beneficios de un sistema automático de reconocimiento del producto (documentadamente). Cuarenta y cinco años después Wallance Flint fue vicepresidente de la Asociación Nacional de Cadenas de Comida, apoyando la estandarización del UPC (*Universal Product Code).*

A fines de los años 40's Joe Woodland y Berny Silver investigaban una forma de leer los precios automáticamente. Sus desarrollos culminaron con una patente en 1949. La patente describe un círculo impreso que asemeja un círculo de tiro al blanco. Este formato se le conoce como *bull's eye code.* Los círculos concéntricos juegan el papel que actualmente tienen las barras y los espacios. Conceptualmente este desarrollo y el principio del código de barras es el mismo. En ese momento la tecnología para desarrollar un código de barras como se conocen actualmente todavía no estaba lista.

A fines de los años 50's y principios de los 60's, se propuso un código estilizado de caracteres que el hombre fuera capaz de reconocer a simple vista y que fueran leídos como un código de barras por los dispositivos lectores. Desafortunadamente este tipo de códigos era mucho más difícil de leer para los dispositivos lectores que un código de barras y presentaba una mayor dificultad al reconocerlo a simple vista, un ejemplo de esto fue el código desarrollado por Girard Feissel constituido por los dígitos del 0 al 9, cada uno formado por siete barras paralelas.

Muchos esfuerzos se hicieron para automatizar los puntos de venta en los supermercados desde 1968. RCA desarrolló una simbología *bull's eye* y un *SCANNER*, que estuvo en operación en una tienda en Cincinnati por un período de 18 meses apartir de 1972. Esta prueba demostró el beneficio en costos, además de permitir el ajuste de estos sistemas. A mitad de 1970 la industria de alimentos formó un comité para seleccionar la simbología y los equipos de punto de venta

para automatizar sus mercados. Esto provocó una evaluación masiva de sistemas que incluyó pruebas de laboratorio hechas por el *Battelle Memorial Institute*, pruebas de tolerancias en la impresión por *The Graphic Arts Technical Foundation*, y pruebas de los sistemas completos en el mercado. Estos esfuerzos concluyeron con la selección del código UPC como el estándar industrial en Abril de 1973. Esta simbología es muy parecida a una propuesta por IBM que se llamó *Delta Distance*.

El éxito de la simbología UPC en los mercados de los Estados Unidos y Canadá animó a industriales extranjeros, principalmente europeos, en adoptar estos sistemas. Esto llevó al desarrollo y adopción del código EAN *European Article Numbering* en diciembre de 1976. Sin embargo, al cabo del tiempo se asociaron a la misma otros países fuera del continente europeo, por lo que el nombre tuvo que ser cambiado al de *International Article Numbering*, aunque sus siglas siguieron siendo EAN. La sede de la organización se encuentra en Bruselas, Bélgica y está concebida como una organización sin fines de lucro y regida por las leyes de ese país.

En México, el código de producto es administrado por la Asociación Mexicana del Código del Producto (AMECOP), la cual ha establecido un sistema dentro del marco determinado por EAN.

Antes de 1974, los códigos de barras sólo podían codificar números y algunos caracteres especiales. En este año el Dr. David C. Allais desarrolló el código 39 que fue la primera simbología comercial alfanumérica en código de barras.

Durante los años 70's el uso del código de barras llegó a ser práctico y económico debido al bajo costo de la electrónica empleada, "microprocesadores" principalmente y el desarrollo de lásers de bajo costo y más pequeños. Muchas compañías generaron sus propias simbologías y equipos de lectura. Esto provocó la aparición de una gran cantidad de simbologías sin tener un estándar aceptado. Algunas simbologías se han estándarizado utilizándose predominantemente en algunas áreas. En el área industrial es muy utilizado el código 39 y el código 2 de 5 intercalado.

Conforme el código de barras fue más accesible en los años 70's, el código UPC proporcionó estabilidad y estimuló la aceptación del uso de estos sistemas y esto mismo llevó a la estandarización de otras simbologías. El estándar militar 1189 (código 39) fue adoptado en enero de 1982. Esto fue seguido por el estándar de ANSI MH10.8M en 1983, el cual cubre los códigos 39, 2 de 5 intercalado y codabar. Al mismo tiempo el grupo de acción de la industria automotríz se estandarizó con el código 39 y desarrolló un formato *well-thought-out* para sus contenedores. Al año siguiente la industria de la salud estableció su estándar HIBC (código 39). Otras industrias como la del papel, aluminio, electrónica, telecomunicaciones y muebles han desarrollado su propio estándar usando el código 39.

Al principio de los 80's hubo esfuerzos por reducir el espacio requerido para el código de barras. En 1981 se dió a conocer el código 128 y en 1982 el código 93. Usando estas simbologías permite reducir hasta un 30% el área necesaria para un código de barras que usara el código 39. El código 128 ha tenido poca aceptación, su uso se ha incrementado en aplicaciones de distribución de productos.

En 1987 el Dr. David C. Allais introdujo el código 49, una simbología no convencional que ofrece grandes ventajas en densidad sobre la mayoría de los códigos convencionales. Un año más tarde Ted Williams introdujo una simbología multi-renglones llamada código 16K. En 1990 *Symbol Tecnologies* anunció la simbología PDF417, multi-renglón que ofrece una mayor densidad.

Algunas de las simbologías que fueron desarrolladas en los inicios del código de barras se muestran en la figura 1.1.

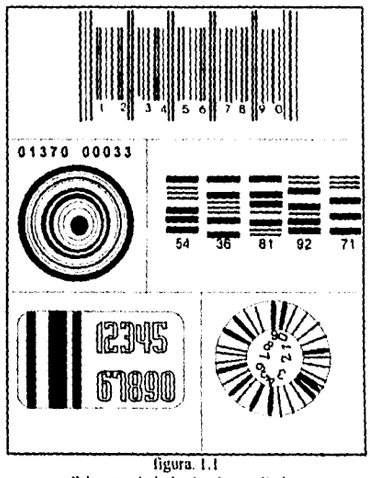


figura. 1.1
Primeras simbologías desarrolladas.

Conceptos Básicos

Un código de barras consiste en una serie de barras y espacios adyacentes paralelos de diferentes anchos que guardan una proporcionalidad entre sí, este conjunto de barras y espacios siguen un grupo de reglas no ambiguas, especificando la manera en que los datos son codificados (figura1.2).

figura 1.2
Mensaje en código de barras simbología 39.

El número máximo de caracteres que pueden ser leídos depende de la capacidad del sistema para almacenar y manejar la información. Generalmente se manejan de 12 a 18 caracteres. Esta restricción se origina a causa de las limitaciones que tienen los lectores en cuanto a la cantidad de información que pueden manipular eficientemente. Para leer la información contenida en el código de barras se requiere de un dispositivo de lectura especial conocido como **lector óptico** ó ***scanner* óptico**.

Los códigos de barras pueden ser impresos horizontalmente denominandose códigos de orientación normal, o verticalmente, a éstos se les denominan códigos de orientación rotada. Siempre que se imprima un código de barras debe imprimirse su equivalencia en caracteres leíbles por el hombre. Esta traducción no puede ser impresa en los lugares adyacentes al inicio o al final del código pero sí arriba o abajo.

Tipos de Simbologías

Las simbologías de códigos de barras caen en dos categorías generales :

- Tipo discreto
- Tipo continuo

El código discreto de cada carácter puede estar solo y puede ser decodificado independientemente de los caracteres adyacentes. Cada carácter es separado del siguiente por un elemento denominado **espacio entre caracteres** el cual no contiene información (figura 1.3).

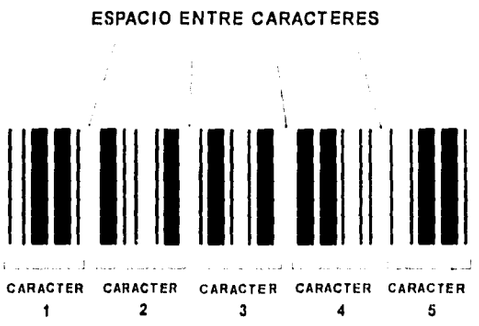


figura 1.3
Simbología de tipo discreto.

Un código continuo no tiene marca entre-caracteres, cada carácter inicia con una barra y termina con un espacio (figura 1.4). Los datos están contenidos en anchos de barras y espacios.

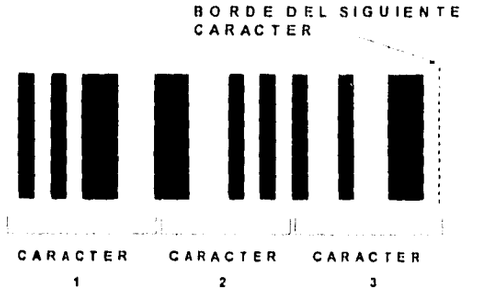


figura 1.4
Simbología de tipo continuo.

Existen dos tipos básicos de códigos de barras, aquellos que emplean sólo el ancho del elemento y aquellos que emplean múltiples anchos.

En simbologías de dos anchos la relación entre los elementos ancho y delgado es denominada N, y se permite variar dentro de un rango generalmente entre 2 y 3, pero debe ser constante para un símbolo dado.

En simbologías de múltiples anchos, barras y espacios pueden tener diferentes grosores. La mayoría de los códigos de múltiples anchos son modulares, esto significa que la longitud de un carácter está subdividida dentro de un número predeterminado de módulos, y el ancho de las barras o espacios es siempre número entero de módulos. Estas simbologías son generalmente contínuas y generalmente decodificadas con algoritmos *edge-similar-edge* de elementos adyacentes más que la medida de los anchos de los elementos.

Algunas simbologías por su estructura sólo pueden codificar mensajes de longitud fija, mientras que otros pueden codificar longitudes variables. De la misma forma hay simbologías que pueden codificar sólo números y se les denomina numéricas y otras simbologías pueden codificar números y letras, a éstas se les denomina alfanuméricas. A continuación se describe la terminología que manejan algunas de las simbologías.

X y Z

X es el término utilizado para decribir el ancho nominal de un símbolo delgado (barra o espacio) en un código de barras, esto con el fin de asegurar la compatibilidad entre equipos de impresión y lectura. Cuando se examina un símbolo desconocido en un código de barras se calcula el promedio de las medidas de los elementos delgados, a esto se le conoce como Z. Por convensión X y Z están expresados en milésimas de pulgada.

Densidad

Los códigos difieren en la cantidad de datos que se pueden codificar en una unidad de longitud dada, esto está relacionado directamente con el ancho del elemento angosto (barra o espacio). Para poder hacer una comparación de densidad es necesario observar que ambos códigos manejen la misma X. Se debe notar que la densidad está dada por los datos. La longitud total debe incluir los caracteres de *Star/Stop*, zonas blancas y caracteres de verificación.

Zona Blanca

Para poder leer un código de barras los diseñadores especificaron una zona a la izquierda y a la derecha del código, que debe estar libre de cualquier impresión, el ancho de esta zona debe ser mínimo diez veces el valor de X. Los códigos de barras deben tener un alto contraste entre los elementos obscuros y los claros. A esta zona también se le conoce como zona estática.

Auto-Verificación

Una simbología es auto-verificable si un error en la impresión no causa que dicho carácter sea interpretado como otro carácter dentro del código.

Carácter de Inicio y Fin (*Start/Stop*)

Un código de inicio *start* es un arreglo particular de barras y espacios que se coloca al principio de un símbolo para indicar al scanner donde comienza el símbolo; generalmente ésto también indica la dirección de lectura del código. Un código de fin *stop*, es un arreglo particular de barras y espacios que se coloca al final del símbolo para marcar el fin de los datos; en ocaciones éste también indica la dirección de lectura de la información.

Carácter de Verificación

Un carácter de verificación es un carácter (o caracteres) colocado(s) en un lugar predeterminndado dentro del mensaje, cuyo valor está dado por una relación matemática basada en los otros caracteres dentro del mensaje. Este es utilizado por el lector de código de barras para validar los datos decodificados. Cuando el caráeter de verificación sólo puede tomar valores etre 0 y 9, se le conoce como dígito de verificación.

Bidireccional

Una simbología bidireccional es quella que se puede leer de deracha a izquierda y de izquierda a derecha sin afectar la decodificación los datos. Actualmente la mayoría de los códigos de barras son bidireccionales.

Auto-generación de Señal de Reloj (*Self-Clock*)

El lector de código de barras necesita información para conocer la medida de la posición relativa de las bordes de todos los elementos o conocer el ancho de los elementos del mensaje dependiendo del algoritmo de decodificación. Algunas de las primeras simbologías incluían una señal de reloj independiente. Las simbologías modernas están diseñadas de tal forma que el

lector de código de barras ya no requiere de una señal separada de reloj para recuperar la información.

Simbologías

Definición de simbología

Simbología es el término usado para describir las reglas no ambiguas, especificando la forma en que los datos son codificados en barras y espacios. Existen una gran variedad de simbologías en el mercado siendo algunas de las más comunes las siguietes:

UPC

El UPC (*Universal Product Code*) ha sido empleado en la industria de supermercados desde 1973. El UPC es un sistema de codificación que está diseñado únicamente para identificar un producto y su fabricante. El código UPC es una simbología continua, numérica y de longitud fija que utiliza cuatro anchos de elemento. Existen tres variaciones de este código, generalmente dos son las más usadas el UPC-A y UPC-D. La simbología UPC-A codifica 12 dígitos (figura 1.5) y UPC-E codifica 6 dígitos (figura 1.6), el tercer tipo es el UPC-D que se utiliza para codificar mensajes de longitud variable.

figura 1.5
Simbología UPC-A

El primer dígito de la versión UPC-A representa el número del sistema indicando el tipo de producto, los siguientes cinco dígitos contienen la identificación del fabricante y los últimos cinco dígitos representan el código del producto. El dígito final es un dígito de verificación.

En una versión del código UPC-A, éste se divide físicamente el símbolo en dos mitades, cada una con seis dígitos y separadas por dos barras centrales, además quedando encerradas por dos barras a la izquierda y dos a la derecha.. Estas barras se pueden tomar como caracteres de inicio y fin. Por esta razón se puede decodificar independientemente la parte izquierda de la derecha.

figura 1.6
Simbología UPC-E

La versión UPC-E codifica seis caracteres y es utilizada para identificar pequeños paquetes, también es conocida como la versión de supresión de ceros que permite a los fabricantes, cuyo código de identificación contiene uno o más ceros, codificar convirtiendo a un código único de seis dígitos, los seis dígitos están entre dos barras a la izquierda y tres a la derecha, éstas son más largas que las de los dígitos, el sexto caráeter es un dígito de verificación. De los seis dígitos tres tienen paridad par y tres tiene paridad impar.

El ancho nominal X empleado en los símbolos UPC es de 13 milésimas de pulgada y el rango de factores de amplificación permisibles es de 0.8 a 2.0.

EAN

El código EAN (*European Article Numbering*) es un super conjunto del UPC. Un sistema lector de código de barras que decodifica el código EAN puede decodificar el código UPC, pero el caso inverso no siempre sucede. El código EAN tiene dos versiones EAN-13 y EAN-8, codificando 13 y 8 dígitos respectivamente (figura 1.7).

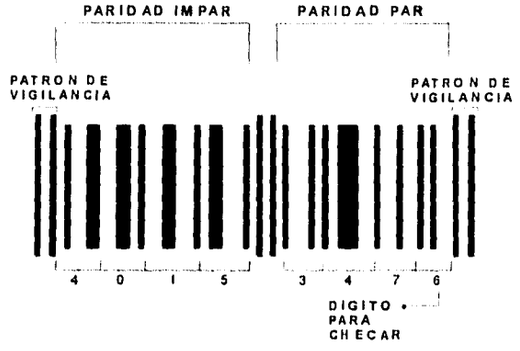


figura 1.7
Simbología EAN-8

En el caso del código EAN-13 los tres primeros dígitos del código EAN-13 representan el código del país, los siguientes cinco contienen la información del fabricante y los siguientes cuatro pertenecen al código del producto y por último el dígito de verificación.

Los dígitos del código EAN-8 tienen dos barras al principio de los datos, los siguietes cuatro dígitos tienen paridad impar, y les siguen dos barras que los separan de un segundo grupo de cuatro caracteres cuya paridad es par, finalizando con dos barras. Esta versión codifica dos dígitos como banderas que especifican el país, cinco dígitos de datos y un dígito de verificación.

ISBN/ISSN

El código ISBN (*International Standar Book Numbering*) no es una simbología de código de barras, pero puede adecuarse a la simbología EAN. La definición del ISBN es la siguiente: se compone de 9 dígitos más uno de control. Los tres primeros dígitos integran el país, los

siguientes pueden ser dos, tres o cuatro dígitos dependiendo del tiraje de la publicación, estos indican el identificador del título, los siguientes dígitos pueden ser cuatro, tres o dos dependiendo de los anteriores y por último el dígito de verificación.

El código ISSN (*International Standar Serial Number*) tampoco es una simbología de código de barras y se puede integrar a la simbología EAN. El ISSN es un código de 8 cifras separadas en dos grupos de cuatro, que identifican un título de una publicación seriada.

Código 2 de 5

El nombre del código deriva de su estructura. Para el caso del código 2 de 5 simplemente significa que por cada símbolo hay cinco barras impresas, de las cuales dos son anchas y tres son angostas. Este código es numérico, la estructura del código está basada en la posición que ocupan las barras anchas y que a cada posición se le asigna un valor de 1, 2, 4, 7 ó P (paridad) como se muestra en la figura 1.8.

fig. 1.8
Representación del número 9 en simbología 2 de 5.

El valor de cada símbolo equivale a la suma de las posiciones que ocupan las dos barras anchas. La P, cuyo valor es cero, se utiliza para llenar los requisitos del sistema, lo que quiere decir que 2 de 5 barras son anchas, por ejemplo en la figura 1.8 el valor de P es cero, ya que existen dos unos en las posiciones de valor 2 y 7 para representar al número 9.

Como no existen dos números de la serie 1, 2, 4, 7 y P que sumen cero ni diez, entonces se utiliza la combinación 4+7=11 para describir el valor de cero. A continuación se muestra una tabla que muestra los valores de las barras dentro del código para cada carácter.

Carácter	Código de Barras				
	1	2	4	7	P
0	0	0	1	1	0
1	1	0	0	0	1
2	0	1	0	0	1
3	1	1	0	0	0
4	0	0	1	0	1
5	1	0	1	0	0
6	0	1	1	0	0
7	0	0	0	1	1
8	1	0	0	1	0
9	0	1	0	1	0

Tabla 1.1
Valores de las barras para la simbología 2 de 5.

La relación existente entre las barras anchas y las angostas varía desde 2.0 hasta 3.0 veces, siendo lo más utilizado que las barras anchas sean 2.5 veces más anchas que las angostas.

Código 2 de 5 Intercalado

El código 2 de 5 intercalado es una simbología numérica, continua de alta densidad y auto-verificable, es empleada principalmente en la industria de la distribución. Cada carácter 2 de 5 intercalado codifica dos dígitos, uno en barras y otro en espacios. Debido a su estructura, los números en este código siempre están en grupos de 2. Tiene 5 barras, de las cuales dos son anchas y tres de ellas son angostas, siendo de la misma forma para los dígitos codificados en espacios , cada dígito tiene un único patrón de 2 de 5 intercalado (tabla 1.2). Por su construcción si se leen partes de un código 2 de 5 intercalado se podrían generar las marcas de inicio y fin, dando una lectura con información errónea, por esta razón este código es más utilizado en aplicaciones que tienen una longitud fija. La ventaja básica del código 2 de 5 intercalado es la reducción del ancho necesario para imprimir cualquier número dado.

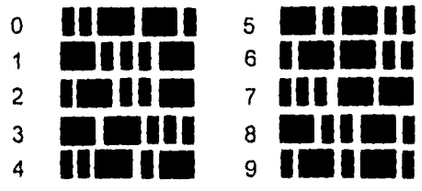


Tabla 1.2
Representación de dígitos en simbología 2 de 5 intercalado.
Cuando se codifica dos dígitos el dígito de posición impar usa la misma representaciónsustituyendo las barras por espacios.

Los dígitos en posición impar son codificados con barras y los dígitos en posición par son codificados con espacios. La estructura completa de un código de barras codificado en la simbología 2 de 5 intercalado incluye el código de inicio (dos barras delgadas y dos espacios delgados), los caracteres de datos y el código de fin (una barra ancha, un espacio delgado y una barra delgada). El número de dígitos a codificar siempre debe ser par.

Código 39

El código 39 fue la primera simbología alfanumérica desarrollada y es ampliamente usada. Es una simbología discreta, de longitud variable y auto-verificable (no requiere de un carácter de verificación pero puede tenerlo). Cada carácter en este código consta de nueve barras de las cuales cinco son barras y cuatro son espacios. El nombre del código deriva del hecho de que de los nueve elementos tres son anchos por esto se denomina código 3 de 9 o código 39. En todos los símbolos, con excepción de $, /, + y % dos de las barras anchas son negras y la otra es blanca (espacio). En el caso de los cuatro símbolos mencionados, las tres barras anchas son blancas. Un mensaje en código 39 comienza y termina con un asterisco, que es el símbolo de inicio/fin (*start/stop*). A continuación se muestra una tabla con el código 39.

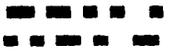
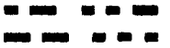
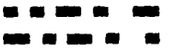
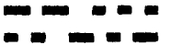
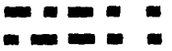
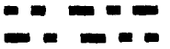
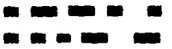
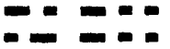
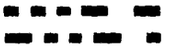
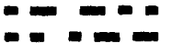
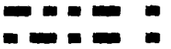
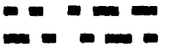
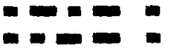
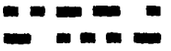
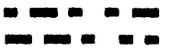
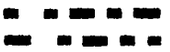
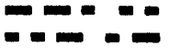
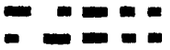
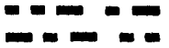
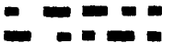
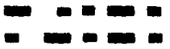
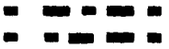
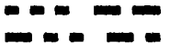
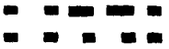
1		M	
2		N	
3		O	
4		P	
5		Q	
6		R	
7		S	
8		T	
9		U	
0		V	
A		W	
B		X	
C		Y	
D		Z	
E		.	
F		Espacio	
G		*	
H		$	
I		/	
J		+	
K		%	
L		—	

Tabla 1.3
Representación de caracteres en simbología 39.

El código 39 es difícil de interpretar visualmente. Los datos binarios o señales generadas correspondientes a cada símbolo están guardados en memoria y su identificación ocurre cuando se compara el dato leído con la tabla almacenada en memoria.

Con el código 39 sólo se pueden representar 43 caracteres diferentes, sin embargo se pueden representar los 128 caracteres del código *ASCII* usando el código 39 *FULL ASCII*, en este los símbolos $, /, % y + son utilizados como códigos de precedencia como se muestra en la tabla 1.4. La decodificación de este código depende del carácter leído, existiendo caracteres de control, los cuales indican su representación en el código *ASCII*.

ASCII	CODE 39	ASCII	CODE 39	ASCII	CODE 39	ASCII	CODE 39
NUL	%U	SP	Space	@	%V		%W
SOH	$A	!	/A	A	A	a	+A
STX	$B	"	/B	B	B	b	+B
ETX	$C	#	/C	C	C	c	+C
EOT	$D	$	/D	D	D	d	+D
ENQ	$E	%	/E	E	E	e	+E
ACK	$F	&	/F	F	F	f	+F
BEL	$G	'	/G	G	G	g	+G
BS	$H	(	/H	H	H	h	+H
HT	$I	)	/I	I	I	i	+I
LF	$J	*	/J	J	J	j	+J
VT	$K	+	/K	K	K	k	+K
FF	$L	,	/L	L	L	l	+L
CR	$M	-	-	M	M	m	+M
SO	$N	.	.	N	N	n	+N
SI	$O	/	/O	O	O	o	+O
DLE	$P	0	0	P	P	p	+P
DC1	$Q	1	1	Q	Q	q	+Q
DC2	$R	2	2	R	R	r	+R
DC3	$S	3	3	S	S	s	+S
DC4	$T	4	4	T	T	t	+T
NAK	$U	5	5	U	U	u	+U
SYN	$V	6	6	V	V	v	+V
ETB	$W	7	7	W	W	w	+W
CAN	$X	8	8	X	X	x	+X
EM	$Y	9	9	Y	Y	y	+Y
SUB	$Z	:	/Z	Z	Z	z	+Z
ESC	%A	;	%F	[	%K	{	%P
FS	%B	<	%G	\	%L	|	%Q
GS	%C	=	%H	]	%M	}	%R
RS	%D	>	%I	^	%N	~	%S
US	%E	?	%J	_	%O	DELL	%T, %X, %Y, %Z

Tabla 1.4
Representación de caracteres en simbología 39 *FULL ASCII*.

La relación existente entre las barras anchas y las angostas es bastante tolerante y permite la posibilidad de imprimirlas de muchas formas; la relación puede variar desde 2.2 y 3.0 veces el ancho de una angosta.

Otras Simbologías

Otras simbologías varían entre sí, éstas pueden codificar conjuntos de caracteres más amplios, con una mayor o menor densidad, versatilidad en la impresión y tolerancias de decodificación. Algunos ejemplos son: Plessey, Norand, Codabar, 128, Telepen, B, MSF, 11, Tekscan, 93, 49, Nixdorf, Delta Distance A, Ames, 16K.

Escogiendo una Simbología

Escoger una simbología en ocasiones es fácil, debido a que muchas veces la aplicación sugiere el tipo de simbología a emplear. Por ejemplo, si un producto va a ser enviado a un supermercado debe utilizarse el código UPC o EAN. Para aplicaciones internacionales que no tienen un estándar definido se deben tomar en consideración los siguietes puntos:

- El área diponible para la etiqueta.
- El tipo de dato a ser codificado.
- El equipo de impresión a utilizarse.
- La disponibilidad en el mercado del equipo de lectura (scanners) e impresión.

Eficiencia de una Simbología

Cuando se está desarrollando o evaluando un código de barras se deben tomar en cuenta las siguietes características:

- Una gran capacidad de representación de caracteres (set de caracteres). Dependiendo de la aplicación ésta puede satisfacerse con códigos numéricos o alfanuméricos.
- Una fácil decodificación con una gran confiabilidad.
- Una decodificación que no utlice equipos de lectura costosos.
- Tener la característica de fácil impresión en cuanto a la tolerancia en la impresión y su estructura física.
- Alta seguridad en los datos, aun cuando la impresión éste en el máximo de la tolerancia permisible.
- Una alta densidad de información.

Desafortunadamente estas características tienden a ser mutuamente excluyentes. Con muy pocas excepciones, aunque la mayoría de las simbologías usadas actualmente cumplen con las cinco primeras.

La información contenida en un mensaje es medida en bits. Si pensamos en un código de barras que tiene un *set* básico, en el cual cada elemento tiene la misma probabilidad, la información contenida en un sólo carácter se define como:

$$I = log_2 C$$

Donde:

I se define como los bits de información por carácter.
C se define como el número de caracteres en el set.

Como un ejemplo, si un mensaje consiste de un sólo carácter que puede tener 16 valores posibles, la información contenida en ese mensaje es de cuatro bits. La tabla 1.5 muestra I en función de los valores de C.

Bits de información como función del tamaño del carácter	
C	**I**
10	3.32 (2 de 5 entrlazado)
26	4.70
43	5.43 (Código 39)
47	5.56 (Código 93)
103	6.69 (Código 128,16K)
2401	11.23 (Código 49)

Tabla 1.5

Por tanto la información para una simbología multicarácter está dada por :

$$I = \log_2 C^N$$

Donde :

C se define como el número de caracteres en el set.
N se define como el número de caracteres en el mensaje.

Seguridad en los Datos

Un buen sistema de código de barras permitirá que los datos sean adquiridos rápida, económica y correctamente. El código de barras puede ser considerado como una tecnología para transmitir información y como tal tiene una razón de error específica. La razón de error específica puede ser muy pequeña si se utiliza una simbología segura, una alta calidad de impresión y además, un equipo de lectura diseñado apropiadamente. En casi todos los sistemas, la razón de error se puede reducir a niveles bajos, si en el mensaje se utilizan caracteres de verificación y redundancia.

Seguridad de una Simbología

Según la calidad de impresión, el ambiente en el que se efectúe la lectura y la forma de hacer la lectura (aceleración del *scanner*, el sistema tiende a leer en forma parcial, etc), darán como resultado simbologías diferentes en razones de error diferentes.

Todas las simbologías son básicamente seguras. Esto significa que el error en los datos puede ser disminuido si el símbolo en el código es impreso cuidadosamente, con las dimensiones nominales exactas y si el proceso de decodificación mide correctamente el ancho de los elementos. Los errores pueden ser causa de la siguiente combinación de elementos:

- El ancho de los elementos no es nominal.
- El reflejo de las barras es excesivo ó el de los espacios es insuficiente, dando lugar a que el lector genere una señal óptica inadecuada.
- La impresión no es nítida debido a la presencia de puntos contrastantes en medio de los espacios o partes borradas de las barras.
- Los sistemas de lectura no miden directamente el ancho de los elementos, en vez de ello, se mide el tiempo que tarda la luz del lector en atravesar al elemento. Debido a la aceleración (especialmente en los *SCANNERS* de mano), el ancho de los elementos que se perciben se deforma.

- Los lectores que se encuentran fijos en un plano se mueven de extremo a extremo del símbolo, empezando y terminando en una zona estática. Las lecturas parciales ocurren a menudo cuando la lectura no se realiza de la forma antes mencionada y muchas veces dan como resultado una información errónea al ser decodificada.

Todas las simbologías están sujetas a la introducción de errores a través de la combinación de algunos de los factores antes mencionados. El grado de resistencia ya sea a uno o a varios de estos errores varía entre simbologías. Se han hecho muchos intentos para calcular la resistencia teórica de de las diferentes simbologías a los efectos antes mencionados. Debido a la impredecible naturaleza de los errores, ésta es una tarea difícil e inconclusa. Uno de los análisis más sencillos de entender se hace calculando el factor de seguridad de la simbología con respecto a los tres tipos de errores de impresión:

1. Dos anchos de elementos diferentes cuando debieran ser iguales.
2. Un elemento es desviadado del ancho nominal.
3. Dentro de las barras no hay uniformidad entre los anchos de los elementos.

Otras posibilidades de error son las que surgen utilizando ciertas velocidades de lectura y que son consecuencia en su mayoría de defectos en la impresión. Este factor de seguridad debe considerar los siguientes puntos:

- Efectos de aceleración del *SCANNER*.
- Errores de resolución del *SCANNER*.
- Cuantización de errores.
- Digitización de errores.

A continuación se presentan los cálculos para los factores de seguridad de algunas simbologías.

Código 39

Uno de los apéndices del USS-39 revela las tolerancias del Código 39. A esta tolerancia se le llama comunmente **Factor R.**

El hecho de que se tengan ya dos errores es motivo de consideración. Citemos los siguientes casos:

- Caso A: representa un elemento delgado malinterpretado como grueso.
- Caso B: un elemento grueso malinterpretado como delgado.

Para el caso A, el algoritmo que se muestra sirve hasta que la tolerancia del ancho de la barra o el espacio excede el valor de:

$$E = \pm\left(\frac{N}{3} - \frac{2}{9}\right)X$$

Donde

E es el error en el ancho de la barra o del espacio.
N es la razón entre ancho y delgado.
X es el ancho de un módulo nominal delgado.

Esta derivación en la tolerancia establece el valor del Factor R =1/8 y asume que un elemento crece a lo ancho en una cantidad E, mientras que el ancho de cualquier carácter se adelgaza en la misma cantidad.

La tolerancia de impresión publicada para el Código 39 es 4/9 de la tolerancia a la cual la falla ocurrirá. El factor de seguridad de 5/9 se reserva para errores en el proceso de lectura, tales como la resolución del *SCANNER* (*substrate scattering*), aceleración del *SCANNER* condiciones de la señal, umbral (*thresholding*) y cuantización.

La aplicación del factor de seguridad permite la publicación de la fórmula de tolerancia para el Código 39:

$$E = \pm\frac{4}{27\left(N - \frac{2}{3}\right)X}$$

Cabe hacer notar que esta aceveración es el peor de los casos, requiriendo dos impresiones de error independientes y opuestas.

Otro caso a examinar es aquel de un solo elemento que se ensancha o se angosta en una cantidad E. Aquí el caso A es el peor de los casos y que cae en error cuando

$$\left(\frac{3N-2}{8}\right)X$$

Comparando este valor al caer en error contra la fórmula de tolerancia de impresión publicada para la del Código 39, se permite un factor de seguridad de 0.605.

Otro caso interesante es aquél de crecimiento o adelgazamiento uniforme de todos los elementos. El caso A es otra vez, el peor de los casos, cayendo en error cuando el crecimiento alcanza un valor de

$$\left(\frac{3N-2}{7}\right)X$$

El factor de seguridad correspondiente es 0.654.

Código 2 de 5 intercalado

Similar al Código 39, el Factor R de tolerancia para caer en error en el Código 2 de 5 intercalado para el peor de los casos es

$$E = \pm\left(\frac{36N-42}{71}\right)X$$

reservando aproximadamente 5/9 (actualmente se utiliza el valor de 0.55625) del total de tolerancia para el lector, se tiene la tolerancia de impresión publicada:

$$t = \pm\left(\frac{18N-21}{80}\right)X$$

Observando el efecto de un error en un solo elemento y que éste ocurre cuando el error del ancho excede

$$E = \pm\left(\frac{18N-21}{32}\right)X \qquad \text{(Caso B)}$$

Comparando esto contra la tolerancia de impresión publicada, es aparente que el 60% del valor total de la tolerancia ha sido atribuída al lector. Examinando el crecimiento uniforme del ancho de una barra, se cae en el error cuando el valor del ancho uniforme excede

$$\left(\frac{36N-42}{57}\right)X \qquad \text{(Caso B)}$$

Comparando esto contra la tolerancia de impresión publicada, 64% del total de la tolerancia está reservada para el lector.

Código UPC

El Código UPC se decodifica usando algoritmos *edge-to-similar-edge*. El ancho total **P** del carácter es medido y las dos distancias **T** son cuantizadas dentro de anchos modulares, asumiendo que P es ancho 7 módulos.

Se definen tres tolerancias:

e = tolerancia de *edge-to-similar-edge*
p = tolerancia extrema (ancho del carácter)
b = tolerancia del ancho de la barra o del espacio

Los valores publicados para **e** y **p** son:

e = 0.14692X
p = 0.29X

Considérese el caso donde se pretende que una distancia **T** sea 5 módulos y se mida en forma corta en una cantidad **e** y, donde la longitud del carácter se mida en forma larga en una cantidad **p**. La distancia modular **T** percibida será

$$7\left(\frac{5-e}{7+p}\right) \quad \text{ó} \quad 7\left(\frac{5\ \ 0.14692}{7+0.29}\right)$$

ó 4.66 módulos.

Este error de 0.34 módulos (5.00-4.66) representa el 68% del error permisible antes de caer en el error definitivo.

Para la mayoría de los caracteres, la decodificación del UPC puede llevarse a cabo con las técnicas *edge-to-similar-edge*, y teóricamente no hay necesidad de especificar en forma cerrada, la tolerancia actual en el ancho de la barra o del espacio.

Los factores de seguridad referidos anteriormente no deberían ser utilizados para evaluar la seguridad relativa de las diferentes simbologías; otros factores tales como el *meter* códigos o redundancias tiene mejores efectos también.

AIM, entre 1986 y 1987 fomentó una prueba de tecnología para siete diferentes simbologías. La prueba la condujo la Universidad Estatal de Nueva York (SUNY) en Strony Brook. Una gran variedad de tecnologías de impresión y ventas fueron usadas para generar una gran variedad de símbolos que contenían un carácter de verificación, éste es un carácter adicional y que se calcula matemáticamente, se basa en la codificación de los caracteres contenidos en el mensaje. Ocho lectores diferentes se utillizaron para la prueba, todos conectados a una computadora que examinaba el proceso para checar el carácter. Aproximadamente tres millones de caracteres fueron leídos en cada una de las siguientes simbologías:

- **Código 39**
- **2 de 5 intercalado**
- **UPC-A**
- **UPC-E**
- **Cadabar**
- **Code 128**
- **Code 93**

Todos los símbolos tenían 12 caracteres de longitud, excepto el UPC-E que contenía seis dígitos. Debido a la naturaleza de la prueba, lecturas cortas en el 2 de 5 intercalado o errores de autodiscriminación con el Código 39 fueron excluídos. Todos los lectores fueron puestos para autodiscriminar tantas simbologías como fuera posible. Caracteres de checado opcionales no fueron usados para el Código 39, 2 de 5 intercalado o Codabar. Las otras cuatro simbologías incluyeron sus correspondientes caracteres de checado.

La prueba indicó que las simbologías del UPC-E y del Codabar tienen una considerable razón de error, más alta que la de las otras, y que el Código 128 y el Código 93 demostraron la más alta seguridad.

Se observaron muchos errores de autodiscriminación . En cada caso, una simbología era malinterpretada por el lector como un UPC-E válido, o corto (uno o dos caracteres) del símbolo en Codabar. Los lectores difieren bastante en su sueceptibilidad a este tipo de error.

Posteriormente, basado en la examinación de los errores observados, se determinó que si el Código 39, 2 de 5 intercalado y el Codabar hubieran utilizado un carácter de checado (que sería examinado por el lector), la seguridad en ellos hubiera resultado ser más grande.

En 1990, la Universidad Estatal de Ohio emprendió la construcción de una simbología para facilitar las pruebas, la cual fue fomentada por AIM y por el Consulado de Asuntos en Comunicaciones de la Industria de la Salud (HIBCC), y es inicialmente designado a examinar específicamente la realización del Código 49 y del Código 16K dentro del ambiente para el cuidado de la salud.

Calidad de Impresión

El funcionamiento de un sistema de código de barras depende en gran medida de la calidad de impresión. Cuando se hace una lectura de un código de barras pueden ocurrir tres cosas:

1. El lector decofifica la información correctamente.
2. El lector no decodifica por existir un error en la lectura.
3. El lector decodifica pero la información no es la misma que la codificada en la etiqueta.

A la primera posibilidad se le conoce como razón de primera lectura (*first read rate FRR*). Cuando la calidad de impresión es buena y los equipos son utilizados por operadores expertos, el FRR debe exceder el 90%. Una impresión de baja calidad debe decrementar el FRR.

Existe una fuerte relación entre el FRR y la razón de error de substitución (*substitution error rate SER*), el algoritmo implementado en los sistemas de lectura puede afectar los valores obtenidos. Esto quiere decir que pueden existir equipos con algoritmos capaces de interpretar cualquier código de barras en la primera lectura , teniendo un 100% de FRR, pero tendría un muy alto SER.

La probabilidad de error de substitución depende mucho de la calidad del símbolo y con ello de la razón de primera lectura. Para el Código 39 sin un carácter de verificación, la probabilidad de error ha sido calculada teóricamente y confirmada experimentalmente de acuerdo con la siguiente relación mostrada en la figura 1.9, nótese que la figura representa la relación del Código 39 para longitudes de símbolo de 20 caracteres.

La figura indica que hay una relación extremadamente fuerte entre la razón de la primera lectura y la de error de substitución. Este error es causado en gran parte por la calidad de la impresión. Entre más alto sea el porcentaje de error de la primer lectura, mayor aún será el incremento en los errores de substitución. Por lo tanto es recomendable que se mantenga una calidad alta en la impresión del símbolo aunque la razón de primeras lecturas no sea del 100%. Otras simbologías muestran una relación semejante aunque los números cambian.

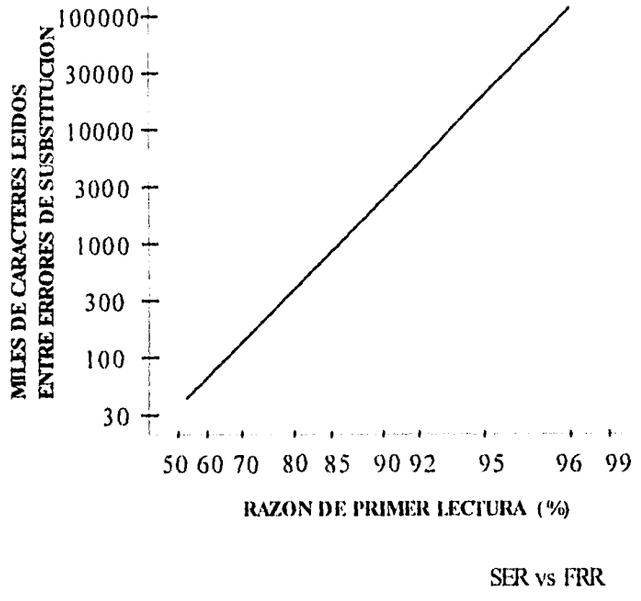


fig 1.9
Gráfica de error de substitución (SER) contra la razón de primeras lecturas (FRR) para simbología 39.

Estándares en Códigos de Barras

Muchos sistemas de adquisición de datos requieren estándares en los códigos de barras. Existen tres tipos de estándares: simbología, aplicación y calidad de impresión.

Simbología Estándar

Una simbología estándar especifica la estructura del código de barras, describe la forma en que la información es codificada dentro de un arreglo particular de barras y espacios con anchos específicos, haciendo así un símbolo en código de barras. Además de describir la forma de codificación de la información, una simbología estándar define otros elementos como son:

Tolerancias de Impresión

Las tolerancias de impresión describen cómo se deben controlar en el proceso de impresión los anchos en barras y espacios. Si las toleracias son excedidas, la razón de lecturas en el primer intento se variará adversamente y la razón de error de substitución se incrementará dramáticamente. Las tolerancias de impresión fueron puestas tomando en cuenta los algoritmos de decodificación y el equipo de lectura.

Las tolerancias de impresión varían en función directa de X. En simbologías de dos anchos, las tolerancias de impresión están también en función de N.

Propiedades Opticas

Debido a que un código de barras es un tecnología óptica, deben observarse ciertas propiedades. El estándar de un simbología de código de barras debe especificar el mínimo permisible de reflectividad de los espacios en un símbolo (directa o indirectamente) así como el máximo permisible de reflictividad en las barras. Todas las mediciones ópticas se realizan a una misma longitud de onda específica, utilizando una geometría particular de medición.

Puntos, Huecos y Filos Rugosos

Defectos de impresión pueden afectar la habilidad de un equipo de lectura en la decodificación de un símbolo. Los puntos se presentan en los espacios de un símbolo. Los huecos se presentan en las barras de un símbolo y los filos rugosos se refieren a la transición de una barra a un espacio (fig 1.10).

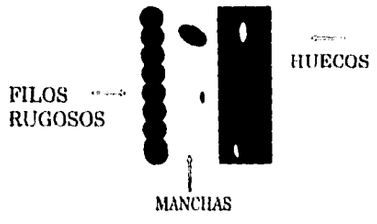


Fig 1.10
Defectos de impresión en códigos de barras.

Lectura Optica

El principio de la lectura óptica es que al iluminar una superficie, si ésta es clara, reflejará la mayor parte de la luz que recibe. Una superficie obscura absorberá la mayor parte de la luz que recibe. La diferencia entre la luz reflejada y la absorbida es llamada CONTRASTE.

Cuando un patrón de barras separadas por espacios pasa por una fuente de luz, la intensidad de la luz reflejada corresponde al ancho de las barras y espacios. La mayoría de los lectores ópticos trabajan con este principio, detectando la luz reflejada por medio de un receptor fotoeléctrico. Una vez que el patrón de barras y espacios es capturado, el patrón puede ser decodificado obteniendo la información original contenida en el código de barras.

Equipos de Lectura

Un lector de código de barras es un elemento capaz de extraer la información que está ópticamente codificada en un símbolo de código de barras convirtiéndola en datos digitales, los cuales pueden ser almacenados en el mismo lector, transferirse a una computadora o intereactuar con un programa residente en el decodificador dependiendo de la aplicación (figura. 1.11).

Para poder obtener la información codificada en un código de barras, un lector de código de barras debe ejecutar siete funciones básicas.

1. Encontrar los elementos que corresponden a un mensaje en código de barras.

2. Determinar el ancho de cada uno de los elementos del código (barras y spacios). Este paso se realiza con un *scanner* óptico en combinación con un programa que se encuantra en el decodificador.

3. Clasificar los elementos del código, dependiendo de la simbología utilizada (Dos para los códigos 39, 2 de 5, etc.; 5 para el código 128).

4. Verificar que la clasificación de los elementos sea consistente con las reglas marcadas por el código de barras.

5. Si es necesario, voltear el orden de los datos. La dirección de lectura se determina examinando los caracteres de inicio/fin.

6. Confirmar las zonas blancas en los extremos del código.

7. Confirmar que los caracteres de verificación sean consistentes con el mensaje.

Se puede considerar a un lector de código de barras como dos elementos separados. El primer elemento es un dispositivo de entrada, el cual emplea técnicas óptico-electrónicas para leer el símbolo en código de barras (*scanner* o lector óptico) y el segundo elemento es un decodificador.

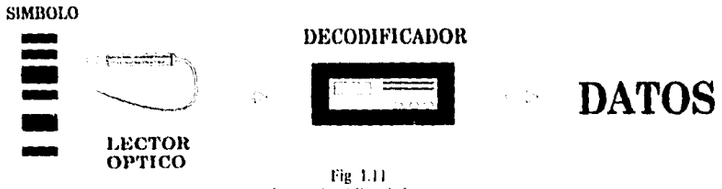


Fig 1.11
Lector de código de barras

Un dispositivo de entrada es generalmente un sistema activo. Este sistema ilumina el símbolo, examinando el monto de luz que es reflejada por un área especifica del símbolo. Los espacios deben reflejar más luz que las barras. A el área examinada por el lector óptico se le llama *spot*. Las dimensiones del *spot* en un eje perpendicular a un eje de las barras deben ser consistentes con el ancho del elemento más delgado.

El *spot* es formado en el lector óptico de dos formas: midiendo la luz reflejada por un haz estrecho (fig 1.12) en ésta la resolución es controlada por la regulación de la luz a través de la apertura, o iluminando el símbolo y midiendo la luz reflejada a través de una apertura estrecha (fig 1.13), en este la resolución es controlada por la regulación del flujo de luz que llega al fotodetector.

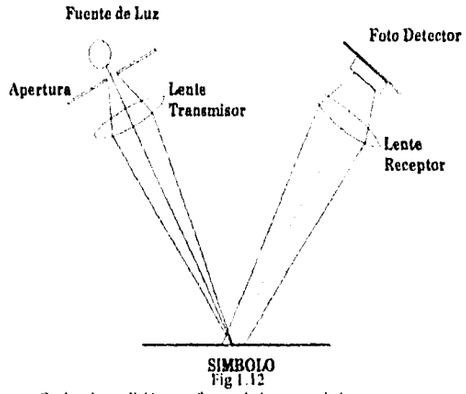


Fig 1.12
Optica de medición con fuente de luz controlada por apertura

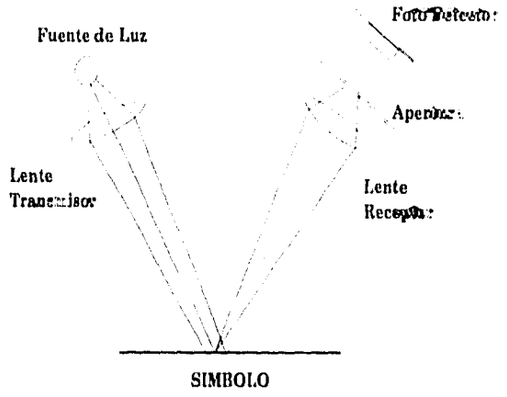


Fig 1.13
Optica de medición a través de una apertura.

La luz reflejada por el *spot* es dirigida a un foto-diodo (o un dispositivo equivalente), el cual genera una corriente que es proporcional a la cantidad de luz reflejada. Un amplificador en el dispositivo de entrada eleva esta señal a un nivel en el cual permita el manejo de la misma. Conforme el *spot* se mueve en forma perpendicular al eje mayor de las barras (a esta operación se le conoce como *scanning*) el voltaje del amplificador varía. En este punto el voltaje analógico es proporcional a la reflectividad de los *spots*. Este voltaje es convertido a un voltaje digital para tener una mejor diferenciación entre barras y espacios. El voltaje final en el dispositivo de entrada puede ser digital y/o analógico , en el caso de que sea un voltaje analógico el decodificador deberá tener un circuito que transforme esta señal a un voltaje digital. En la figura 1.14 se muestran las formas de onda de un lector óptico al leer un código de barras.

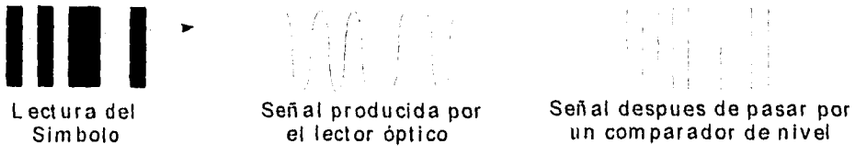


Fig 1.14
Formas de onda al leer un código de barras.

Cuando se lee un código de barras con una velocidad constante, es fácil determinar la duración en tiempo de las barras y espacios del código. Cuando la velocidad con la que se lee el

código de barras no es uniforme, es mucho más difícil de determinar el ancho de los elementos examinanado la señal eléctrica, teniendo aquí la importancia de los algoritmos de decodificación.

Muchas longitudes de onda se han utilizado en las fuentes de luz de los sistemas de lectura de códigos de barras, estas longitudes de onda han estado generalmente dentro del espectro visible al ojo humano que va de 750 nanómetros a 450 nanómetros.

Los sistemas recientemente desarrollados utilizan una longitud de onda de 900 nanómetros, algunos de los primeros sistemas operan con una longitud de onda de 600 nanómetros, que es la longitud de onda emitida por las lámparas de helio-neón (HeNe).

Las dos longitudes de onda más utilizadas son las de 900 y 633 nanómetros, la longitud de onda de 900 nanómetros al no ser visible por el ojo humano se utiliza en ambientes donde los códigos de barras se pueden manchar con grasa, sangre, etc. además esto ha originado que se utilizen en algunas aplicaciones de seguridad.

Al elemento de lectura se le conoce como "*scanner* de código de barras", éstos pueden clasificarse dependiendo de su mecanismo de funcionamiento y por su construcción como se muestra en la tabla 1.6:

	haz fijo	haz móvil
Fijo	Sin contacto	Sin contacto
Movil	Con y Sin contacto	Sin contacto

Tabla 1.6
Clasificación de lectores ópticos

Profundidad de campo

Todos los dispositivos de entrada que no tienen un contacto físico con el código tienen un rango de distancia dentro del cual pueden interpretar correctamente la información, a este rango de distancias se le conoce como profundidad de campo (DOF).

La profundidad de campo está en función del valor X. Grandes valores de X dan como resultado una gran profundidad de campo. Algunos de los tipos de "scanners" disponibles en el mercado se mencionan a continuación.

- Lector de tipo Lápiz.
- Lector de tipo "Slot".
- Lector de tipo CCD
- Lector de tipo pistola Láser
- Lector de láser empotrable

Decodificadores

Los decodificadores son el segundo elemento de un sistema lector de código de barras. Estos se encargan de examinar la señal producida por los dispositivos de entrada, descifrando el mensaje codificado en ella, para posteriormente transmitir, almacenar o utilizar esta información dependiendo de la aplicación.

Todos los sistemas de lectura de códigos de barras incluyen un módulo especial que decodifica la información, generalmente este módulo es un programa que se ejecuta en un microprocesador, aunque también puede ser implementado con otro tipo de lógica. Todos los decodificadores realizan los siguientes pasos.

1. De la señal del dispositivo de entrada determina qué parte representa una barra o un espacio.

2. Mide los anchos de cada uno de los elementos de la señal, entregada por el dispositivo de entrada.

3. Clasifica los elementos dependiendo del código de barras. Para códigos de dos anchos, los clasifica en gruesos o delgados. Para código de multiples anchos los clasificará según las reglas de cada código.

4. Decodifica la información comparando la información obtenida de la señal entregada por el dispositivo de entrada con una tabla que contiene todos los caracteres del código empleado. En estas tablas se maneja la posibilidad de que se puede leer un código de deracha a izquierda o de izquierda a derecha, para este efecto habrá que incluir en la tabla los caracteres leídos en ambas direcciones.

5. Si es necesario, debe invertir el orden de los caracteres para tratar de decodificar los caracteres que han sido leídos de derecha a izquierda. La dirección de lectrura es generalmente definida al examinar los caracteres de inicio/fin. Si en una lectura se cambia el sentido de la misma la decodificación se aborta.

6. Confirma que la información decodificada es la correcta, este paso lo puede ejecutar realizando algunas de las siguientes acciones.

   - Confirmado la validez de las zonas blancas.
   - Checando los caracteres de verificación.
   - Verificando la velocidad de lectura de la información. Esta debe ser menor a la permisible por el dispositivo de entrada.

- Verificar la longitud del mensaje, marcas entre caracteres u otros aspectos que aseguren la validez de la información.

7. Transmitir la información decodificada al siguiente elemento del sistema, para que haga uso de ella dependiendo de la aplicación.

Rendimiento

El rendimiento de un sistema de código de barras se evalúa en términos de dos parámetros: el de razón de lectura y el de razón de error de substitución.

El parámetro de razón de primera lectura es definido como el número de lecturas con error entre el número de lecturas sin error. Un buen sistema debe ofrecer una razón de primera lectura aproximada del 80%; una razón de primera lectura baja es causada generalmente por problemas de impresión o bien por problemas de operación en el que la persona que maneja el lector no lo posiciona correctamente para que se puedan leer los datos codificados.

También existe la posibilidad de que la baja en la razón de primera lectura se deba a que el lector esté sucio, y sólo bastará con limpiarlo, en el peor de los casos la razón de lectura baja se puede deber a imperfecciones del dispositivo lector.

El parámetro de error de substitución ocurre cuando todos los caracteres codificados son leídos e interpretados como si fuesen otros completamente distintos. La razón de error de substitución depende directamente de la estructura del patrón del código de barras adoptado, de la calidad de impresión y del algoritmo de decodificación que se programe.

Si el sistema de adquisición de datos no cuenta con alguna forma de tratamiento de estos datos corrompidos, entonces aceptará un dato inválido.

El código 39 es quizá el que posee la más alta razón de error de substitución de los códigos comerciales, ya que por su amplia gama de caracteres resulta más complicado de manejar. Un buen sistema de código de barras debe ofrecer una razón de error de substitución de alrededor de un carácter mal interpretado por cada millón de caracteres leídos.

Impresión de Códigos de Barras

El código de barras puede reproducirse con diferentes técnicas, cada una de ellas con diferentes operaciones y aplicaciones muy particulares, pudiéndose clasificar en dos grupos:

- Técnicas de impresión externa.
- Técnicas de impresión interna.

En las técnicas externas se requiere de un elemento conocido como película maestra (ejemplo de positivo y negativo) que puede ser elaborada con medios técnicos que permiten aceptar tolerancias. En el caso de la impresión ésta produce inevitablemente variaciones en la ganancia inherentes a cada técnica de impresión, materiales de aplicación (papel, cartón, plástico u otros materiales) y las tintas usadas en la impresión. Dentro de las impresiónes externas se consideran las siguientes:

- Impresión en tipografía.
- Impresión en offset.
- Impresión en rotograbado.
- Impresión en flexografía.
- Impresión en serigrafía.

Las técnicas de impresión interna son aquellas en las que el usuario puede realizar con la ayuda de sistemas de cómputo y equipo, impresiones sobre etiquetas, papel especial, etiquetas autoadheribles, etc.

De esta forma se tienen las bases para poder desarrollar un sistema de código de barras, así como, de los elementos necesarios para la evaluación de un sistema de código de barras. En el siguiente capítulo se describirán los elementos que integran al sistema.

Capítulo 2

Elementos del Sistema

El presente capítulo tiene por objeto el mostrar las características tanto físicas como cualitativas de los componentes más importantes del sistema, lo que justificará su elección y que son parte de los sistemas digitales.

Para aquellos lectores que no estén muy familiarizados con el término, y con el fin de lograr una mejor comprensión de éste capítulo y del siguiente, se hará una introducción breve de los sistemas digitales actuales, los cuales ofrecen mayores ventajas con respecto a los circuitos desarrollados con componentes discretos: tienen bajo costo, ocupan espacios reducidos, consumen poca potencia, ofrecen alta confiabilidad, trabajan a altas velocidades y reducen el número de conexiones externas.

Circuitos Integrados

Los sistemas digitales se componen de circuitos integrados. Un circuito integrado IC (*Integrated Circuit*)es un cristal semiconductor de silicio, llamado pastilla (*chip*) que contiene

componentes eléctricos tales como transistores, diodos, resistencias, capacitores e incluso puede contener a otros circuitos integrados. Internamente son conectados formando un circuito electrónico y que se conecta al exterior por medio de sus terminales (*pin*).

Los circuitos integrados se encuentran inmersos en una cápsula de plástico o cerámica y no se puede tener acceso a ninguno de sus componentes. Los tamaños están normalizados así como el número de terminales que puede ir generalmente de 2 a 68. Las dimensiones por ejemplo, para una compuerta lógica AND (un circuito que cumple con la función lógica de conjunción), son de 20x8x3 milímetros y tiene 14 terminales. Un microprocesador puede medir 50x15x4 milímetros con 40 terminales. La figura 2.1 ilustra un modelo de "chip".

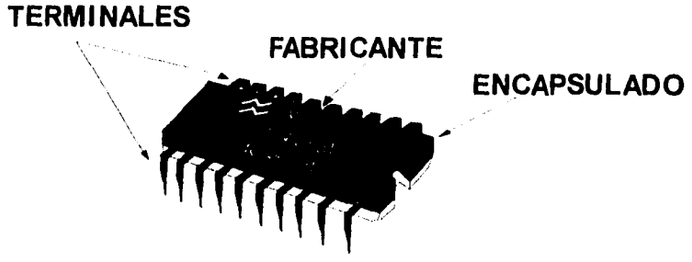


Figura 2.1
La figura muestra la apariencia física de un CI, señalando su número de fabricante, las terminales metálicas y el tipo de encapsulado, características más comunes de estos circuitos.

Los circuitos integrados se clasifican en **analógicos** y **digitales**. Los primeros son los que operan con señales analógicas. Una señal analógica es aquella que tiene un valor para **cualquier instante** del tiempo. Se les denomina así porque su magnitud en el tiempo es **análoga** al comportamiento de la señal física que representan. Algunos ejemplos de los circuitos analógicos son los amplificadores operacionales y los comparadores. Una señal binaria es una señal digital, esta última resulta de una serie de valores tomados en un **instante determinado** del tiempo. Los circuitos digitales operan con señales binarias y están formados de compuertas lógicas digitales.

Según el número de compuertas que puedan encapsularse en una pastilla se hace la referencia del tipo de integración del elemento: a pequeña escala SSI, a mediana escala MSI, a grande escala LSI y a muy grande escala VLSI.

Familias Lógicas

Las compuertas digitales a parte de ser clasificadas por su operación lógica son también clasificadas por la tecnología con la que son fabricados, esta tecnología recibe el nombre de **familia**. Las familias más comunes son:

TTL	Lógica de transistores (Transistor-Transistor- Logic)
ECL	Lógica de Acoplamiento de Emisor (Emitter-Coupled-Logic)
CMOS	Semiconductor de Oxido de Metal Complementario (Complementary-Metal-Oxide-Semiconductor)

Los circuitos básicos de cada familia se evalúan y se comparan de acuerdo a ciertos parámetros. Cada familia tiene características favorables según el tipo de diseño digital que se vaya a construir. Se tienen familias muy rápidas que consumen mucha potencia como la TTL, por ejemplo. También el caso contrario, las que consumen muy poca potencia pero se vuelven más lentas como las CMOS, estas últimas se caracterizan además por ser circuitos de alta densidad

Selección de Elementos

El sistema de control automático de checado de tarjetas para el personal de la Facultad de Ingeniería se divide en dos bloques, en el diseño se tomaron en cuenta las necesidades de la **Facultad de Ingeniería**: El primer bloque se encarga de identificar y almacenar la información referente a cada trabajador. El segundo bloque se encarga procesar la información y generar los reportes de asistencias, faltas, tiempos extras, etc. En la siguiente figura se muestran estos bloques.

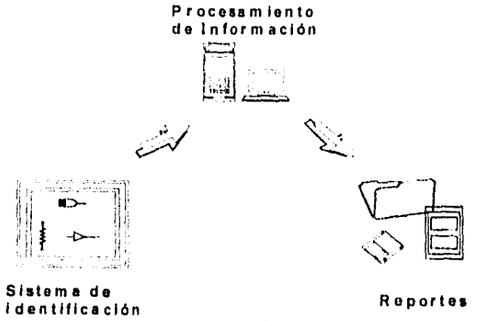


Figura 2.2
Concepción del sistema.

Algunas de las restricciones de diseño fueron proporcionadas por la misma Facultad, al tener que cumplir con algunos requisitos legales, como:

- No desaparaecer la tarjeta de asistencia actual, puesto que ésta es un comprobante legal y sirve para hacer cualquier aclaración.
- No desplazar el reloj checador mecánico que actualmente se utiliza.
- Se puede modificar el tamaño de las tarjetas actuales de checado (hacerlas más largas).

Por estas razones se desarrollará un sistema de indentificación que sea casi independiente del reloj checador actual. Se acoplará a éste un lector óptico en la entrada de las tarjetas de asistencias, aprovechando el movimiento de la tarjeta para leer el el R.F.C. del trabajador (código de barras) y de este modo identificar al trabajador. A continuación se describirán los elementos que componen al primer bloque del sistema (bloque de indentificación y almacenamiento).

Bloque de Identificación y Almacenamiento

Este bloque debe ser capaz de realizar las siguientes acciones:

- Decodificar el R.F.C. de los trabajadores (código 39).
- Almacenar la información referente a cada trabajador.
- Tener la capacidad de interactuar con una computadora personal para procesar la información almacenada.

Para la realización de este bloque se pensó en un microcontrolador debido a que para efectuar la decodificación de un código de barras es necesario elaborar un programa, el cual procese la señal entregada por el lector óptico. Estos dispositivos ofrecen generalmente un puerto serie integrado, el cual permite la comúnicación con una PC, así como, puertos de entrada-salida con los cuales se puede controlar periféricos, como son los dispositivos de almacenamiento (ésta es una de las razones por la que se prefirió un microcontrolador y no un microprocesador, un microcontrolador ofrece en un mismo encapsulado un microporcesador además de puertos de entrada/salida).

A continuación se hace una descripción básica de los principales elementos que componen el sistema y se presentan además las características técnicas de los elementos seleccionados.

Memoria de Sólo Lectura

Una **ROM** (Read Only Memory) es esceneialmente un dispositivo el cual almacena un conjunto fijo de información binaria. Dicha información se especifica por el usuario y luego se graba en la unidad. La unidad tiene internamente enlaces que se abren o se cierran al grabarla. La característica más importante de esta memoria es que los enlaces una vez hechos, quedan fijos (dependiendo del tipo de memoria ROM se pueden modificar) y no se borrarán aún cuando no circule corriente por el circuito.

En general, una memoria ROM consiste en **n** líneas de entrada y **m** líneas de salida. Cada combinación de bits de las variables de entrada se llama **dirección**. Una dirección es esencialmente el número de la localidad en el que se desean almacenar los **n** bits. Cada combinación de bits que sale por las líneas de salida recibe un nombre dependiendo del número de bits que lo forman: byte, word, doble word. El número de bits por byte, word o doble word es igual al número de líneas de salida **m**. La localidad está formada por varias celdas y a todo el conjunto se le conoce como REGISTRO. La capacidad de los registros varia entre 8, 16 ó 32 bits según el tipo de memoria. El número de direcciones posible que puede haber es de 2^n, por lo que también se puede decir que hay 2^n bytes diferentes que se pueden almacenar en el dispositivo. Así, si se desea leer un byte en específico, se debe dar la dirección en donde dicho byte está almacenado. La figura 2.3 representa cómo se distribuye la información dentro de una memoria. A esta distribución también se le conoce como Mapa de Memoria.

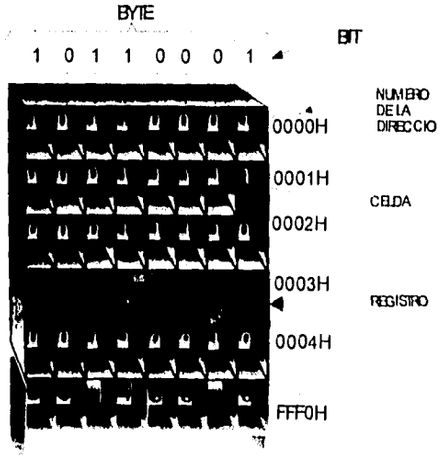


Figura 2.3
Distribución de la información en una memoria.

La memoria comercial empleada en el proyecto es la NMC27C64 de NATIONAL SEMICONDUCTOR y construída con tecnología CMOS. La capacidad de almacenamiento de esta memoria es de 64 Kbytes con 13 líneas de entrada para direcciones y 8 líneas de salida para datos. La potencia que consume es muy baja y trabaja a alta velocidad. Opera con un voltaje de alimentación de 5 Volts. Otras de sus características se muestran en la tabla.

NMC27C64 (8K x 8) CMOS EPROM
Tiempo de acceso: 150 ns
Potencia en activo: 55 mW max Potencia en estado de espera: 0.55 mW max
Estructura compatible con microprocesadores CMOS
Temperatura de operación: de -40°C a 85° C
Programación rápida y segura
Operación estática, no necesita reloj
Entradas y Salidas compatibles con TTL y CMOS

Tabla 2.1
La tabla muestra los datos técnicos que da el fabricante a cerca de esta memoria tipo EPROM.

La tabla muetra los tiempos de acceso y las características principales de la memoria NMC27C64. Estas características permiten tener una buena interacción entre ésta y el microcontrolador ó microprocesador. La capacidad de esta memoria y su velocidad son ideales para este proyecto, ya que 8 líneas de datos son suficientes y además, el espacio en memoria que ocupan el programa para el microprocesador junto con las tablas de código ASCII y de código 39 no requieren de mayor tamaño.

Memoria de Acceso Aleatorio

Las memorias **RAM** (Random Access Memory) funcionan en una forma similar a como lo hacen las memorias **ROM**, salvo que la información almacenada en ellas no es fija. Es necesario que la memoria esté constantemente polarizada pues de lo contrario se pierde toda la información.

Este tipo de memorias son muy útiles porque se utilizan generalmente para almacenar valores variables dentro de la operación del sistema. Dentro de un registro el byte almacenado puede ser reemplazado por uno nuevo. De igual forma, son dispositivos de entrada/salida, debido a que pueden ser escritos o leídos, se manejan por medio de líneas de datos y direcciones.

La memoria MCM6264 de MOTOROLA SEMICONDUCTOR fue la que se eligió para el sistema. Sus especificaciones son muy parecidas a las de la memoria ROM y se muestran en la siguiente tabla:

MCM6264 (8K x 8) CMOS RAM ESTATICA
Voltaje de alimentación: 5 Volts
No requiere reloj
Tiempo de acceso : 45 ns (máximo)
Consumo de corriente: 90 mA (máximo)
Salidas tipo TREE STATE
Entradas y Salidas compatibles con TTL
Terminal para activar o desactivar toda la operación del CI

Tabla 2.2

La tabla muetra las características de la RAM donde se puede ver que la velocidad de ésta es mayor a la de la ROM, pero sus características de potencia son muy similares. El hecho de que sean compatibles con TTL, permite tener en el sistema en su conjunto, combinaciones de ambas familias de circuitos integrados.

El micro-Controlador

Un sistema digital está definido por los registros que contiene y por las operaciones que puede efectuar con la información almacenada en ellos. El número de las distintas operaciones que se pueden realizar en un sistema dado es finito y se dice que la complejidad del diseño depende de la secuencia de operaciones para **procesar** los datos, lo que implica tener funciones de control y el desarrollar un programa. Un **microprocesador** **µP** es un circuito integrado con las características antes mencionadas. El manejo de las operaciones se controla por medio de un programa que cumple con las restricciones del diseñador del sistema digital. El **µP** va siempre acompañado de periféricos (HARDWARE) con los que interactúa para llevar a cabo las órdenes del programa (SOFTWARE).

Se le llama microcontrolador (**µC**) al circuito integrado que además de efectuar las operaciones de un microprocesador cuenta con algunos periféricos internamente instalados que simplifican mucho el diseño de un sistema digital. Dichos componentes internos son por lo general, memorias tipo RAM y ROM, contadores, temporizadores, convertidores A/D, etc. pueden ser expandidos con ROM, RAM externas, puertos y periféricos, para producir una aplicación de control más poderosa.

El programa del microprocesador es inalterable y por esta razón puede ser almacenado en una memoria de sólo lectura, así como las tablas y los valores que deben permanecer constantes. Lo anterior causa que el microprocesador se comporte de una manera preestablecida.

La comunicación entre el μP y sus periféricos se lleva a cabo vía las líneas (*bus*) de direcciones y el de datos, además de las líneas de control. El *bus* de direcciones se usa para accesar a las diferentes localidades de memoria RAM o ROM. Las localidades se encuentran distribuidas en un mapa de memoria, de manera que al asignar cualquier número de dirección, el μP se dirigirá a la memoria ROM, RAM o a cualquier periférico manejado por direcciones, obedeciendo la distribución del mapa de memoria. El número de líneas disponible en el *bus* de direcciones determina el tamaño máximo de memoria.

Selección del microcontrolador

Actualmente en el mercado existe una gran variedad de microcontroladores, de las cuales destacan tres familias principales: Intel, Motorola y Zilog, este último desarrolla sus procesadores para aplicaciones de control.

Existen microprocesadodores de 8, 16 y 32 bits. Para nuestra aplicación es suficiente el manejo de 8 bits.

En los cursos de microprocesadores y de diseño de sistemas con microprocesadores, se dan las bases necesarias para realizar sistemas básicos con algunos microcontroladores. De esta forma se conocen características generales de algunos microcontroladores de las familias de Intel y Motorola; por ejemplo, el 80C51 y el MCHC11, ambos microcontroladores fueron desarrollados para aplicaciones automotrices, por lo que tienen una estructura semejante. El costo de ambos microprocesadores es también muy parecido, aunque el HC11 es un poco más barato.

Para el caso de nuestro desarrollo se contaba ya con el microprocesador 8XC552 de Intel, además de existir la infraestructura necesaria para su manejo. Sin embargo, se hizo un análisis cualitativo entre los dos microcontroladores. Ambos tienen funciones muy semejantes. Aunque para ventaja en este proyecto, el microcontrolador de Intel cuenta con un dispositivo específico llamado *timer* ó temporizador, muy necesario en la etapa de decodificación. Por lo antes mencionado, se eligió el microcontrolador 8XC552 de Intel.

Descripción del microcontrolador 8XC552 de Intel

Este microcontrolador fue creado para la industria automotriz, está basado en el microcontrolador 80C51 de Intel, siendo una microcontrolador de 8 bits. De este circuito se dispone de tres versiones, tabla 2.3.

Versión	256 bytes de RAM Interna	Sin ROM Interna	8K de EPROM Interna	8K de ROM Interna
80C552	X	X		
83C552	X			X
87C552	X		X	

Tabla 2.3
Versiones de microcontrolador 8XC552.

En la tabla 2.3 se muestra que la diferencia entre las tres versiones es el tipo de memoria ROM, ésta puede ser programable una sola vez (87C552) o indefinidamente (83C552), o puede puede no incluirse dentro del circuito integrado (80C552).

Las características principales de las tres versiones son: cinco puertos de Entrada/Salida de 8 bits y un puerto de Entrada (convertidor A/D), dos temporizadores/contadores de eventos y un tercer temporizador/contador de eventos acoplado a registros de comparación y captura, dos interfases seriales (UART e I²C) y un temporizador *watch-dog* (fig 2.4), además de manejar 15 fuentes de interrupción con dos niveles de jerarquía.

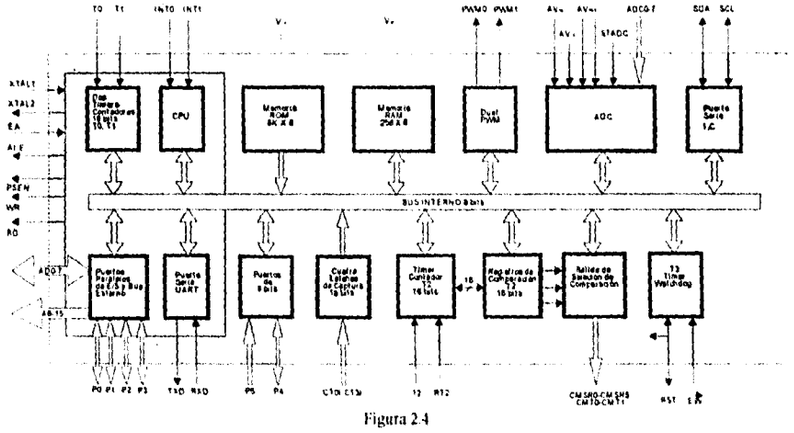


Figura 2.4
Diagrama de bloques de la estructura interna del microcontrolador 8XC552 de Intel.

De las tres versiones de este microcontrolador se eligió el µC 80C552, ya que se puede utilizar una memoria ROM externa para el desarrollo del programa de decodificación teniendo la posibilidad de grabar esta memoria el número de veces necesario.

Este microcontrolador maneja un mapa de memoria distinto para datos e instrucciones. La memoria de instrucciones (ROM) cuando se maneja externamente puede direccionar hasta

64K. El manejo de memoria de datos (RAM) puede ser también hasta de 64K. los primeros 256 bytes son internos y estan divididos en dos grupos, el primer grupo que se ubica de la dirección 00H a la 7FH es direccionable directa e indirectamente, en este bloque se manejan cuatro bancos de registros, el valor del *stack pointer* cuando ocurre un *reset* y memoria para datos; el segundo grupo está ubicado entre las direcciones 80H a la FFH y se puede direccionar de dos formas: directa e indirectamente. Cuando se accesa de forma directa se accesa a los **regitros de control** del microcontrolador (*special function registers SFR*) los cuales configuran el modo de operación del mismo, al hacerlo de forma indirecta se utiliza este bloque como memoria para datos. En la figura 2.5 se muestra los mapas de datos y direcciones.

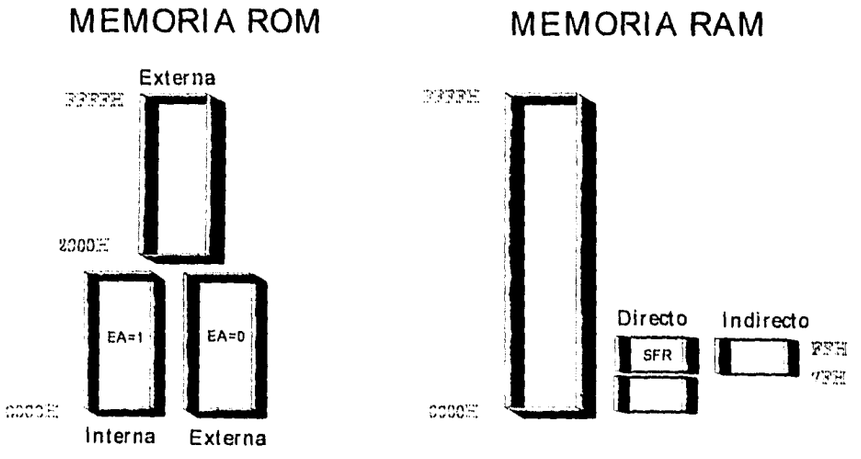


Figura 2.5
Mapa de memoria del microcontrolador 80C552.

Acontinuación se hace una breve descripción del *timer* 2, parte fundamental para el dasarrollo del sistema de identificación.

Descripción del *Timer* 2 (T2)

El *timer* 2 es un temporizador/contador de 16 bits, se compone de dos registros de 8 bits TMH2 (parte alta) y TML1 (parte baja). El *timer* 2 puede recibir dos señales como reloj. La primera es la frecuencia de oscilación del cristal del microcontrolador entre 12 y la segunda es

una señal externa. Tiene una factor de preescala, este factor de división pude ser 1, 1/2, 1/4 o 1/8 de la señal elegida. Cuando el factor de preescala es cambiado o cuando la fuente de la señal de reloj cambia, el contenido de sus registros es borrado, existe otra forma de limpiar este contenido y es a través de una señal externa de *reset* (reinicializar) en la terminal RT2.

La lectura del *timer* 2 debe ser "en el vuelo" ya que no posee *latches* (elementos que retienen el valor de la lectura), esta precaución se debe tomar en la programación que se desarrolle. Este *timer* tiene la capacidad de generar dos interrupciones, la primera cuando el byte menos significativo TML1 se sobrecarga, la segunda interrupción ocurre cuando el byte mas significativo TML2 se sobrecarga, generando además banderas de interrupción, las cuales deben ser borradas por software.

El *timer* 2 tiene una lógica de comparación y captura, está conectado a cuatro registros de captura de 16 bits CT0, CT1, CT2 , CT3 y a tres registros de comparación de 16 bits. Estos registros de captura almacenan el estado del *timer* 2 cuando una transición de voltaje ocurre en la terminal correspondiente a cada registro en el puerto 1, generando una interrupción, éstas funciones ayudan a la medida de tiempo entre eventos, estos registros se pueden comparar con el valor del registro de comparación alterando como resultado el estado del puerto 4. A continuación se muestran los registros de control del *timer* 2.

Registro de Habilitación de Interrupciones para T2 (IEN1)

Programando este registro se puede saber qué tipo de interrupciones producidas por el *timer* 2 serán atendidas por el microcontrolador y estas pueden ser de tres tipos: por sobre carga ya sea de 16 u 8 bits, por comparación de registros o por captura de registros (fig. 2.6 y tabla 2.4).

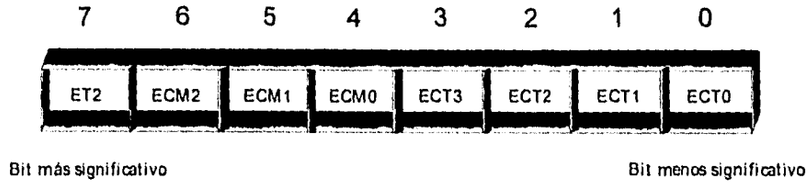


Figura 2.6
Bits de control de interrupciones del registro IEN1
para el *timer* 2.

Bit	Símbolo	Función
IEN1.7	ET2	Habilitación de interrupción por sobre carga
IEN1.6	ECM2	Habilitación de Int por comparación 2 (T2)
IEN1.5	ECM1	Habilitación de Int por comparación 1 (T2)
IEN1.4	ECM0	Habilitación de Int por comparación 0 (T2)
IEN1.3	ECT3	Habilitación de Int por reg de captura 3 (T2)
IEN1.2	ECT2	Habilitación de Int por reg de captura 2 (T2)
IEN1.1	ECT1	Habilitación de Int por reg de captura 1 (T2)
IEN1.0	ECT0	Habilitación de Int por reg de captura 0 (T2)

Tabla 2.4

Muestra el tipo de interrupción que se habilita al seleccionar un bit del registro.

Registro de Control para T2 (TM2CON)

Con este registro se elige el tipo de interrupción por sobre carga (16 u 8 bits), se habilita la posibilidad de limpiar el *timer* 2 con una señal externa, se elige la señal que servirá como reloj al *timer* 2, así como su factor de división. La figura 2.7 muestra un esquema del registro TM2CON y la tabla 2.5, señala cual es la función que realiza cada uno de dichos bits, al ser puestos en estado lógico alto. Por último, la tabla 2.6 permite saber con qué combinaciones de 1's y 0's en los bits T2P0 y T2P1 se obtienen las preescalas para el reloj del T2 así como las combinaciones en los bits T2MS0 y T2MS1 para decidir el modo de operación de T2.

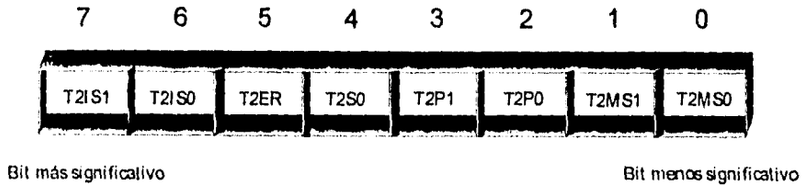


Figura 2.7

Diagrama esquemático del registro TM2CON

Bit	Símbolo	Función
TM2CON.7	T2IS1	Selección de Int por 16 bits
TM2CON.6	T2IS0	Selección de Int por byte
TM2CON.5	T2ER	Habilitación de *reset* externo
TM2CON.4	T2B0	Bandera de Int por sobre carga
TM2CON.3	T2P1	Pre escala
TM2CON.2	T2P0	Pre escala
TM2CON.1	T2MS1	Modo de selección
TM2CON.0	T2MS0	Modo de selección

Tabla 2.5

Indicación por cada bit del registro TM2CON

T2P1	T2P0	Pre escala del T2
0	0	Reloj del micro
0	1	Reloj del micro/2
1	0	Reloj del micro/4
1	1	Reloj del micro/8

T2MS1	T2MS0	Modo de operación del T2
0	0	T2 Apagado
0	1	Fuente del T2=reloj del micro/12
1	0	Modo de prueba
1	1	Fuente del T2=terminal T2

Tabla 2.6
Combinaciones para la preescala y modo de operación de T2.

Registro de Control de Captura para T2 (CTCON)

En este registro se elige que tipo de transición causará la captura del valor del *timer* 2, esta transición puede ser un canto de bajada, un canto de subida o ambos especificando que registro o registros serán utilizados para guardar el contenido del *timer* 2. La figura 2.8 muestra un esquema del registro, mientras que la tabla 2.7 describe la función de cada uno de los bits del mismo registro.

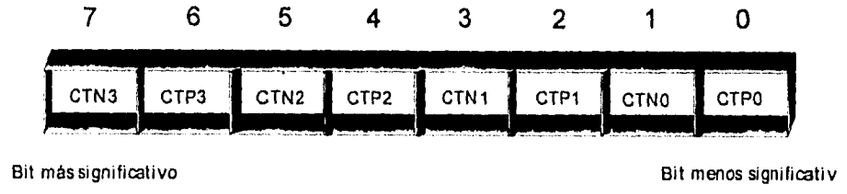


Figura 2.8
Esquema del registro CTCON

Bit	Simbolo	Función
CTCON.7	CTN3	Registro de captura 3 disparado por un canto de bajada CT3I
CTCON.6	CTP3	Registro de captura 3 disparado por un canto de subida CT3I
CTCON.5	CTN2	Registro de captura 2 disparado por un canto de bajada CT2I
CTCON.4	CTP2	Registro de captura 2 disparado por un canto de subida CT2I
CTCON.3	CTN1	Registro de captura 1 disparado por un canto de bajada CT1I
CTCON.2	CTP1	Registro de captura 1 disparado por un canto de subida CT1I
CTCON.1	CTN0	Registro de captura 0 disparado por un canto de bajada CT0I
CTCON.0	CTP0	Registro de captura 0 disparado por un canto de subida CT0I

Tabla 2.7
Descripción de cada uno de los bits del registro CTCON.

Registro de Banderas del *timer* 2 (TM2IR)

El estado de cada bit de este registro describe que interrupción con respecto al *timer* 2 se ha efectuado y lo hace señalando con un estado lógico alto. Las banderas deben ser borradas por software. La figura 2.9 y Tabla 2.8 muestran respectivamente el esquema del registro TM2IR y lo que significa la bandera de cada uno de los bits del registro.

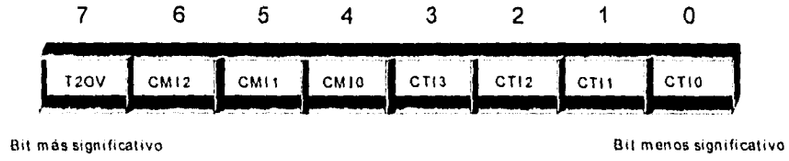


Figura 2.9
Esquema del registro TM2IR

Bit	Símbolo	Función
TM2IR.7	T2OV	Bandera de Int por sobre carga de 16 bits
TM2IR.6	CM2	Int por reg de comparción CM2
TM2IR.5	CM1	Int por reg de comparción CM1
TM2IR.4	CM0	Int por reg de comparción CM0
TM2IR.3	CTI3	Int por reg de captura CT3
TM2IR.2	CTI2	Int por reg de captura CT2
TM2IR.1	CTI1	Int por reg de captura CT1
TM2IR.0	CTI0	Int por reg de captura CT0

Tabla 2.8
Tabla de banderas de interrupción del registro TM2IR

Reloj de Tiempo Real

El bloque de identificación y almacenamiento cuenta con un dispositivo el cual lleva la fecha, hora, minutos y segundos. Este dispositivo proporciona al microprocesador la fecha y hora de entrada de los trabajadores así como la hora y fecha de salida. Los datos son tomados a partir de que el trabajador checa su tarjeta.

Se eligió un reloj de tiempo real con la característica de poderse conectar directatamente al microprocesador, siendo independiente de él. Estos dispositivos tienen una base de tiempo independiente, generalmente un cristal.

El reloj de tiempo real elegido fue el MC146818 de MOTOROLA que tiene las siguientes características generales.

- Diseñado con el concepto MOTEL el cual permite conectar este dispositivo con diferentes microprocesadores, microcomputadoras, y grandes computadoras.
- Cuenta segundos, minutos y horas del día.
- Cuanta días de la semana , días del mes, mes y año.
- Opera en un rango de voltajes de 3 a 6 volts.
- Puede operar con tres frecuencias de cristal diferentes: 4.194304 MHz, 1.048576 MHz o 37.768 KHz.
- Disipa entre 40 a 200 μW cuando opera a la más baja frecuencia.
- Disipa entre 4 a 20 mW cuando opera a la más alta frecuencia.
- Representación binaria o en BCD de la fecha y hora.
- Posibilidad de operar en formatos de 12 y 24 horas.
- Copatibilidad con el bus de microprocesadores.
- Bus multiplexado para tener una mayor eficiencia en el número de terminales.
- Bytes de registros de control y registros de tiempo.
- Bytes de memoria RAM estática.
- Señales de interrupción.
- Circuitos de protección contra voltajes estáticos en las terminales de entrada.

El concepto MOTEL quiere decir que este tipo de dipositivos pueden ser conectados a una gran variedad de microprocesadores sin la necesidad de una lógica externa. Los microprocesadores que tengan una estructura de bus basda en un microprocesador 6800 de MOTOROLA o en un mieroprocesador 8080 de Intel (MOT motorola EL intel) pueden manejar este dispositivo.

Mapa de memoria del reloj de tiempo real

El mapa de memoria de MC146818 consiste de 50 bytes de memoria RAM estática, los cuales son direccionables en todo momento por el microprocesador (escritura y lectura), 10 bytes de memoria RAM los cuales contiene la hora, fecha, alarma y calendario, así como 4 bytes de control del reloj. La figura 2.10 muestra el mapa de memoria del reloj y la ubicación de los 14 bytes que indican la fecha y la hora.

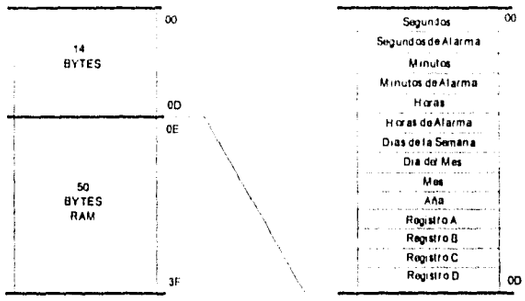


Fig 2.10
Mapa de memoria de la memoria del Reloj de Tiempo Real.

El programa del microcontrolador puede obtener la información de fecha y hora accesando directamente a los registros que contienen la información. Al inicio del programa de aplicación debe inicializar los 10 registros que contienen la información de fecha y hora, así como programar los 4 registros de control del reloj de tiempo real.

Ciclo de actualización

El MC146818 ejecuta un ciclo de actualización una vez cada segundo. Este ciclo tiene por objeto actualizar los 10 registros que contienen la información del reloj, además de comparar los registros de alarmas para generar una interupción.

Cuando se utilizan cristales de 4.194304 MHz o 1.048576 MHz el ciclo de actualización dura 248 µS, mientras que al utilizar el cristal de 32.768 KHz el cilo dura 1948 µS. Durante este tiempo los datos del reloj no pueden ser accesados por el microprocesador.

Cuando se accesa la información del reloj de tiempo real durante un ciclo de actualización los datos leídos por el microprocesador son indefinidos. Para evitar este echo se tienen tres métodos para accesar la información del reloj. El primero es utilizar la interrupción de fin de ciclo de actualización, cuando esta interrupción está habilitada, después de cada ciclo de actualizacion se generará una interrupción, la cual indicará que se tienen 999 ms para accesar la información del reloj. El segundo método es utilizar el bit de ciclo de actualización en progreso, cuando este bit es un 1 los datos leídos del reloj no son válidos, cuando este bit es un cero se tienen 244 µS para leer la información. El tercer método es utilizar una interrupción periódica para leer la información.

Registros

En los registro de control del MC146818 se elige el cristal a utilizar, el formato que tendrán los datos (binario o BCD), si es que se generan interrupciones, además de monitorear bits para saber cuando se puede hacer una lectura del reloj.

Registro A

En este registro se avisa acerca del ciclo de actualización y la figura 2.11 muestra el esquema del mismo. **UIP** (update in progres bit) es el bit de ciclo de actualización en progreso, éste funciona como una bandera que debe ser monitoreada por el programa del microprocesador. Cuando este bit es un 1 quiere decir que el ciclo de actualización se está ejecutando o que se va a ejecutar.

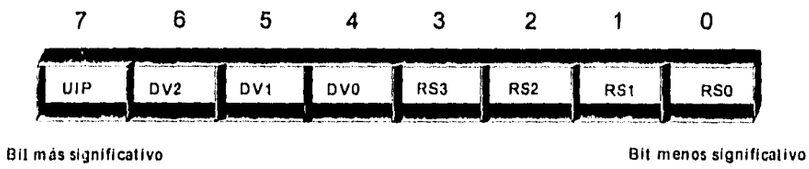


Figura 2.11
Esquema del registro A del Reloj de Tiempo Real

Cuando es un 0 quiere decir que el ciclo no se está ejecutando por lo menos 244 µS en todas las bases de tiempo que se muestran en la tabla 2.9 El bit UIP es un bit de sólo lectura y no es afectado por un *reset*. Cuando se escribe un 1 al bit SET de registro B se desahabilita el ciclo de actualización y el bit UIP es borrado.

UIP bit	Base de Tiempo	Ciclo de actualización (t_{UC})	Tiempo mínimo antes del siguiente cilclo de actualización (t_{BUC})
1	4.194304 MHz	248 µS	
1	1.048576 MHz	248 µS	
1	32.768 KHz	248 µS	
0	4.194304 MHz		244 µS
0	1.048576 MHz		244 µS
0	32.768 KHz		244 µS

Tabla 2.9
Tabla de relación para las bases de tiempo del Reloj de Tiempo Real

DV2, DV1, DV0 Estos bits permiten programar los factores de división y además identificar qué base de tiempo se está usando. La tabla 2.10 muestra según la base de tiempo cómo es su modo de operación y finalmente el número de los bits de división.

Base de Tiempo	Bits de división registro A			Modo de Operación	*Reset*	N bits de división
	DV2	DV1	DV0			
4.194304 MHz	0	0	0	SI		N=0
1.048576 MHz	0	0	1	SI		N=2
32.768	0	1	0	SI		N=7
Otro	1	1	0	NO	SI	
Otro	1	1	1	NO	SI	

Tabla 2.10
Selección del cristal de oscilación.

Según las combinaciones de los bits **RS3, RS2, RS1, RS0** se seleccionan uno de los 15 *taps* del divisor para la generación de una interrupción o para la generación de una señal cuadrada de salida, como se muestra en la tabla 2.11.

Bits de Selección				Cristal de 4.194304 o 1.045876 MHz		Cristal de 32.768 KHz	
RS3	RS2	RS1	RS0	Interrupción periódica t_{PI}	Frecuancia de la señal de salida	Interrupción periódica t_{PI}	Frecuancia de la señal de salida
0	0	0	0	X	X	X	X
0	0	0	1	30.517 µS	32.768 KHz	3.90625 mS	256 Hz
0	0	1	0	61.035 µS	16.348 KHz	7.8125 mS	128 Hz
0	0	1	1	122.070 µS	8.192 KHz	122.070 µS	8.129 KHz
0	1	0	0	244.141 µS	4.096 KHz	244.141 µS	4.096 KHz
0	1	0	1	488.281 µS	2.048 KHz	488.281 µS	2.048 KHz
0	1	1	0	976.562 µS	1.024 KHz	976.562 µS	1.024 KHz
0	1	1	1	1.953125 mS	512 KHz	1.953125 mS	512 Hz
1	0	0	0	3.90625 mS	256 KHz	3.90625 mS	256 Hz
1	0	0	1	7.8125 mS	128 KHz	7.8125 mS	128 Hz
1	0	1	0	15.625 mS	64 Hz	15.625 mS	64 Hz
1	0	1	1	31.25 mS	32 Hz	31.25 mS	32 Hz
1	1	0	0	62.5 mS	16 Hz	62.5 mS	16Hz
1	1	0	1	125 mS	8Hz	125 mS	8 Hz
1	1	1	0	250 mS	4 Hz	250 mS	4 Hz
1	1	1	1	500 mS	2 Hz	500 mS	2 Hz

Tabla 2. 11
Tabla que muestra las diferentes salidas de tiempo del Reloj
de Tiempo Real, para dos frecuencias de Cristal diferentrs.

Registro B

La figura 2.12 muestra el esquema de este registro. En él se habilitan las diferentes interrupciones del Reloj de tiempo real y cada uno de los bits del registro realiza las siguientes funciones:

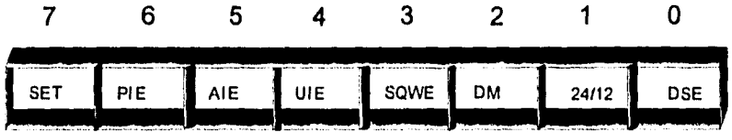


Figura 2.12
Esquema del registro B para habilitar las interrupciones.

SET. Cuando este bit tiene el valor de 0 el ciclo de actualización se puede llevar acabo una vez por segundo. Cuando el valor es 1 cualquier ciclo de actualización es abortado y el programa del microprocesador debe inicializarlos registros de fecha y hora.

PIE (*periodic interrupt eneable*). Este bit habilita una interrupción periódica, permitiendo que la bandera de interrupción PF en el registro C cause que la señal IRQ esté en cero. Para habilitar esta interrupción se debe escribir un 1 en este bit. La frecuencia de esta interrupción es seleccionada por los bits RS3, RS2, RS1 y RS0 del registro A. Cuando este bit es un 0 la interrupción es deshabilitada.

AIE (*alarm interrupt enable*). Este bit es la habilitación para una interrupción por alarma . Cuando este bit es un 1 y al ocurrir que los tres registros de tiempo sean iguales a los tres registros de alarma, la bandera de alarma AF provoca que la señal IRQ sea cero. Cuando este bit es 0 esta opción es deshabilitada.

UIE. (*update ended inerrupt enable*). Este bit es la habilitación de la interrupción por fin de ciclo de actualización, este permite que la bandera UF en el registro C genere una interrupción. Cuando ocurre un *reset* o el bit SET es un 1 el bit UIE es puesto a 0.

SQWE..Cuando este bit es un 1 una señal cuadrada con un frecuancia especificada por los bits RS3, RS2, RS1 y RS0 se presenta en la terminal SQW. Cuando este bit es un cero la terminal SQW tiene un nivel de 0.

DM. (*Data made*). Con este bilt se selecciona el formato que usarán los registros de fecha y hora. Cuando este bit es un 1 el formato es binario, cuando este bit es un cero el formato es BCD.

24/12. Este bit establece el formato que usarán las horas. Cuando este bit es un 1 utiliza un formato de 24 hrs, cuando este bit es un 0 utiliza un formato de 12 hrs.

DSE. (*Daylight saving enables*). Este bit controla dos actualizaciones especiales, esto es, se pueden programar los "tiempos de verano" que se manejan en algunas partes del mundo, especialmente en Europa y que consisten en adelantar o atrasar una hora el reloj para aprovechar la luz del día: El último lunes de abril el tiempo se incrementa de 1:59:59 AM a 3:00:00 AM. El último lunes de octubre cuando la hora alcanza las 1:59:59 AM cambia a 1:00:00 AM. Estas actualizaciones no ocurren cuando este bit es 0.

Registro C

El siguiente esquema (figura 2.13), muestra los bits del registro. Cada uno de ellos es una bandera que indica cual fue la naturaleza de la interrupción:

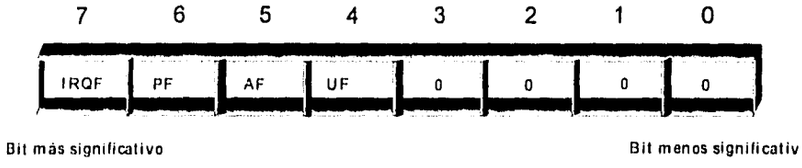


Figura 2.13
Esquema del registro C para las banderas de interrupción

IRQF. Este bit es la bandera de petición de interrupción. Puesto en 1 cuando alguna bandera o algunas PF, AF o UF es encendida (1 lógico), esto provoca que la terminal IRQ tenga cero volts, este bit es borrado cuando el programa lee el registro C o cuando existe un *reset*.

PF. Este bit representa la bandera de interrupción periódica, es un bit de sólo lectura. Puesto a 1 cuando un tap del contador controlado por RS3, RS2, RS1 y RS0 detecta un flanco. Este bit es borrado cuando el programa lee el registro C o cuando existe un *reset*.

AF. Este bit representa la bandera de interrupción por alarma, esto ocurre cuando los registros de alarma coinciden con los registros de fecha y hora, provocando que el bit sea un 1. Este bit es borrado cuando el programa lee el registro C o cuando existe un *reset*.

UF. Este bit representa la bandera de interrupción por el fin de ciclo de actualización, este bit es puesto a 1 cuando el ciclo finaliza. Este bit es borrado cuando el programa lee el registro C o cuando existe un *reset*.

Bit 3 al 0 Estos bits no son utilizados, son de sólo lectura y dan como resultado 0.

Registro D

Este registro se utiliza para monitorear si los datos almacenados en el reloj de tiempo real son válidos, cuando existe alguna falla en su polarización.

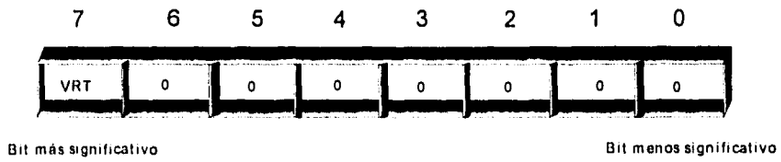


Fig 2.14
Esquema del registro D, este registro indica que el
Reloj de Tiempo Real esté correctamente polarizado.

VRT. Este bit indica si los datos en RAM son válidos. Este bit es proporcionado por el monitoreo de la terminal PS (power sense) la cual indica si la polarización es adecuada. Cuando esta polarización es incorrecta este bit es un 0. Este bit es de sólo lectura y la única forma de ponerlo a 1 es haciendo una lecura del registro D.

Bit 6 al 0. Estos bits no son utilizados, son de sólo lectura y cuando se leen se tiene 0.

A continuación se muestra el circuito de la figura 2.15 propuesto por el fabricante para sensar la polarización. La terminal de entrada PS (power sense) determina si el contenido de la RAM es valido. Cuando se polariza el circuito esta terminal debe mantenerse en un nivel bajo durante el tiempo t_{PLH} después debe ser puesto en 1 leyendo el registro D.

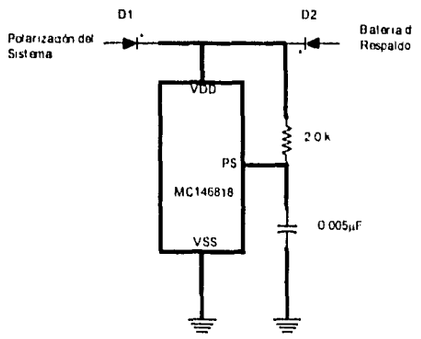


Figura 2.15
Circuito para sensar la polarización del Reloj de Tiempo Real.

Memoria IC (*Integrated Circuit*)

En este tipo de memorias se requiere de menos tiempo para ser escritas o borradas en comparación con las memorias EPROM (*Erasable Programable ROM*).

Basada en el mismo proceso que las memorias EPROM, la tecnología de las memorias IC es muy parecida a aquéllas en su confiabilidad, su no-volatilidad (ésto es debido a una batería interna que mantiene actualizada la información) y su densidad.

Los registros de comandos de las memorias IC permiten que sean compatibles con los microprocesadores y de esta forma se les puede borrar, programar o realizar en ellas otras operaciones, dentro de un sistema digital. El diseñar sistemas con memorias IC es lo mismo o muy semejante a hacerlo con cualquier otra tecnología.

Se requiere asegurar la alimentación permanente de este tipo de memorias ya que sólo así se garantiza que los datos se mantengan en ella.

Las tarjetas IC están diseñadas bajo las normas de la PCMCIA (en inglés, "Asociación Internacional de Tarjetas de Memoria para Computadoras Personales). La norma especifica el aspecto físico, el eléctrico, la información sobre su estructura y las características sobre el formato de los datos. La superficie de este tipo de tarjetas es de 85.6 milímetros por 54.0 milímetros con una altura de 3.3 milímetros. Estas tarjetas también reciben el nombre de TC que son las siglas de "Thin Card" (Tarjeta Delgada). La figura 2.15.a muestra el tamaño de una tarjeta de este tipo comparada con una tarjeta telefónica figura 2.16.b, ambas son similares en área.

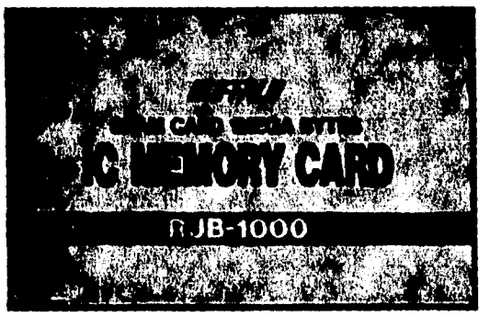


Figura 2.16.a
Tarjeta del tipo ICRAM

Figura 2.16 b
Las figuras 2.16 a y b permiten hacer una comparación física del tamaño de a tarjeta IC con una tarjeta telefónica.

El número de terminales es de 68 e incluye 26 líneas de direcciones usadas para accesar directamente 64 Megabytes. Las tarjetas IC están disponibles con capacidades de 1 y 4 Megabytes. Estas tarjetas seguirán creciendo cada vez, haciéndose más competitivas con los manejadores (*drives)* de disco.

Las tarjetas IC tienen la ventaja de poder manejarse tanto en diferentes sistemas con microprocesadores como en una PC, en esta última utilizando un *drive* especial para estas tarjetas. Al conectar el *drive* a la PC se tienen dos opciones, la primera es darle el formato a la tarjeta como si fuera un disco y usarla como tal. La segunda opción es el hacer uso de una programación (*software)* exclusiva para estas tarjetas. Ambas opciones permiten que la tarjeta sea escrita y borrada por medio de la PC.

El hecho de que estas tarjetas sean portátiles permite que se les den diversas aplicaciones, pues se puede manejar la misma información en un sistema independiente con microprocesador e interactuar con una PC, sin que estén necesariamente conectados el sistema y a la PC. Para que la información contenida en la tarjeta IC no sea alterada, es necesario mantener en forma constante la alimentación de la misma. El diseño físico de las tarjetas permite contener una batería intercambiable dentro de ellas, con el objeto de poderlas transportar y mantener su información.

Descripción Técnica de la Tarjeta de Memoria

En el presente proyecto se optó por usar una tarjeta IC para registrar la hora de entrada y salida de cada trabajador. El fácil manejo de ésta permitirá contar con dos opciones. La primera sería que el sistema que contiene a la tarjeta estuviera anexo al reloj checador mecánico, el cual podría no estar necesariamente junto a la PC que lleva la nómina. De esta manera, la tarjeta sería retirada del sistema y transportada al *drive* especial ya conectado a la PC. La otra opción sería utilizar el mismo sistema y obtener la información de la tarjeta a partir de la comunicación serial entre el sistema y la PC.

La capacidad de almacenamiento de la tarjeta se calculó de la siguiente manera:

El número total de trabajadores que registran su asistencia en el edificio principal de la Facultad de Ingeniería (que es donde se instalará este sistema) es de 295.

Cada trabajador checa su tarjeta por lo general 2 veces al día. Además, se encuentran los trabajadores que disponen de plaza y media y los que trabajan tiempos extras, que son aproximadamente el 25% del total y para estos casos se tienen cuatro registros al día por trabajador. Lo anterior da como resultado un promedio de 750 registros diarios de checado.

Semanalmente habrá un total aproximado de 3,800 registros incluyendo sábados, domingos y días festivos. Al mes se tendrán entonces 15,200 registros.

El formato por registro de cada trabajador que se almacenará en la tarjeta de memoria IC contiene 28 caracteres: 2 para el código de inicio y fin del R.F.C., 11 para el R.F.C., 2 para el día de la semana, 2 para la hora, 2 para los minutos y 2 para los segundos. Cada carácter representa 1 Byte, por lo que son 28 bytes por registro de checado de cada trabajador. Si se tienen un total de 15,200 registros al mes, entonces serán un total de (15,200)x(28)=425,600 Bytes ó 425 KBytes.

Una tarjeta IC de 1 MegaByte podrá almacenar aproximadamente 2.5 meses de registros sin ser borrada. Además ofrece a la Facultad el poder ampliar el número de registros de checado al día ó el agregar más trabajadores a la lista, sin que se tema por la capacidad de la tarjeta.

A continuación se presenta una tabla con las características eléctricas de la Tarjeta.

Características Eléctricas
Tipo de dispositivo: CMOS- SRAM
Voltaje de polarización: de 3 a 5 Volts
Densidad: 1 MegaByte x 8 Bits
Tiempo de Acceso: 200 ns
Consumo de potencia : activo: 120 mA en espera: 1.5 mA
Temperatura de operación: de 0° a 60° C
Temperatura de almacenamiento: de -20° a 60° C

Tabla 2.12
Características eléctricas de la tarjeta IC.

De la tabla se puede ver que el tiempo de acceso es un poco mayor al de la memoria EPROM (tabla 2.1), sin embargo, el microprocesador también puede escribir y leer en ella sin problema.

El SCANNER

Como se mencionó en el capítulo uno se optó por la tecnología de código de barras para identificar al trabajador. Las barras impresas en diversos tipos de superficies pueden ser leídos cuidadosamente, inclusive a distancia, por lectores llamados SCANNERS.

El dispositivo de lectura óptica ó SCANNER que se eligió aquí es el SR11, fabricado por "DATALOGIC".

El SR11 es un SCANNER tipo SLOT con encapsulado metálico de uso rudo. Su diseño permite ser utilizado en aplicaciones tales como control de accesos en estacionamientos, laboratorios, áreas de trabajo o de descanso, control de personal, etc. Este dispositivo es capaz de leer información impresa en alta, media y baja densidad. La alta resolución es ideal para códigos de alta densidad generados por tecnologías de impresión muy buenas, tales como las térmicas, las fotográficas o las tipo *offset*. La baja resolución es muy efectiva para la lectura de códigos de baja densidad o que se encuentran hasta cierto punto desgastados que fueron impresos por impresoras de matriz de puntos o de inyección de tinta. La alta calidad del sistema de óptica interno lee incluso, marcas flexibles sobre superficies luminosas tales como las tarjetas de plástico laminado.

Descripción Técnica del SCANNER

En el SR11, la luz emitida por el LED llega a un lugar específico dentro de la superficie del código (spot). Esta luz puede ser roja o infrarroja (ésta es invisible al ojo humano). La luz se refleja en diferentes intensidades dependiendo del color de las barras y del fondo, regresando al lugar donde se encuentra el área del lector. Una barra obscura absorbe la mayor parte de la luz mientras que un espacio blanco la refleja casi en su totalidad. Cuando un objeto rotulado con código de barras pasa por el lugar de lectura ó SLOT, la luz será modulada por las áreas blancas y negras del código de barras. Esta luz es enfocada por unos lentes montados sobre una celda fotoeléctrica y transformada en una señal eléctrica. Después de amplificarla, la señal es **digitalizada** por medio de un comparador de voltaje, el cual sensa la media de la reflectancia del código, y habilita la lectura de los rótulos de bajo contraste. La señal digital que transmite el SR11 consiste en una secuencia de pulsos correspondientes a los elementos del código de barras.

Toda la óptica y electrónica necesarias para efectuar la interpretación del código de barras están instaladas dentro del área de lectura. Los componentes se encuentran montados sobre una tarjeta de circuito impreso y protegidos por un encapsulado de metal. Tiene un LED (*light emitter diode*) localizado al frente para indicar las "lecturas buenas", éste indica cuando una lectura ha sido aceptada. Consta de un cable de 2 metros de largo para poder conectar el lector al decodificador. El cable puede ser orientado en una de tres direcciones posibles con el fin de facilitar su montaje.

El SR11 emite luz infrarroja y es recomendable para barras en color negro sobre superficies blancas, que están cubiertas con una pintura especial . El caso de la pintura se maneja cuando se protege al código de barras con una mica que lleva una banda de pintura vegetal que no permite que la etiqueta sea vista por el ojo humano y además impide que sea reproducido, pero es transparente para el scanner.

A continuación se muestran el diagrama eléctrico (fig 2.17) del *SCANNER* y la tabla 2.13 con las especificaciones del mismo. El tipo de salida es de colector abierto operando en activo bajo, ésto significa que la salida está en **alto** cuando el LED incide sobre una barra y está en **bajo** cuando el LED incide sobre un espacio.

DIAGRAMA ELECTRICO

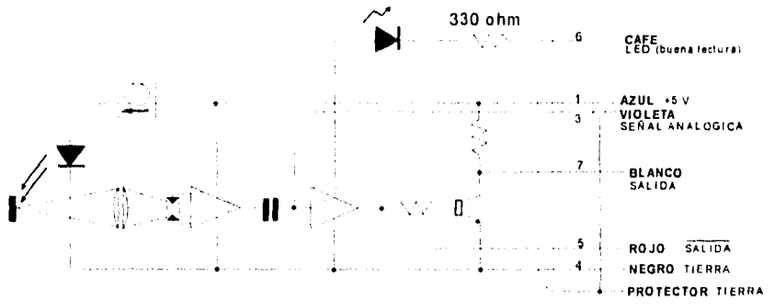


Figura 2.17
SCANNER SR1H

Características Eléctricas	
Voltaje de alimentación:	típico 5.0 V de DC; mínimo 4.5 V de DC; máximo 6.0 V de DC
Rizo máximo:	150 mV
Consumo de corriente:	típico 17 mA
Características ópticas	
Fuente de luz:	LED infrarrojo, λ = 940 nm
Razón de lectura:	7.0 a 2000 mm/seg
Ancho del área de lectura:	3 mm
Radio de error; b=barra, e=espacio:	b/b 10 %; e/e : b/e 20%; 1* b/2* b 20%
Características de la salida	
Señal:	digital
Tipo:	colector abierto, activo bajo*
Tiempos de levantamiento y de caída:	tiempo de levantamiento, 10 a 90 % 1.5 µs (típico)
	tiempo de caída, 90 a 10 % 1.5 µs (típico)
Carga:	20 V de DC (máximo), 20 mA (máximo)
Especificaciones mecánicas	
Dimensiones:	(anchura, altura, profundidad). 126 x 52x 29 mm
Peso:	300 g
Encapsulado:	aluminio
Cable (conectado a la unidad):	2 m
Conector del cable:	DE-9P
Características del ambiente	
Temperatura para operar:	- 10° a + 60° C
Humedad:	5 a 90 %, no condensada
cond. máx. de luz en el ambiente:	6000 lux (ángulo de luz 60°)

Tabla 2.13
Características eléctricas del *SCANNER* SR1H.

Pantalla de Cristal Líquido (*Display*)

El bloque de identificación y almacenamiento cuenta con una pantalla de cristal líquido, con la cual se da a conocer a los usuarios el estado actual del sistema, este dispositivo maneja la siguiente información:

- Despliega el R.F.C. leído.
- Despliega la hora en la que se leyó el R.F.C..
- Despliega la hora del reloj de tiempo real.
- Ayuda desplegando información para actualizar el reloj de tiempo real.
- Informa al usuario cuando este módulo esta interactuando con una computadora personal.
- Informa cuando la tarjeta TC-RAM esta en el límite de su capacidad.
- Informa cuando la tarjeta TC-RAM no está presente.
- Informa cuando la batería de la tarjeta TC-RAM está baja.
- Informa cuando la protección contra escritura de la tarjeta TC-RAM está puesta.

Por las carecteristicas de longitud de los mensajes que se quieren desplegar se eligió el *display* AND-491, el cual tiene como características principles el manejar dos renglones de 16 caracteres cada uno. Puede ser conectado a un bus de datos de 4 u 8 bits, además de un control de intensidad. A continuación se da una explicación de la programación y funcionamieto de este elemento.

Registros

Este *display* cuenta con dos registros, un registro de instrucciones IR y un registros de datos DR. El registro de instrucciones almacena los camandos a ejecutarse en el *display* como pueden ser: borrar el *display*, mover el cursosr, etc., mantiene la dirección de los datos a escribirse en el *display*. La figura 2.18 muestra un esquema de la posición del cursor en el *display* y el registro correspondiente a dicha posición.

Posición del Cursor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Dirección DD RAM (HEX)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
Posición del Cursor	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Dirección DD RAM (HEX)	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Fig 2.18
Esquema de los registros del *Display*

El registro de datos almacena la información que se quiere escribir en el *disaplay*. La información a desplegar puede provenir de dos fuentes, un dato externo almacenado en la memoria DD RAM del *display* o un dato grabado en la memoria CG RAM del *display*. En la memoria CG RAM el diseñador puede crear sus propios caracteres para posteriormente desplegarlos. El registro IR sólo puede ser escrito por el microcontrolador, mientras que el DR es un registro de escritura y lectura. Para accesar a estos registro se deben manejar dos señales que son RS (Register Selector) y R/W (señal de lectura-escritura), dependiendo del estado de ambas señales es la operación que se ejecuta en el *display* y que se indican en la tabla 2.14.

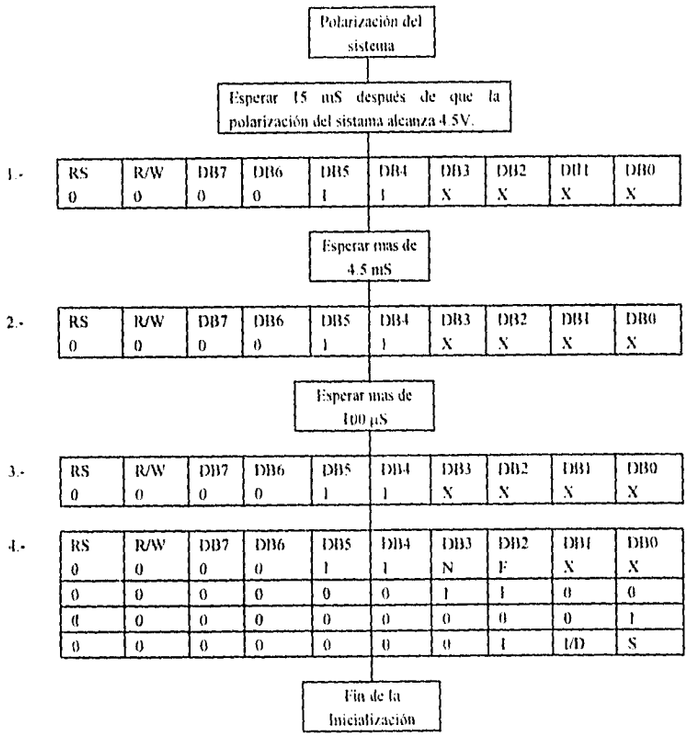
RS	R/W	Acción
0	0	Accesa al reg IR tamando la información como un comando.
0	1	Lee la bandera de "busy-flag" para saber si puede accesar al dispositivo.
1	0	Escribe la información a memoria RAM (DD o CG RAM) desplegando la infromación. Registro DR
1	1	Lee la información de memmoria RAM (DD o CG-RAM). Registro DR

Tabla 2.14
Manejo de señales RD y WR del registro

El *display* ocupa cierto tiempo para realizar una instrucción o deplegar la información, para que la información o el comando no se pierdan en este tiempo el *display* cuenta con una bandera, la cual indica al microprocesador si puede accesar a este dispositivo, por tanto antes de accesar a este dispositivo siempre habra que preguntar por esta bandera antes de mandar la información.

Inicialización

Para que este dispositivo pueda funcionar se debe inicializar, programando su forma de operación, esto es, el número de bits que utilizará en su bus de datos, además de otros factores. Esta inicialización debe realizarse un tiempo después de que la polarización del sistema ha alcanzado los 4.5V. A continuación se muestra un diagrama de como se realiza la inicialización del *display* en el sistema, esta inicialización es la propuesta por el fabricante.



Los pasos número 1 al 3 son para asegurar que el *display* va a utilizar un bus de datos de 8 bits, el cuarto paso indica al *display* el número de bits en el bus de datos, además del número de líneas que manejara el *display* (en este caso dos líneas), el tamaño de los carateres, activa el *display* posteriormente lo borra y termina programando la dirección en la que el *display* se moverá (la pantalla completa y el cursor).

Comúnicación entre una PC y el Sistema de Identificación

El sistema de identificación y almacenamiento cuenta con la posibilidad de transmitir la información que se tiene almacenada en él a una computadora personal, para el procesamiento de la misma. El microcontrolador seleccionado incorporá dos formas de comúnicación serial, un

puerto tipo UART de tipo "full duplex", esto quiere decir que permite la recepción y la transmisión de información al mismo tiempo. El otro medio de comúnicación serial es un puerto de tipo I^2C, el cual permite la comúnicación bidireccional entre dispositivos conectados a un mismo bus.

Para la comúnicación con la PC se eligió el puerto tipo UART, ya que es el mismo tipo de puerto serie que poseen las computadoras personales. El UART del microprocesador tiene cuatro posibles modos de operación, en todos éstos para iniciar una transmisión hay que accesar al registro SBUF.

- **Modo 0.** Los datos son recibidos y transmitidos a través de la terminal de recepción RXD, una señal de reloj es transmitida a través de la terminal TXD. El baud rate es 1/12 de la frecuencia de oscilación del cristal del microcontrolador.

- **Modo 1.** Se transmiten 10 bits a través de TXD o se reciben a través de RXD, el mensaje se compone de un bit de inicio, 8 bits de datos y un bit de fin, en este modo el *baud rate* es variable.

- **Modo 2.** Se transmiten 11 bits a través de TXD o se reciben a través de RXD, el mensaje se compone de un bit de inicio, 8 bits de datos, un bit programable, éste puede ser un bit de paridad (el bit mandado o recibido esta alojado en RB8 en los registros de control) y un bit de fin. En este modo el baud rate puede ser 1/32 o 1/64 de la frecuencia de oscilación del cristal del microcontrolador.

- **Modo 3.** El modo 3 recibe y transmite los mismos bits que el modo la única diferencia es que el *baude rate* es variable.

En los modos 1 y 3 donde el *baude rate* es variable, este se puede determinar por la razón de sobre carga de los *timers* 1 y/o 2. En el caso en el que el baud rate es generado por el *timer* 1 hay que tener la interrupción por sobre carga deshabilitada, además el valor del *baude rate* estará en función de SMOD (este se encuentra entre los registros de control) como lo muestra la siguiente fórmula.

$$Baud\ rate = (Razon\ de\ sobre\ carga\ timer\ 1)\frac{2^{smode}}{32}$$

Para este caso el *timer* 1 puede operar como contador o como *timer* en cualquiera de sus tres modos. Cuando se utiliza el modo de auto recarga el *baud rate* queda definido por la siguiente fórmula.

$$Baude\ rate = \left(\frac{2^{smod}}{32}\right)\left(\frac{Frecuencia\ de\ oscilación\ del\ micro}{12 \times [256 - TH1]}\right)$$

En la tabla 2.15 se muestran algunos valores comunes de *baude rate* y los valores necesarios de recarga del *timer*1, así como los modos de operación del mismo.

Baud Rate	f_{osc}	SMOD	*Timer* 1		
			C/T	MODE	Valor de Recarga
Mode 0 max 1MHz	12MHz	X	X	X	X
Node 1 max 375K	12MHz	1	X	X	X
Modo 1,3 : 62.5K	12 MHz	1	0	2	FFH
19.2K	11.059MHz	1	0	2	FDH
9.6K	11.059MHz	0	0	2	FDH
4.8K	11.059MHz	0	0	2	FAH
2.4K	11.059MHz	0	0	2	F4H
1.2K	11.059MHz	0	0	2	E8H
137.5	11.986MHz	0	0	2	1DH
110	6MHz	0	0	2	72H
110	12MHz	0	0	1	FEEBH

Tabla 2.15
Valores para el *baud rate* en la comunicación serial

Registro de Control del Puerto Serie (SCON)

En este registro se define el modo de operación, el número de bits a ser transmitido, además de las banderas de recepción y transmisión, así como la habilitación de recepción del microprocesador. La figura figura 2.19 muestra los bits dek registro SCON.

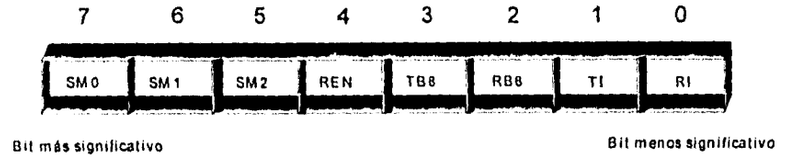


Figura 2.19
Esquema del registro SCON

Los bits SM0 y SM1 sirven para definir el *baud rate* en la transmisión, como se muesta en al tabla 2.16.

SM0	SM1	Mode	Descripción	Baud Rate
0	0	0	Shift reg	$f_{osc}/12$
0	1	1	8 bits UART	variable
1	0	2	9 bits UART	$f_{osc}/64$ o $f_{osc}/32$
1	1	3	9 bits UART	variable

Tabla 2.16
Selección de *baude rate*.

- **SM2.** Habilita la comúnicación del microprocesador en los modos 2 y 3. Si SM2 tiene un valor lógico de 1, entonces la bandera RI no se activará si el noveno dato RB8 es un 0 lógico. En modo 1, la bandera de recepción RI no se activará si no llega un bit de paro válido. En el modo 0 SM2 debe ser 0.

- **REN.** Habilita la recepción serial, ésto se hace por software.

- **TB8.** Este es el noveno dato que será transmitido en los modos 2 y 3.

- **RB8.** Este es el noveno dato que fue transmitido en los modos 2 y 3. En el modo 1 si el bit SM2=1 representa el bit de paro recibido. En el modo 0 no se utiliza.

- **TI.** Es la bandera de transmisión. Puesta por hardware al transmitir el octavo bit en el modo 0 o al transmitir el bit de paro en los otros modos. Debe borrarse por software.

- **RI.** Es la bandera de recepción, puesta por hardware al final del octavo bit en el modo 0 o al recibir el bit de paro en los otros modos, debe borrarse por software.

Hasta ahora se han presentado las características más importantes de los componentes del sistema con el fin de justificar su elección. A continuación se presenta ya la interacción que habrá entre éstos dentro del sistema, así como la descripción precisa de las funciones que desempeñarán.

Capítulo 3

Arquitectura del Sistema

Estructura del Sistema

Cada componente del sistema tiene funciones asignadas que obedecen a los comandos del microcontrolador. Las funciones se efectúan de acuerdo a una organización previa. A continuación se presenta cómo es esta organización y cómo se integran los diferentes dispositivos al resto del sistema.

Mapa de Memoria

En el capítulo anterior se habló ya del concepto de MAPA DE MEMORIA. En sí, un mapa de memoria es la distribución del **área de trabajo** entre los periféricos del sistema, para almacenar y manejar la información. El acceso a los elementos en el mapa de memoria se lleva a cabo por medio de las líneas de direcciones y de control que provienen del microcontrolador. A cada dispositivo le es asignado un rango de direcciones para trabajar, así sólo se habilitará en el

momento en el que el microcontrolador mande su dirección. Las líneas de direcciones provienen de los puertos (dispositivos del microcontrolador para recibir o enviar información) que funcionan como salidas.

La distribución del mapa de memoria para este proyecto se hizo de la siguiente manera:

El microcontrolador trabaja con dos mapas de memoria, un mapa interno y otro externo. El mapa interno tiene una capacidad de 256 Kbytes y controla la información de su memoria RAM interna y la información que se almacena en ella es la de los registros especiales del microcontrolador (SFR). Algunos de estos registros almacenan información proveniente del exterior (de captura), comandos que se deben ejecutar (de control) y resultados de operaciones aritméticas realizadas (aritméticos). Se ofrece también espacio en esta memoria para ser utilizado por el usuario.

El mapa de memoria externo es el más grande y el más importante, pues es el que interactúa directamente con los periféricos del sistema. Este mapa tiene la capacidad de manejar tanta memoria como el número de las líneas de direcciones lo permitan. El modo de direccionamiento que se utilizó se explica a continuación.

Métodos para el Direccionamiento de Memoria

Era necesario hacer que el sistema contara con suficientes líneas de direcciones como para direccionar 1 Mbyte, que es la capacidad de la tarjeta IC. A continuación se describirá cómo puede hacerse uso de una memoria IC de 1 ó 4Mbytes si se cuenta con un sistema cuyo *bus* de datos no alcanzaría a direccionar esta cantidad de memoria.

El manejo en el mapa de memoria se hace de acuerdo a un método que combina dos métodos tradicionales de **mapeo**. El primer método es el que se utiliza a nivel interno del microcontrolador, es decir cuando se accesa directamente a las direcciones de su memoria interna, ROM ó RAM. Este método resulta bueno y rápido pero sólo para los casos en que no se requiera de más memoria que la que ofrecen estos microcontroladores. En otras palabras, el mapeo está limitado a la memoria total del microcontrolador.

El segundo método consiste en el direccionamiento hacia memorias externas (cuando se instalan periféricos que amplían la capacidad de memoria del sistema), por medio de los dispositivos de entrada/salida del microcontrolador como pueden ser los puertos. Por tratarse de más dispositivos, el proceso requiere de más tiempo de acceso.

El método mixto consiste en el manejo de bloques grandes de memoria (mayores al que ofrece el microcontrolador) con un mínimo de líneas en el *bus* de direcciones. Este método utiliza una **ventana ó página movible**. Según el dispositivo que se desee seleccionar, la ventana se recorrerá hacia la zona de direcciones donde dicho dispositivo trabaja. Gráficamente se podría representar de la siguiente manera: (Figura. 3.1)

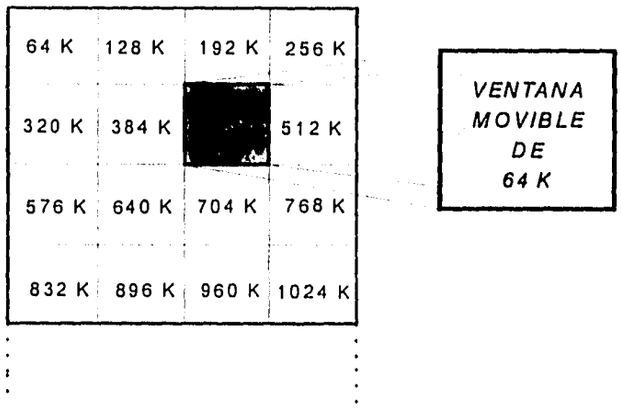


Figura 3.1
Método de direccionamiento por ventanas.

La tabla 3.1 muestra cómo es que dichas páginas se van formando. Con este mapa se pueden direccionar hasta 4 Mbytes de memoria. Las direcciones de la A_0 a la A_{12} son utilizadas para direccionar el dispositivo 1 (D1), el cual será seleccionado siempre y cuando los siguientes tres bits más significativos permanezcan en **cero lógico,** A_{13} , A_{14}, A_{15}. Este direccionamiento abarca en total 8 Kbytes, lo que podría ser utilizado para manejar una RAM ó ROM comercial de 8 Kbytes. En el momento en que el bit A_{13} cambie a **uno**, se habrá sobrepasado el límite de D1 y la cuenta de los anteriores bits (de A_0 a A_{12}) volverá a empezar. Este cambio en A_{13} puede ser generado por un controlador 1. Dicho controlador se encargará de seleccionar entre D1 y otro dispositivo D2. Siempre que A_{13} cambie de cero a uno, la cuenta de los 13 bits anteriores se inicializará. A éste rango del mapa que abarca 8 Kbytes de memoria se dice que es una **ventana** ó **página** movible de 8 Kbytes. Si se colocara un dispositivo decodificador cuyas líneas de entrada fueran de A_{13} a A_{15}, como el controlador 1, entonces a la salida de éste habría 8 combinaciones posibles, lo que significa que podrían colocarse 8 dispositivos de memoria diferentes con espacio cada uno de 8 Kbytes. Así, la página de 8 Kbytes brincaría de un dispositivo a otro.

$A_{23}A_{22}A_{21}A_{20}$	$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}$	$A_{12}A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$	
0 0 0 0	0 0 0 0	0 0 0	0 0 0 0 0	0 0 0 0	0 0 0 0	
0 0 0 0	0 0 0 0	0 0 0	0 0 0 0 0	0 0 0 0	0 0 **1** 0	D1
0 0 0 0	0 0 0 0	. . .	. . . . .	. . . .	. . . .	
0 0 0 0	0 0 0 0	0 0 0	**1 1 1 1 1**	**1 1 1 1**	**1 1 1 1**	
0 0 0 0	0 0 0 0	0 0 1	0 0 0 0 0	0 0 0 0	0 0 0 0	
0 0 0 0	0 0 0 0	0 0 1	**1** 0 0 0 0	0 0 0 0	0 0 1 0	D2
0 0 0 0	0 **0** 0 0	. . .	. . . . .	. . . .	. . . .	
0 0 0 0	0 0 0 0	0 0 1	**1** 1 1 1 1	1 1 1 1	1 1 1 1	
0 0 0 0	0 0 0 0	0 1 0	0 0 0 0 0	0 0 0 0	0 0 0 0	
0 0 0 0	0 0 0 0	. . .	. . . . .	. . . .	. . . .	D3
0 0 0 0	0 0 0 0	0 1 **0**	1 1 1 1 1	1 1 1 1	1 1 1 1	
0 0 0 0	0 0 0 0	. . .	. . . . .	. . . .	. . . .	s/d
0 0 0 0	0 0 0 0	**0 1 0**	**1** 1 1 1 1	1 1 1 1	1 1 1 1	
0 0 0 0	0 0 0 0	0 1 1	0 0 0 0 0	0 0 0 0	0 0 0 0	D6
0 0 0 0	0 0 0 0	0 1 1	1 1 1 1 1	1 1 1 1	1 1 1 1	
. . . .	. . . .	. . .	. . . . .	. . . .	. . . .	s/d
0 0 0 0	**1 1 1 1**	1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	
0 0 0 **1**	0 0 0 0	0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	
. . . .	. . . .	. . .	. . . . .	. . . .	. . . .	D7
0 0 0 **1**	1 1 1 1	1 1 1	1 1 1 1 1	1 1 1 1	1 1 1 1	
. . . .	. . . .	. . .	. . . . .	. . . .	. . . .	
. . . .	. . . .	. . .	. . . . .	. . . .	. . . .	s/d
1 1 1 1	1 1 1 1	1 1 1	1 1 1 1 1	1 1 1 1	1 1 1 1	
control 2		control 1	posibles	direcciones	para página	

s/d Sin dispositivo designado a la dirección.

Tabla 3.1.
Manejo del mapa de direcciones por ventanas.

Depende del diseñador qué bit ó bits sean los que se usarán para controlar los cambios de la página y también el tamaño que tendrá ésta.

En el siguiente mapa de memoria se muestra cómo se forman las páginas de 8 Kbytes y finalmente el de 1 Mbyte:

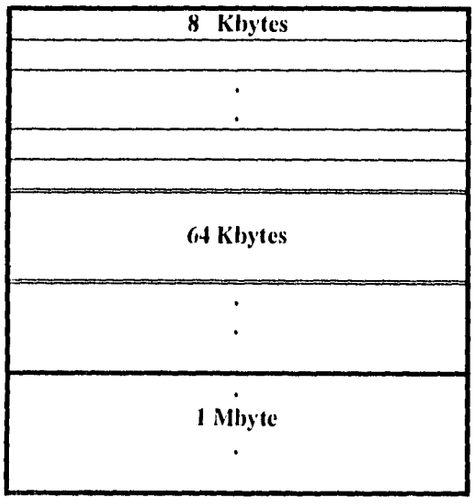


Figura 3.2.
Ventanas utilizadas

El controlador asigna turno de operación al dispositivo D1, D2, etc, ya que dos dispositivos no pueden operar en el mismo espacio de memoria ni al mismo tiempo. En el mapa se aprecia cómo se puede controlar la tarjeta IC a partir del controlador 2. Si la tarjeta es de 1Mbyte es obvio que necesitará 1 Mbyte de espacio en el mapa de memoria, así como la memoria RAM de 8 Kbytes necesitó 8 Kbytes en el mapa. La tarjeta IC deberá tener su propio espacio de 1 Mbyte sin ser utilizado por ningún otro dispositivo. El resto de los periféricos, que para este caso constan de una capacidad de memoria mucho menor, pueden ser distribuidos sin problema en otra ventana de 1 Mbyte. Así, si el diseñador designa a A_{20} en estado lógico **alto** para la tarjeta IC y en **bajo** para el resto de los periféricos, para poder accesar al dispositivo D1, A_{20} debe estar en **alto**, A_{15}, A_{14} y A_{13} en cero y A_{12} **no importa** en qué estado se encuentre.

De lo anterior se puede deducir que el controlador 1 depende a la vez del controlador 2, ya que el controlador 1 sólo se encarga de los periféricos de menor capacidad de memoria.

En la tabla 3.1 se puede observar que las combinaciones en las direcciones de A_0 a A_{12} se repiten durante todo el mapa de memoria. Sin embargo, este hecho no afecta la habilitación de

los dispositivos existentes ya que es el diseñador quién decide con qué combinación de bits en los controladores se activará el dispositivo. El diseñador también decide dónde colocará a los controladores dentro del mapa.

Cabe mencionar que en el mapa, que no necesariamente se ocupan todas las direcciones, ya que puden haber páginas que no sean utilizadas nunca (s/d). Además estas páginas **en blanco** podrían ser espacios reservados para posteriormente agregar algún otro componente al sistema.

Asignación de Direcciones en los Elementos del Sistema

El microcontrolador consta de 3 puertos que al funcionar como salidas es posible utilizarlos como líneas de direcciones hacia los periféricos.

Al utilizar memorias externas, el puerto 0 (P0) del microcontrolador funciona como *bus* de direcciones y datos. Este puerto está **multiplexado** y el microcontrolador se vale de un *latch* externo para separar las direcciones de los datos. P0 maneja las primeras ocho direcciones (de A_0 a A_7) que corresponden a la parte baja del *bus* de direcciones. El puerto 2 (P2) se encarga de las direcciones de la parte alta que van de A_8 a A_{15}. El puerto 4 (P4) de la A_{16} a la A_{20} ya que sólo se van a direccionar 2 Mbytes. En la tabla 3.2 se muestra la utilización de los puertos del microcontrolador para el manejo de dispositivos.

Puerto de salida:	P4	P2	P0 y P2
Líneas de direcciones:	A_{23} ..... A_{16}	A_{15} .....A_{13}	A_{12} ..... A_0
Página:	1 Mbyte		8 Kbytes
Dispositivos que se controlan:	Tarjeta IC / resto de los perif.	entradas al Decodificador	RAM, ROM RTR, *display*

Tabla 3.2
Manejo de puertos para direccionar los dispositivos.

El mapa de memoria está dividido en dos páginas de 1 Mbyte. En la primer página trabaja la tarjeta IC y en la segunda página lo hacen el resto de los periféricos. La elección de cualquiera de las dos páginas de 1 Mbyte es por medio de la línea de dirección A_{20} que corresponde al bit 4 de P4 (P4.4). Cuando éste está en un nivel bajo (cero lógico), se activa la primer página de 1 Mbyte donde trabaja la tarjeta IC. Cuando el estado lógico es un **1**, se habilita un decodificador de tres entradas que se localiza en la segunda página de 1 Mbyte, el cual se encarga de seleccionar uno de los demás periféricos: la memoria RAM, el reloj de tiempo real (RTR) ó el *display*.

Las direcciones que controlan las entradas al decodificador son: A_{13}, A_{14} y A_{15}. De las ocho posibles combinaciones en las entradas del decodificador, sólo interesan tres, por lo tanto, de la salida sólo se tomarán las tres primeras líneas de selección. A éstos se les ha denominado Y_0, Y_1 y Y_2, controlando respectivamente, a la RAM, al RTR y al *display*. La tabla siguiente muestra cómo está organizado el decodificador para seleccionar los dispositivos.

DISPOSITIVO	ENTRADA A15	ENTRADA A14	ENTRADA A13	SALIDA
RAM	0	0	0	Y0
RTR	0	0	1	Y1
DISPLAY	0	1	0	Y2

Tabla 3.3
Selección de salidas de decodificador.

Cada uno de los dispositivos trabaja en una página de 8 Kbytes. El resto de las combinaciones en las entradas al decodificador podrían dar lugar a la elección de las siguientes páginas de 8 Kbytes en el mapa. Sin embargo, sólo hay cuatro dispositivos (dos de ellos asignados a las mismas combinaciones) distribuídos en las primeras tres páginas de 8 Kbytes, el resto no tiene dispositivos asignados, y aunque se dieran las combinaciones a la entrada en el decodificador, no se alteraría el estado de los dispositivos existentes. Lo que también representa una garantía en el manejo separado de los mismos, porque no interfieren entre sí. Se puede decir que el mapa tiene espacios **reservados** para otros dispositivos. De esta forma el mapa de memoria del sistema queda conformado de la siguiente manera.

DIR	**MAPA DE DATOS**	**MAPA DE INSTRUCCIONES**
1 M	TARJETA IC	
8 K	RAM	ROM
8 K	RTR	
8 K	*DISPLAY*	
2 M	. . .	

Tabla 3.4
Mapa de memoria del sistema

En la figura 3.3 se muestra un diagrama a bloques de la arquitectura del microcontrolador.

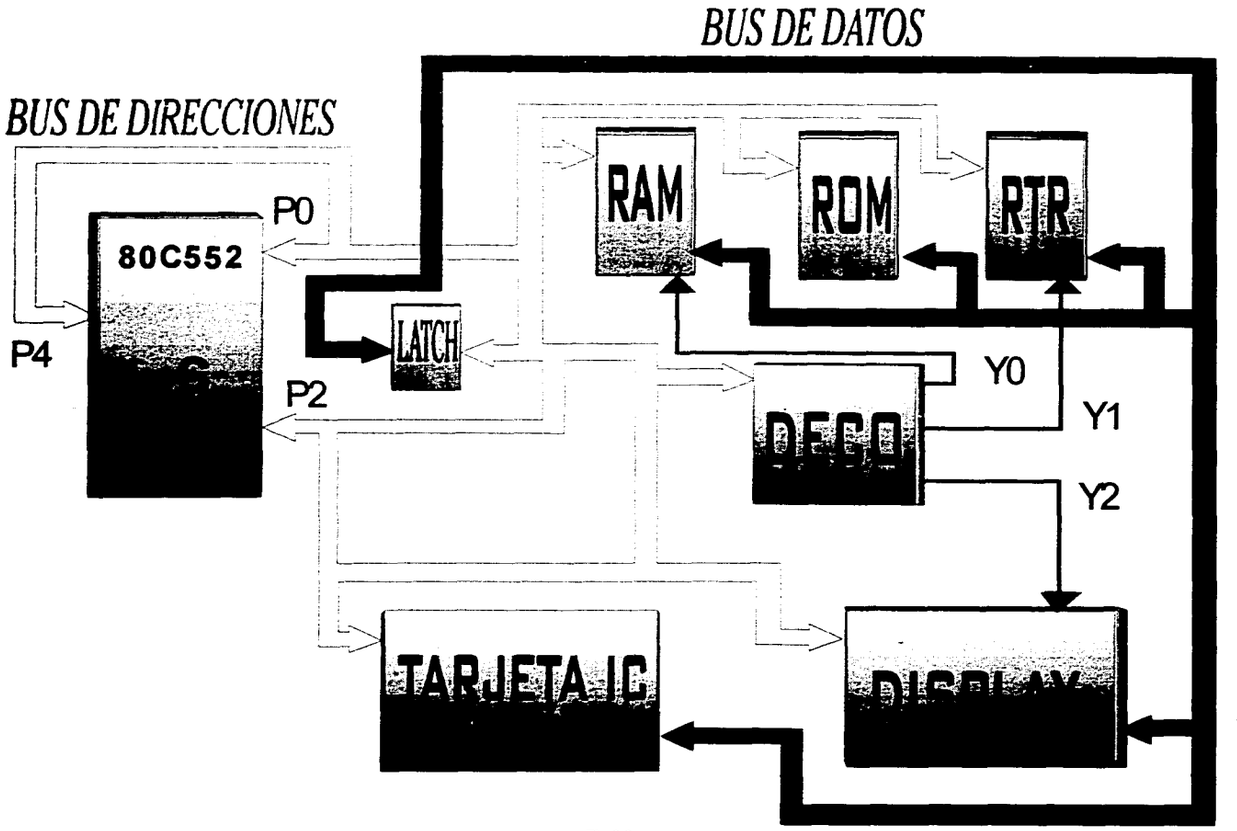


Fig 3.3
Arquitectura abloques del microcontrolador.

La ROM

El microcontrolador lee la memoria ROM externa para ejecutar el programa que el diseñador grabó en ella. Para ello, se vale de P0 y P2. Cuando el microcontrolador lee una instrucción de la ROM, lo hace poniendo la **parte baja**, de A_0 a A_7 (los 8 bits menos significativos) de la dirección en P0, que significan los primeros 256 bytes. En el momento en que P0 tiene la parte baja de las direcciones, otra terminal del microcontrolador llamada ALE (*Adress Latch Enable*) habilita un *latch* externo que retiene las direcciones hasta que se manden en conjunto con la **parte alta**, de A_8 a A_{15} (los 8 bits más significativos), la cual se encuentra en el puerto 2, en este momento P0 se pone en "flotación" esperando leer la instrucción o el dato. Una vez que P2 tiene la parte alta de las direcciones, se activa entonces la terminal del microcontrolador llamada PSEN (*Program Store Enable*) que a su vez, habilita a la memoria ROM externa (en la terminal OE de esta última) quién recibe el conjunto de las direcciones de A_0 a A_{12}. La cuenta de las direcciones la lleva un dispositivo interno del microcontrolador llamado *Program Counter*.

La ROM una vez habilitada, recibe los bits de las líneas de direcciones y según lo que se encuentre guardado en dichas direcciones, será proporcionado un dato o una instrucción por las 8 líneas de salida de la misma (de AD_0 a AD_7). Estas líneas están conectadas al *bus* de datos que se multiplexa en P0.

La RAM

El microcontrolador utiliza señales diferentes para accesar a una RAM o a una ROM. Esto significa que la RAM y la ROM no están en el mismo mapa. Se manejan dos mapas de memoria: para datos (RAM) y para instrucciones (ROM). Lo anterior permite que la memoria RAM sea leída y escrita dentro de la misma página que la ROM sin que interfieran una con la otra. El microcontrolador decide qué función realizará la RAM mediante las terminales RD para leer y WR para escribir. Estas terminales son independientes de PSEN, y sólo se activan una a la vez. Las líneas de direcciones de entrada a la RAM provienen también de P0 y P2 del microcontrolador. El *bus* de datos se maneja de la misma forma como en la ROM.

El RTR

El reloj de tiempo real (RTR) es un dispositivo en el cual se puede leer y escribir información y se encuentra en el mapa de memoria de datos (RAM). Este elemento se sitúa en el

segundo bloque de 8K dentro de la segunda ventana de 1 Mbyte de periféricos, es habilitado por la línea Y1 del decodificador. Debido a su diseño interno se conectó directamente al *bus* de datos del µC, utilizando las terminales de RD y WR para distinguir entre lectura y escritura. Además, el microcontrolador le proporciona la señal de ALE para que pueda distinguir entre direcciones y datos. El acceso a este dispositivo es igual que a cualquier tipo de memoria RAM.

Para la operación del bloque de identificación y almacenamiento es importante el no perder la información sobre la hora y fecha, por lo que se utilizó el circuito de respaldo de polarización propuesto por el fabricante, del cual se hace referencia en el capítulo anterior fig. 2.14. El reloj de tiempo real quedó dentro del mapa de memoria como se muestra en la tabla 3.5.

Reloj de Tiempo Real	
Dir HEX	Registro
102000	Segundos
102002	Minutos
102004	Horas
102006	Dias de la Semana
102007	Dia del Mes
102008	Mes
102009	Año
10200A	Reg A
10200B	Reg B
10200C	Reg C
10200D	Reg D
10200E . 10203F	50 Bytes RAM
102040 . . 103FFF	. . . . .

Tabla 3.5
Direcciones del reloj de tiempo real

Este dispositivo puede ser actualizado mediante un teclado, para no depender directamente de una PC y que el usuario pueda hacerlo en el lugar donde se encuentra el sistema de identificación y almacenamiento. Este tecaldo consta de seis funciones que son: actualizar hora y fecha, incrementar hora o fecha, movimiento del cursor a la deracha o izquierda, aceptar la información actual y una función de escape. El teclado se conectó al puerto 5 (P5) del µC que puede ser utilizado para entradas digitales. Las líneas ocupadas fueron de P5.0 a P5.5. La configuración de este teclado se mustra en la figura 3.4. El arreglo RC se tiene para evitar **rebotes** del **push-button**, la constante de tiempo de este arrglo se calculó a partir de una señal vista en el osciloscopio, cuyos transitorios duraban aproximadamente de 30 a 40µS.

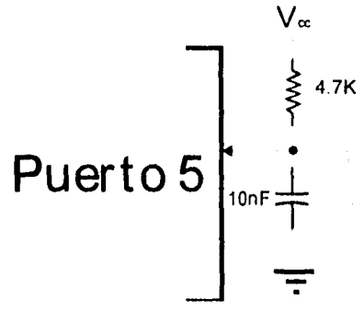


Fig 3.4
Diagrama de conexión del teclado.

El LCD

La pantalla de cristal líquido (LCD) se encuentra en el tercer bloque de 8 Kbytes de la segunda ventana de 1 Mbyte, por lo que es controlado mediante la línea Y2 del decodificador, este dispositivo se maneja como una memoria de tipo RAM. Se conectó directamente al *bus* de datos del µC, pero sus líneas de control tuvieron que ser manejadas mendiante una lógica combinacional, ya que las señales para efectuar una lectura o escritura en el *display* están defasadas con respecto a las del µC, además para poder escribir un dato en el *display* es necesario mandar primero un comando y enseguida el dato, teniendo que accesar a diferentes registros en él. El acceso de estos registros se realiza mediante las líneas RS y R/W del *display*. Estas líneas fueron controladas mediante las líneas de direcciones A8 y A9 respectivamente, teniendo direcciones diferentes para cada registro en el *display*. El lugar que ocupa el LCD dentro del mapa de memoria queda definido de la siguiente manera:

LCD		
Dir HEX	A9A8	Registro o Acción
104000	0 0	IR
104100	0 1	BS
104200	1 0	DR (escritura)
104300	1 1	DR (lectura)
.. 105FFF	. .	.

Tabla 3. 6
Direcciones de la pantalla de cristal liquido.

La línea de habilitación E tuvo que ser manipulada con lógica combinacional debido a la duración de esta señal y al orden en que se presenta. Para enteder lo anterior se muestra el diagrama de escritura del *display*:

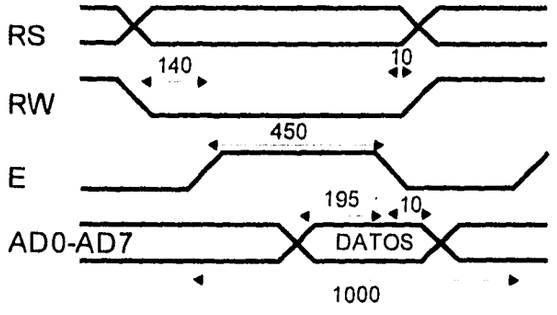


Fig 3.5
Diagrama de tiempos de escritura del *display*.

El diagrama de escritura del μC es el siguiente:

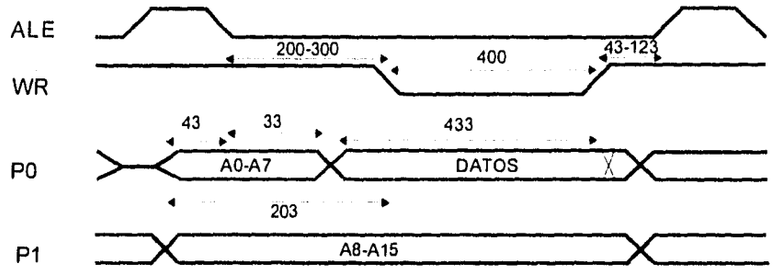


Fig 3.6
Diagrama de tiempos de escrittra del microcontrolador.

Si se conectara el *display* en forma directa, es decir, la señal de WR del microcontrolador a la señal R/W del *display* y la señal Y2 de decodificación a la terminal E del *display*, se tendría el siguiente diagrama (figura 3.6), donde se observa que la duración de la señal de habilitación para el *display* dura más tiempo del que necesita para recibir la información y dicha señal se mantiene después de que los datos han cambiado por direcciones y así no se asegura que el *display* toma los datos indicados.

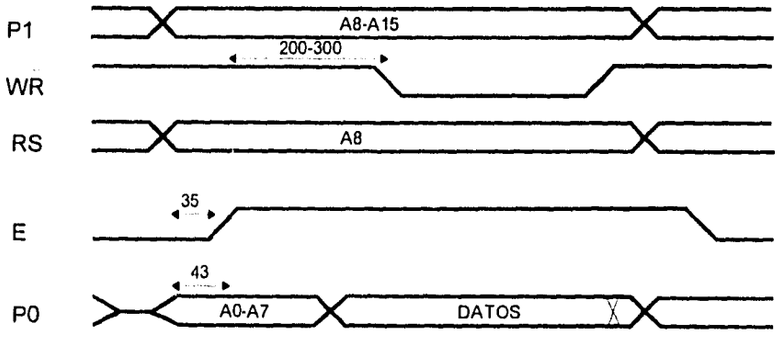


Fig 3.7

Utilizando un arreglo de compuertas 74LS00, con un retraso de 10 a 15 nS y las señales de RD, WR y Y2 (fig 3.8) se tiene el siguiente diagrama de señales (fig. 3.9), donde la duración

de la señal de habilitación del *display* es mayor y se presenta en el momento necesario para que tome los datos que se le envían al *display*.

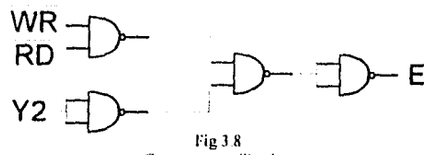


Fig 3.8
Compuertas utilizadas.

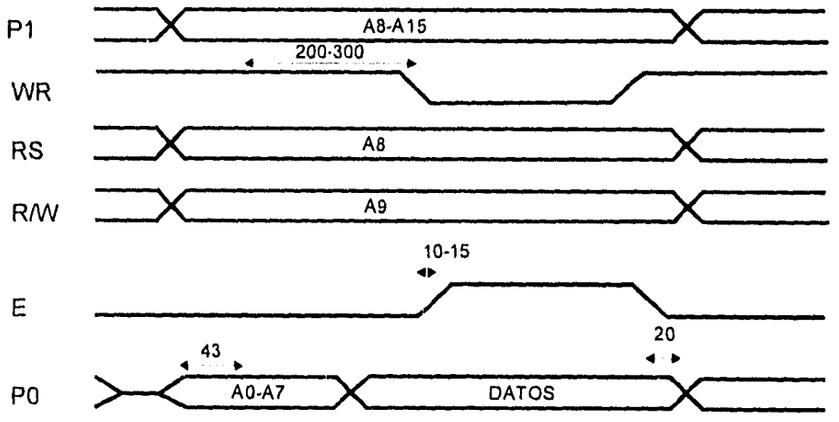


Fig 3.9
Diagrama de tiempos de escritura despues de pasar por compuertas.

Comunicación Serial

Para la comunicación serial se utizó una interfaz RS-232, la cual recomienda una comunicación fiable hasta una distancia de 15 metros máximo. Se eligió esta interfaz debido a que el lugar donde se colocará el sistema puede estar cerca de una comuputadora personal para interactuar con él.

Las características elécricas de la interfaz RS-232 son: niveles de voltaje de +12V y -12V

Para cumplir con las características eléctricas demandadas por esta interfaz se eligió el IC MAX-232, el cual sólo necesita una fuente de 5 volts para dar los niveles de +12 y-12 volts, requeridos por la interfaz para la transmisión ;de forma análoga convierte estos niveles a sus correspondientes niveles de 0 y 5V en la recepción, esto evita tener fuentes de +12 y -12V, además de eliminar un IC, ya que anteriormente se tenía un IC para obtener un nivel de +12V y otro para -12V, donde cada IC convertía el voltaje de la interfaz a niveles de TTL/CMOS y viceversa.

La forma de conexión entre el sistema de identificación y la PC se muestra en la figura 3.10. Para efectuar esta conexión se utilizaron conectores DB9 para la PC y el sistema de identificación, el estándar marca que deben usarse conectores DB25 para la PC, pero se ha observado que las computadoras actuales los puertos seriales vienen con conectores DB9.

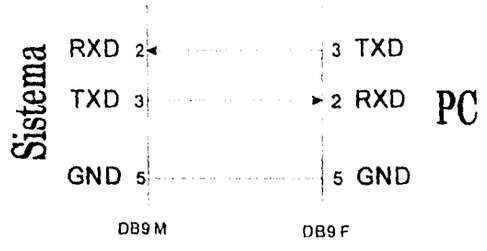


Fig 3.10
Conexiones del puerto serie.

Lector Optico

El lector óptico se conectó directamente a la terminal P1.0 (CT0) del microcontrolador, esto fue ya que la salida digital del lector óptico es capaz de proporcionar la corriente necesaria a la terminal del microcontrolador, así como los niveles de voltaje. El lector óptico necesitó de líneas de polarización y tierra para poder operar, ya que contiene un comparador de nivel que convierte la señal analógica a una señal digital, además de necesitar polarización para su fuente de luz.

Este lector tiene una terminal que sirve para encender un LED e indicar cuando se tiene una lectura correcta, esta línea no se utilizó ya que se generó una señal luminosa y audible para cuando la lectura es incorrecta y una señal luminosa cuando la lectura es correcta.

La Tarjeta IC

Como se mencionó antes, la página de 1 Mbyte donde trabaja la tarjeta IC se habilita con P4.4 en estado lógico bajo. En la siguiente figura se muestra como se conectó la tarjeta IC.

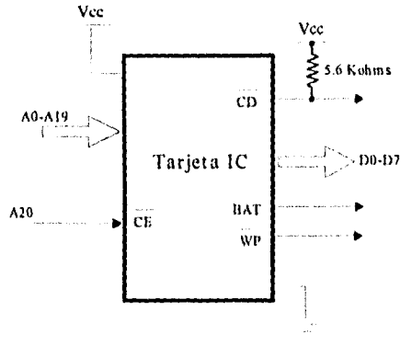


Fig 3.11
Diagrama de conexiones de la tarjeta IC.

El *bus* de direcciones de los puertos 0 y 2 no era suficiente para accesar toda la capacidad de la tarjeta IC ya que sólo direccionan hasta 64 Kbytes. Las líneas de direcciones que hacían falta para direccionar 1 Mb se tomaron también del P4, estas fueron P4.0 a P4.3 (A16-A19). Por lo que se manejaron páginas de 64Kbytes controladas por las líneas del P4, esto significa que al tratar de escribir a este dispositivo hay primero que especificar en qué página de 64K se escribirá la información. Para las líneas de datos se conectó directamente el *bus* de datos del µC.

Esta tarjeta cuenta con tres terminales, las cuales permiten sensar si la tarjeta está presente en el sistema, si tiene puesta la protección contra escritura y si su batería interna está descargada.

La terminal que sensa la presencia de la tarjeta tiene un nivel de 0V cuando la tarjeta está en el sistema, ya que está conectada a tierra internamente. Por la razón anterior se conectó una resistencia a V_{cc} para asegurar que al no estar presente la tarjeta, el voltaje en la terminal sea de 5 V. El valor de resistencia es de 5.6 K, circulando por ella una corriente de 0.89 mA, que es menor a la corriente máxima que puede manejar dicha terminal (5 mA).

Cuando la terminal que sensa el voltaje de la batería tiene 0V quiere decir que hay que reemplazar la batería y cuando esta terminal tiene 5V la batería esta en buenas condiciones.

Cuando la terminal que sensa la protección contra escritura tiene un nivel lógico de 0 quiere decir que la protección contra escritura esta puesta, cuando el nivel lógico es de 1 la protección contra escritura no está puesta, esta terminal se conectó a la terminal del μC T1, esta terminal se pude accesar por bit (puesto que sólo se desea conocer el estado lógico de la terminal y no del puerto completo).

Al conectar esta tarjeta se tuvieron algunos problemas como: al insertar la tarjeta a su conector o al sacarla del mismo, el sistema se **perdía**, tendía a inicializarse o ejecutaba otra parte del programa de aplicación. Esto se produjo por alteraciones en los niveles de voltaje (*glitches*), ya que la tarjeta está conectada a las líneas de direcciones y datos del μC así como a las líneas de polarización y tierra. Haciendo una búsqueda en la literatura se encontró que las líneas más sensibles a este tipo de ruido son las líneas de polarización y tierra, ya que éstas tienen acceso a todos los lugares en una tarjeta de circuito.

Se encontraron varias soluciones para este problema. La primera solución es minimizar la distancia existente entre las líneas de polarización y tierra, esto se puede lograr manejando un plano de tierra y un plano de polarización. La segunda opción es mantener lo más cerca posible las líneas de polarización y las líneas de tierra. Una tercera opción es crear una red de tierra, teniendo así una trayectoria cerrada para la circulación de corriente. Todas estas soluciones tienen como finalidad evitar que se cierre un circuito para la circulación del **sobrevoltaje** ó **sobrecorriente**, aunque esto no siempre se consigue, una cuarta opción es la utilización de capacitores de desacoplamiento, estos tienen como finalidad absorber la energía necesaria para soportar el sobre voltaje.

Existen dos tipos de capacitores de desacoplamiento, los que desacoplan a los circuitos integrados y los que desacoplan tarjetas completas. Los capacitores para circuitos integrados manejan un valor de 0.1μF a 1μF, generalmente cerámicos y colocados lo más cerca del circuito integrado, mientras que los capacitores para desacoplar tarjetas manejan valores de 10 a 100μF, generalmente electrolíticos, en éstos el lugar del capacitor no es tan crítico. El valor de ambos capacitores está en función de los tiempos de duración de estos sobrevoltajes.

El primer capacitor utilizado fue un capacitor de desacoplamiento de 100μF electrolítico, con éste se atenuó en gran medida el ruido. Se hicieron pruebas con otros capacitores seleccionando finalmente uno de 33μF de tantalio, el cual dió el mejor resultado. A continuación se muestra cómo ayuda el capacitor a manejar estos sobrevoltajes.

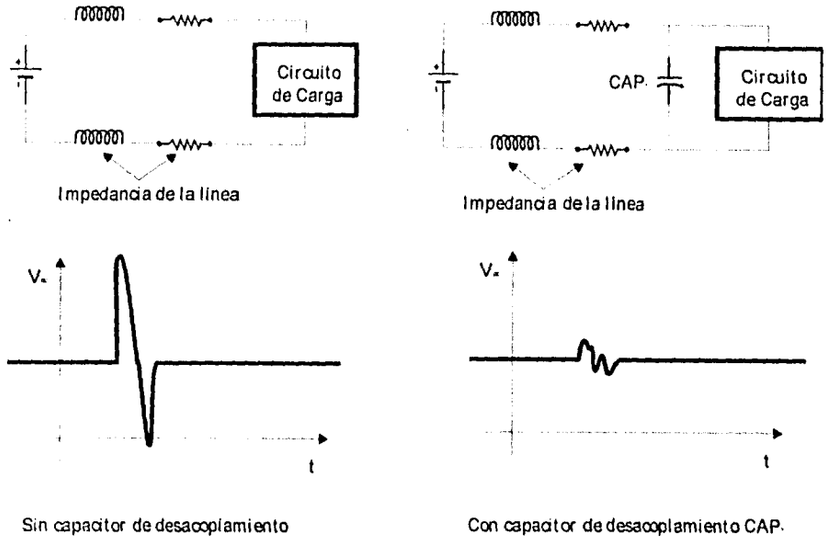


Fig 3.12
Efectos de los capacitores de desacoplamiento.

Fuente de Alimentación

Todos los elementos del bloque de identificación necesitan un voltaje de polarización de 5V, el cual es suministrado por una fuente de alimentación. La fuente de alimentación del bloque de identificación está basada en un regulador LM7805 que mantiene un nivel de 5V en la salida si se tiene un voltaje de 7V a 25V en la entrada, lo cual ofrece cierta protección contra sobre-voltajes, además ofrece protección contra cortos circuitos (esta protección está basada en la temperatura del regulador limitando la corriente de salida). La regulación de este elemento es suficiente ya que el elemento más sensible al rizo en la fuente es el lector óptico. El factor de rizo máximo que soporta el lector óptico para operar es de $150mV_{pp}$.

En la siguiente tabla se muestran algunas de sus características eléctricas del regulador.

Símbolo	Parámetro	Condiciones		Min	Típico	Máximo	Unidades
Vo	Voltaje de Salida	T_j=25° C		4.8	5.0	5.2	V
V_R línea	Regulación de línea	T_j=25° C	7.0V≤V_i≤25V, I_O=200mA		3.0	100	mV
			8.0V≤V_i≤20V, I_O=200mA		1.0	50	
Vo	Voltaje de Salida	7.0≤V_i≤20V 5.0≤I_O≤350mA		4.75		5.25	mV
V_R Carga	Regulación con resp. a la Carga	T_j=25° C	5.0≤I_O≤500mA		20	100	mV
			5.0≤I_O≤200mA		10	50	
$\Delta V_O/\Delta T$	Sensibilidad de la salida con resp. a la temp.	I_O=5mA			1.0		mV/°C

Tabla 3.8
Característica eléctricas del regulador 7805.

El diseño de una fuente de alimentación se puede dividir en una etapa de acondicionamiento de voltaje, una etapa de rectificación, una etapa de filtrado y una etapa de regulación (fig 3.13). Para la etapa de reducción de voltaje se utilizó un transformador con relación de transformación 127:12, lo cual da un voltaje pico de 16.97. Para la etapa de rectificación se utilizó un puente de diodos con una capacidad de corriente de 2 A. En la etapa de filtrado se utilizó un capacitor de 330μF, esto garantiza un rango de voltaje para la correcta operación del regulador de voltaje.

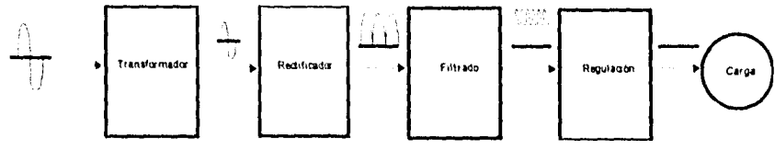


Figura3.13
Diaframa a bloques de la fuente de alimentación.

En la fig 3.14 se observa que el voltaje mínimo de entrada ($V_{min\text{-}ent}$) es el menor voltaje que deberá entrar al regulador, por tanto este debe conservar un nivel que permita asegurar la buena operación del regulador. El voltaje de dc (V_{dc}) es la componente de directa a la silida de la etapa de filtrado. El voltaje de rizo pico a pico (V_{rpp}) es la componente de alterna a la silida de la etapa de filtrado desde su valor máximo hasta su valor mínimo, el voltaje de rizo pico (V_{rp}) es la mitad del voltaje rizo pico a pico, el voltaje pico es el voltaje máximo a la salida de la etapa de filtrado (V_p).

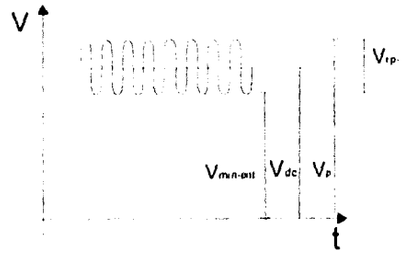


Fig 3.14
Señal de salida de la etapa de filtrado.

Escogiendo un voltaje de 13 V como entrada mínima al regulador, se tiene:

$$_{min\text{-}ent} = V_p - 2V_{rp}$$

Donde I_{cd} es la componente de corriente diecta que consume la carga y que circula a la salida de la etapa de filtrado.

$$_{rp} = \frac{(V_p - V_{min\text{-}ent})}{2}$$

$$_{rp} = 1.985V$$

Se sabe que el voltaje RMS a la salida de la etapa de filtrado es:

$$_{r(RMS)} = \frac{2.4\,I_{cd}}{C}$$

$$C = \frac{2.4 I_{cd}}{V_{r(RMS)}}$$

$$C = 314.081\mu F$$

Como no exiten capacitores de este valor se utilizó un capacitor de 330μF. Los valores obtenidos de voltaje fueron: $V_{min\text{-}ent}$=13.192V y Vr_p=1.9V.

Como se mencionó antes el regulador sólo puede garantizar una buena operación en un rango de voltajes, por lo que hay que proteger al dispositivo contra sobre voltajes y además proteger a los componentes del sistema contra sobre corrientes, para tal efecto se eligió la siguiete configuración, que se conectó como etapa previa al regulador.

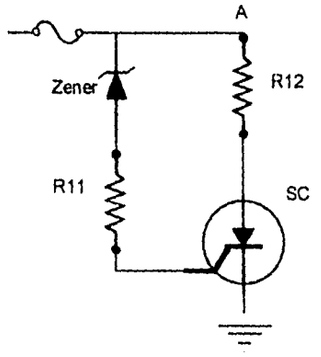


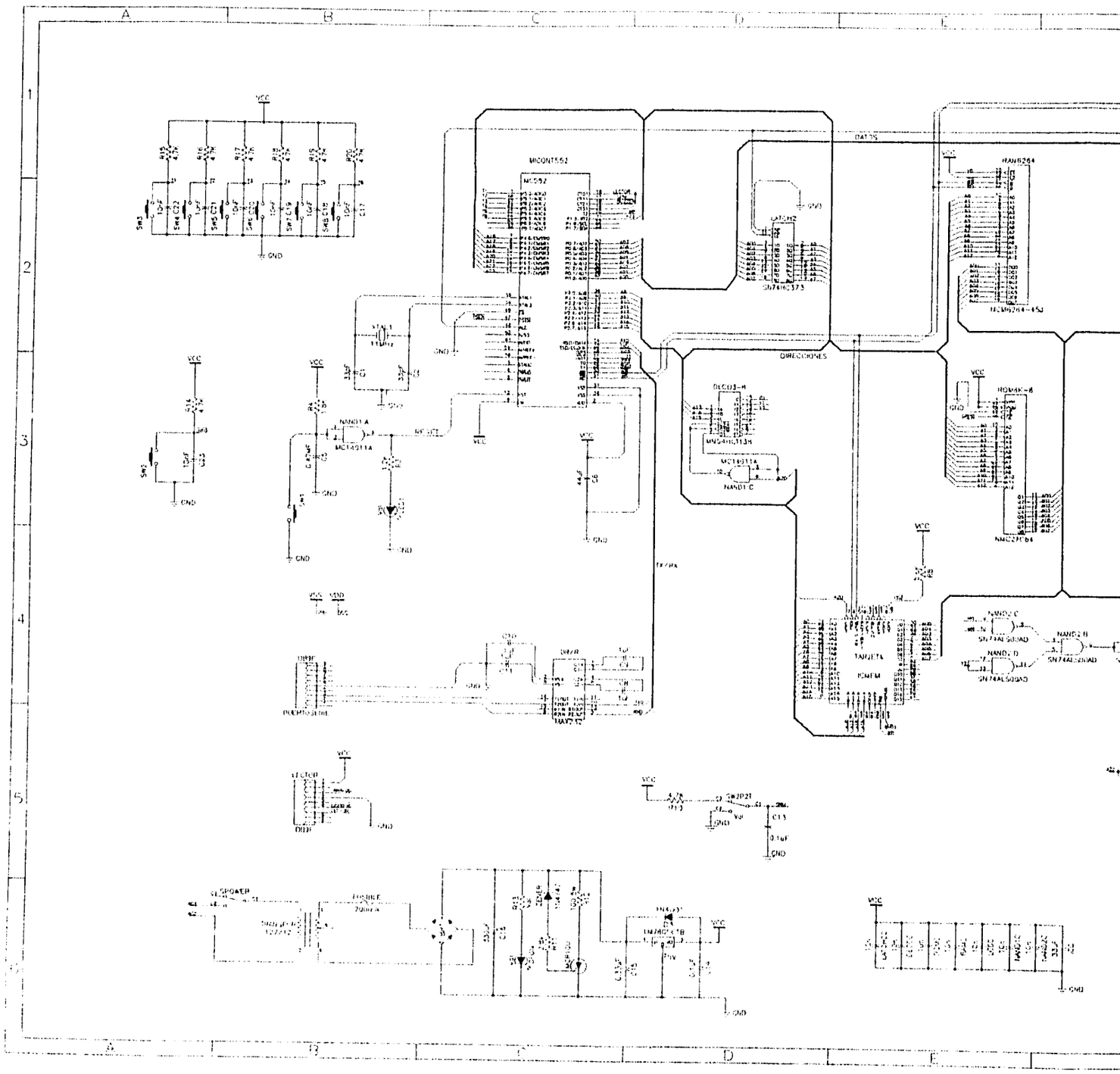
Fig 3.15
Protección contra sobre corrientes y sobre voltajes.

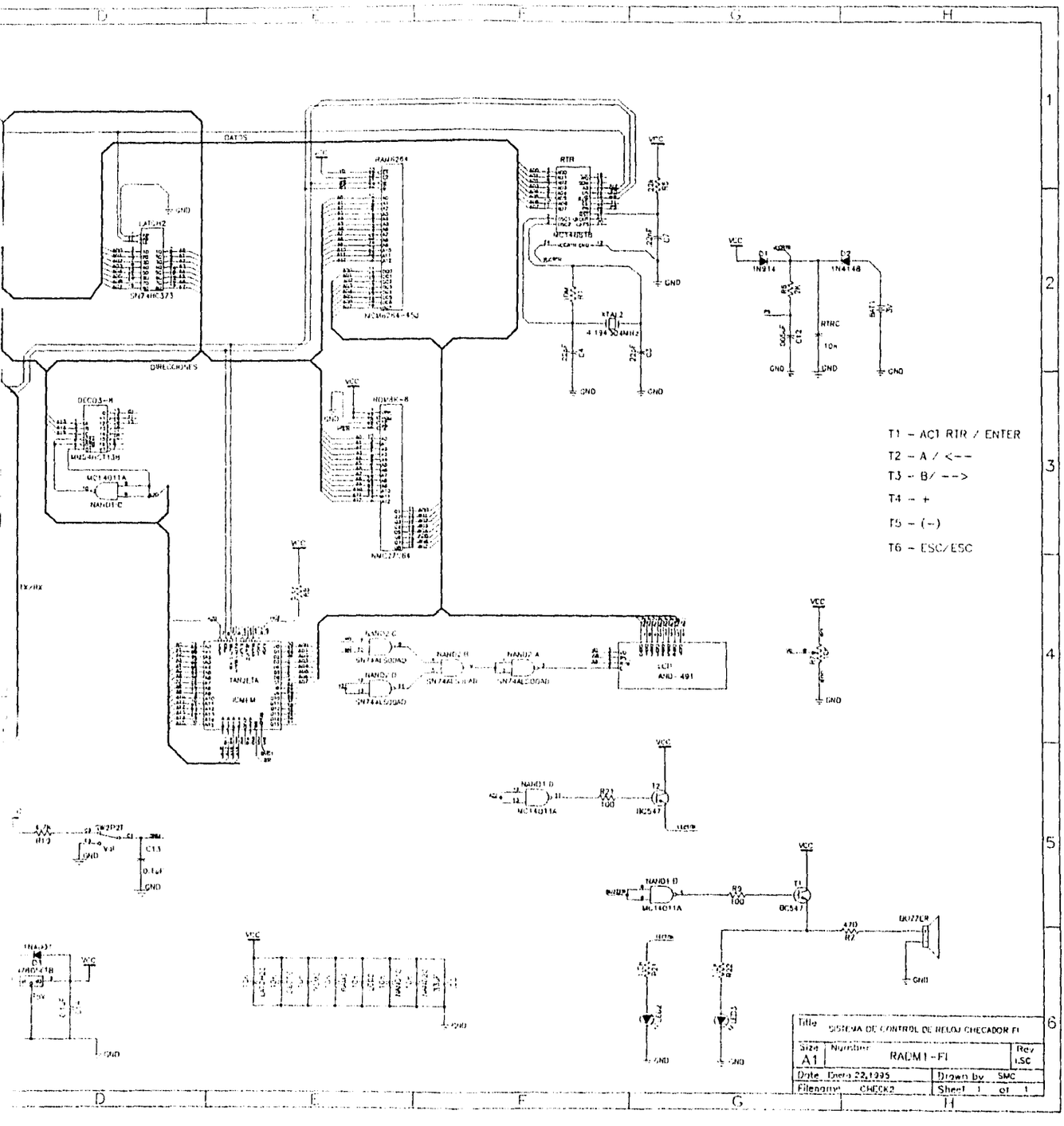
Esta configuración protege mediante un fusible, contra sobre corrientes y sobre voltajes gracias al arreglo formado entre el SCR, el diodo zener y las reistencias. Esta protección consiste en un fusible de 200 mA. El amperaje del fusible se escogió en base al consumo promedio del circuito que es aproximadamente de 150 mA. Esta corriente es menor que la corriente máxima que soporta el regulador, con esto se protege a los dispositivos, ya que en caso de existir un corto circuito, el regulador seguiría suministrando corriente. Para tener una buena regulación, el

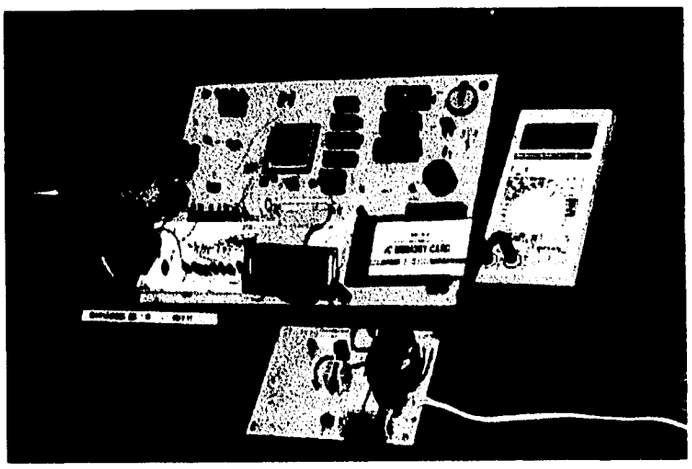
voltaje máximo en el regulador debe ser de 25V, así que se escogió un voltaje de 20V como voltaje máximo, con este voltaje y suponiendo una corriente de 200mA, se calculó una resistencia para abrir el fusible cuyo valor fue de 100Ω, protegiendo así al regulador. Para este efecto el sobre voltaje debe vencer primero el voltaje de encendido del diodo zener que es de 20V, en esta forma se controla el encendido del SCR, la resistencia 11 se utiliza para limitar la corriente de entrada a la compuerta del SCR.

El diagrama completo de la etapa de identificación y almacenamiento se presenta en la siguiente hoja. También se muestran fotografías del prototipo y el acabado final del sistema.

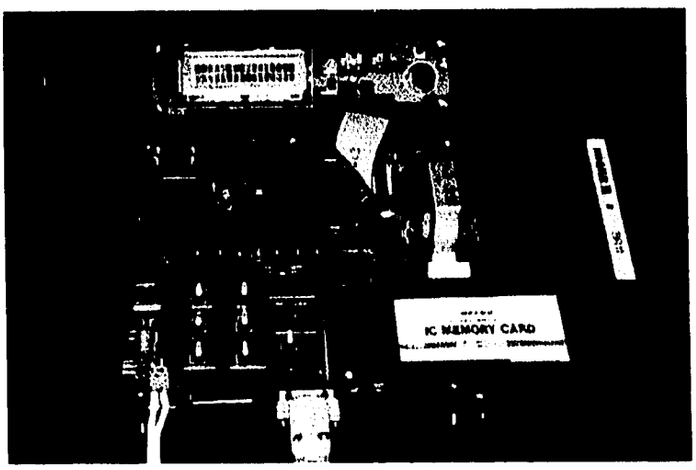
De esta forma se tiene la arquitectura del sistema de identificación y almacenamiento, la forma en que los dispositivos operan entre sí. En el siguiente capítulo se explicará la forma en que programan los dispositivos para que interactúen entre sí.


A
B
C
D
E
1
2
3
4
5
VCC
GND
MICONT552
MC552
DATOS
DIRECCIONES
SN74HC373
HM6264
NMC27C64
ROMAK-8
MN54HC138
MC14011A
NAND1:A
NAND1:B
NAND1:C
NAND1:D
SN74ALS00AD
TARJETA
PCMEM
TX/RX
BUZZER
DRIVER
PIEZOBUZZER
SW2P2T
LM7805CT
1N4007
1N4148
SW1
SW2
SW3
SW4
SW5
SW6
SW7
SW8
SW9
SW10
SW11
SW12
SW13
SW14
SW15
SW16
SW17
SW18
SW19
SW20
SW21
SW22
SW23
SW24
SW25
SW26
SW27
SW28
SW29
SW30
SW31
SW32
SW33
SW34
SW35
SW36
SW37
SW38
SW39
SW40
SW41
SW42
SW43
SW44
SW45
SW46
SW47
SW48
SW49
SW50
SW51
SW52
SW53
SW54
SW55
SW56
SW57
SW58
SW59
SW60
SW61
SW62
SW63
SW64
SW65
SW66
SW67
SW68
SW69
SW70
SW71
SW72
SW73
SW74
SW75
SW76
SW77
SW78
SW79
SW80
SW81
SW82
SW83
SW84
SW85
SW86
SW87
SW88
SW89
SW90
SW91
SW92
SW93
SW94
SW95
SW96
SW97
SW98
SW99
SW100
SW101
SW102
SW103
SW104
SW105
SW106
SW107
SW108
SW109
SW110
SW111
SW112
SW113
SW114
SW115
SW116
SW117
SW118
SW119
SW120
SW121
SW122
SW123
SW124
SW125
SW126
SW127
SW128
SW129
SW130
SW131
SW132
SW133
SW134
SW135
SW136
SW137
SW138
SW139
SW140
SW141
SW142
SW143
SW144
SW145
SW146
SW147
SW148
SW149
SW150
SW151
SW152
SW153
SW154
SW155
SW156
SW157
SW158
SW159
SW160
SW161
SW162
SW163
SW164
SW165
SW166
SW167
SW168
SW169
SW170
SW171
SW172
SW173
SW174
SW175
SW176
SW177
SW178
SW179
SW180
SW181
SW182
SW183
SW184
SW185
SW186
SW187
SW188
SW189
SW190
SW191
SW192
SW193
SW194
SW195
SW196
SW197
SW198
SW199
SW200
SW201
SW202
SW203
SW204
SW205
SW206
SW207
SW208
SW209
SW210
SW211
SW212
SW213
SW214
SW215
SW216
SW217
SW218
SW219
SW220
SW221
SW222
SW223
SW224
SW225
SW226
SW227
SW228
SW229
SW230
SW231
SW232
SW233
SW234
SW235
SW236
SW237
SW238
SW239
SW240
SW241
SW242
SW243
SW244
SW245
SW246
SW247
SW248
SW249
SW250
SW251
SW252
SW253
SW254
SW255
SW256
SW257
SW258
SW259
SW260
SW261
SW262
SW263
SW264
SW265
SW266
SW267
SW268
SW269
SW270
SW271
SW272
SW273
SW274
SW275
SW276
SW277
SW278
SW279
SW280
SW281
SW282
SW283
SW284
SW285
SW286
SW287
SW288
SW289
SW290
SW291
SW292
SW293
SW294
SW295
SW296
SW297
SW298
SW299
SW300
SW301
SW302
SW303
SW304
SW305
SW306
SW307
SW308
SW309
SW310
SW311
SW312
SW313
SW314
SW315
SW316
SW317
SW318
SW319
SW320
SW321
SW322
SW323
SW324
SW325
SW326
SW327
SW328
SW329
SW330
SW331
SW332
SW333
SW334
SW335
SW336
SW337
SW338
SW339
SW340
SW341
SW342
SW343
SW344
SW345
SW346
SW347
SW348
SW349
SW350
SW351
SW352
SW353
SW354
SW355
SW356
SW357
SW358
SW359
SW360
SW361
SW362
SW363
SW364
SW365
SW366
SW367
SW368
SW369
SW370
SW371
SW372
SW373
SW374
SW375
SW376
SW377
SW378
SW379
SW380
SW381
SW382
SW383
SW384
SW385
SW386
SW387
SW388
SW389
SW390
SW391
SW392
SW393
SW394
SW395
SW396
SW397
SW398
SW399
SW400
SW401
SW402
SW403
SW404
SW405
SW406
SW407
SW408
SW409
SW410
SW411
SW412
SW413
SW414
SW415
SW416
SW417
SW418
SW419
SW420
SW421
SW422
SW423
SW424
SW425
SW426
SW427
SW428
SW429
SW430
SW431
SW432
SW433
SW434
SW435
SW436
SW437
SW438
SW439
SW440
SW441
SW442
SW443
SW444
SW445
SW446
SW447
SW448
SW449
SW450
SW451
SW452
SW453
SW454
SW455
SW456
SW457
SW458
SW459
SW460
SW461
SW462
SW463
SW464
SW465
SW466
SW467
SW468
SW469
SW470
SW471
SW472
SW473
SW474
SW475
SW476
SW477
SW478
SW479
SW480
SW481
SW482
SW483
SW484
SW485
SW486
SW487
SW488
SW489
SW490
SW491
SW492
SW493
SW494
SW495
SW496
SW497
SW498
SW499
SW500
SW501
SW502
SW503
SW504
SW505
SW506
SW507
SW508
SW509
SW510
SW511
SW512
SW513
SW514
SW515
SW516
SW517
SW518
SW519
SW520
SW521
SW522
SW523
SW524
SW525
SW526
SW527
SW528
SW529
SW530
SW531
SW532
SW533
SW534
SW535
SW536
SW537
SW538
SW539
SW540
SW541
SW542
SW543
SW544
SW545
SW546
SW547
SW548
SW549
SW550
SW551
SW552
SW553
SW554
SW555
SW556
SW557
SW558
SW559
SW560
SW561
SW562
SW563
SW564
SW565
SW566
SW567
SW568
SW569
SW570
SW571
SW572
SW573
SW574
SW575
SW576
SW577
SW578
SW579
SW580
SW581
SW582
SW583
SW584
SW585
SW586
SW587
SW588
SW589
SW590
SW591
SW592
SW593
SW594
SW595
SW596
SW597
SW598
SW599
SW600
SW601
SW602
SW603
SW604
SW605
SW606
SW607
SW608
SW609
SW610
SW611
SW612
SW613
SW614
SW615
SW616
SW617
SW618
SW619
SW620
SW621
SW622
SW623
SW624
SW625
SW626
SW627
SW628
SW629
SW630
SW631
SW632
SW633
SW634
SW635
SW636
SW637
SW638
SW639
SW640
SW641
SW642
SW643
SW644
SW645
SW646
SW647
SW648
SW649
SW650
SW651
SW652
SW653
SW654
SW655
SW656
SW657
SW658
SW659
SW660
SW661
SW662
SW663
SW664
SW665
SW666
SW667
SW668
SW669
SW670
SW671
SW672
SW673
SW674
SW675
SW676
SW677
SW678
SW679
SW680
SW681
SW682
SW683
SW684
SW685
SW686
SW687
SW688
SW689
SW690
SW691
SW692
SW693
SW694
SW695
SW696
SW697
SW698
SW699
SW700
SW701
SW702
SW703
SW704
SW705
SW706
SW707
SW708
SW709
SW710
SW711
SW712
SW713
SW714
SW715
SW716
SW717
SW718
SW719
SW720
SW721
SW722
SW723
SW724
SW725
SW726
SW727
SW728
SW729
SW730
SW731
SW732
SW733
SW734
SW735
SW736
SW737
SW738
SW739
SW740
SW741
SW742
SW743
SW744
SW745
SW746
SW747
SW748
SW749
SW750
SW751
SW752
SW753
SW754
SW755
SW756
SW757
SW758
SW759
SW760
SW761
SW762
SW763
SW764
SW765
SW766
SW767
SW768
SW769
SW770
SW771
SW772
SW773
SW774
SW775
SW776
SW777
SW778
SW779
SW780
SW781
SW782
SW783
SW784
SW785
SW786
SW787
SW788
SW789
SW790
SW791
SW792
SW793
SW794
SW795
SW796
SW797
SW798
SW799
SW800
SW801
SW802
SW803
SW804
SW805
SW806
SW807
SW808
SW809
SW810
SW811
SW812
SW813
SW814
SW815
SW816
SW817
SW818
SW819
SW820
SW821
SW822
SW823
SW824
SW825
SW826
SW827
SW828
SW829
SW830
SW831
SW832
SW833
SW834
SW835
SW836
SW837
SW838
SW839
SW840
SW841
SW842
SW843
SW844
SW845
SW846
SW847
SW848
SW849
SW850
SW851
SW852
SW853
SW854
SW855
SW856
SW857
SW858
SW859
SW860
SW861
SW862
SW863
SW864
SW865
SW866
SW867
SW868
SW869
SW870
SW871
SW872
SW873
SW874
SW875
SW876
SW877
SW878
SW879
SW880
SW881
SW882
SW883
SW884
SW885
SW886
SW887
SW888
SW889
SW890
SW891
SW892
SW893
SW894
SW895
SW896
SW897
SW898
SW899
SW900
SW901
SW902
SW903
SW904
SW905
SW906
SW907
SW908
SW909
SW910
SW911
SW912
SW913
SW914
SW915
SW916
SW917
SW918
SW919
SW920
SW921
SW922
SW923
SW924
SW925
SW926
SW927
SW928
SW929
SW930
SW931
SW932
SW933
SW934
SW935
SW936
SW937
SW938
SW939
SW940
SW941
SW942
SW943
SW944
SW945
SW946
SW947
SW948
SW949
SW950
SW951
SW952
SW953
SW954
SW955
SW956
SW957
SW958
SW959
SW960
SW961
SW962
SW963
SW964
SW965
SW966
SW967
SW968
SW969
SW970
SW971
SW972
SW973
SW974
SW975
SW976
SW977
SW978
SW979
SW980
SW981
SW982
SW983
SW984
SW985
SW986
SW987
SW988
SW989
SW990
SW991
SW992
SW993
SW994
SW995
SW996
SW997
SW998
SW999
SW1000


T1 – AC1 RTR / ENTER
T2 – A / <--
T3 – B/ -->
T4 – +
T5 – (-)
T6 – ESC/ESC
Title SISTEMA DE CONTROL DE RELOJ CHECADOR FI
Size A1
Number RADM1-FI
Rev 1.SC
Date Enero 22,1995
Drawn by SMC
Filename CHECK2
Sheet 1 of 1



Prototipo utilizado para el desarrollo.



Circuito final.

Capítulo 4

Programación del Sistema y la PC

Una parte del sistema es la programación ó *sofi ware* que, aunque no se ve físicamente, suele ser también importante, ya que constituye la manera en que el sistema se va a comportar.

Para poder llevar a cabo la parte de la programación, se debe contar con la infraestructura necesaria: editores de texto, compiladores, emuladores y grabadores. Para el caso de nuestro desarrollo, en la parte correspondiente a la programación del microcontrolador, se hizo uso de un editor de texto en ASCII, de un compilador para microcontroladores 8X5XX de Intel, de un emulador para el mismo grupo de microprocesadores y finalmente, de un grabador de memorias universal así como de un borrador de memorias ultravioleta (para UVPROM).

Para la programación de la computadora personal, se optó por utilizar lenguaje "C", el cual ofrece muchas ventajas, ya que se pueden utilizar rutinas para manejo de archivos, gráficos y lógica, todas ellas, muy necesarias para el objetivo de la presente tesis.

La programación en "C" se hizo por medio de una versión de Turbo C, la cual ofrece a parte de hacer ejecutable un programa, revisar paso a paso las líneas de éste así como otras funciones que facilitan mucho el buen desarrollo de un programa. Además, es un lenguaje común y cuya bibliografía se puede conseguir en las bibliotecas de la Facultad de Ingeniería.

Programación del Sistema de Identificación

La programación del sistema de identificación se basa en un programa principal apoyado de varias subrutinas. Las subrutinas entran en operación a partir de una interrupción o conforme se vayan necesitando dentro del programa. La programación se puede dividir en tres grandes bloques. El primer bloque es el que se encarga de decodificar el R.F.C. del trabajador, el segundo bloque se encarga de anexar información al R.F.C. de los trabajadores almacenando esta información en la tarjeta IC y el tercer bloque se encarga de la comunicación entre el sistema de identificación y una computadora personal. A continuación se describen cada uno de estos bloques.

Decodificación

La señal que entrega el lector óptico es un tren de pulsos, proporcional en longitud al número de caracteres en el mensaje (se manejaron 11 caracteres para el R.F.C.), de 0 a 5V de amplitud, con pulsos de una duración de 2 a 15mS. En las figuras 4.1.a. y 4.1.b. se muestran las señales observadas en el osciloscopio.

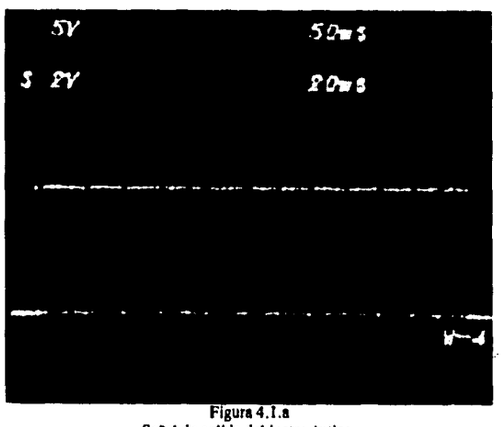
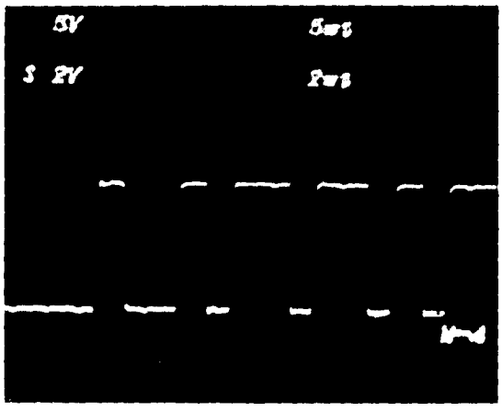


Figura 4.1.a
Señal de salida del lector óptico.



Figuraura 4.1 .b
Señal del lector óptico vista en el osciloscopio.

Para poder realizar la decodificación de este tren de pulsos se realizaron los siguietes pasos:

- Medir y almacenar la duración de cada uno de los pulsos y valles de la señal.
- Discriminar cuáles de estos tiempos pertenecen a un código de barras 39.
- Clasificar los tiempos (Generación automática de rangos).
- Obtener un carácter en código 39.
- A partir del carácter en código 39 obtener un carácter ASCII.

Después de ejecutar estos pasos se obtiene el R.F.C. del trabajador, es importante mencionar que durante estos pasos se verifica la veracidad de la información decodificada.

Se observó de la señal que entrega el lector óptico en el osciloscopio, que la duración de los pulsos es proporcional a la de las barras en la etiqueta leída. Se observó también que en dicha señal figura un pulso con una duración mucho mayor a todas las demás, esto es ocasionado por el espacio en blanco que existe antes del código de barras. Una de las restricciones en la impresión de los códigos es que antes de comenzar el código debe existir un espacio en blanco de por lo menos 10 veces el ancho nominal de una barra delgada del código.

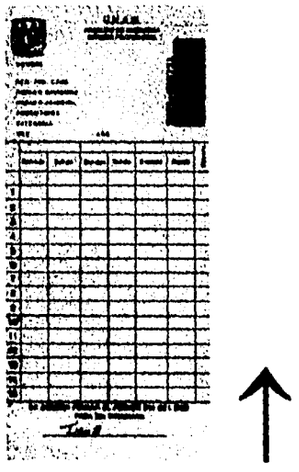
Para efectuar la medición y almacenamiento de los pulsos y valles se programó uno de los temporizadores del microcontrolador. Se escogió entonces un valor de preescala para llevar a cabo las mediciones. La siguiente tabla muestra las opciones para la preescala:

Interrupción por sobre carga Timer 2		
Factor preescala	8 bits	16 bits
1	276.8 µS	71 mS
1/2	553.6 µS	142.2 mS
1/4	1.10 mS	248.5 mS
1/8	2.21 mS	569.1 mS

Tabla 4.1
Tiempos para la generación de interrupción del temporizador 2.

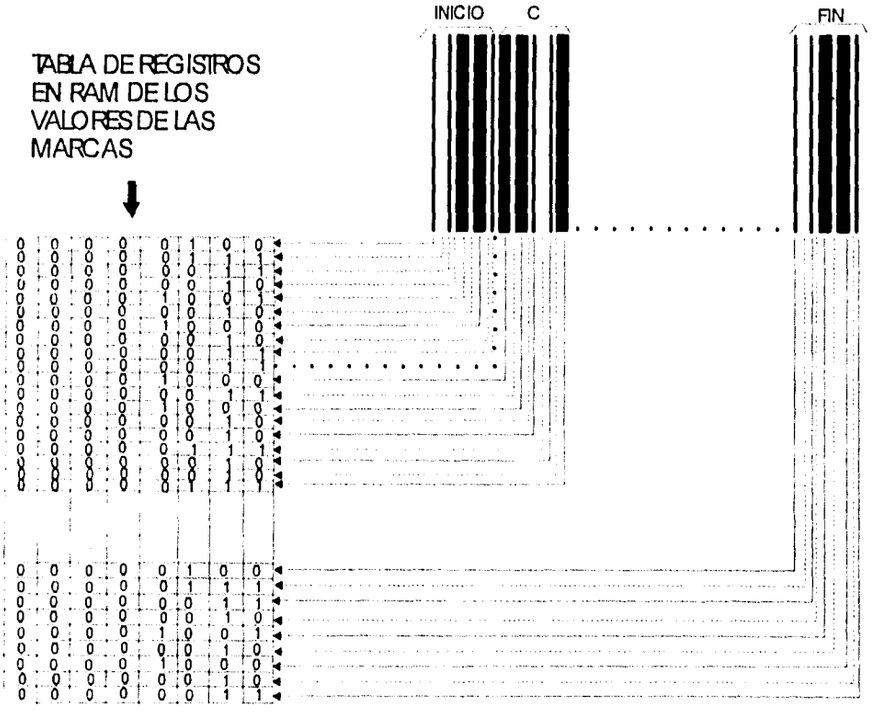
El uso de este temporizador permite conocer el ancho de de los pulsos y de los valles. Dichos valores son almacenados posteriormente en RAM.

Se empezaron a hacer pruebas con la tarjeta de checado que utilizan oficialmente los trabajadores de la Facultad de Ingeniería. Se le colocó a ésta la etiqueta del código como se muestra en la figura 4.2. Al pasar la tarjeta de checado por el lector, éste detecta también las rayas que conforman la cuadrícula donde se sellan la hora y el día de checado. Los valores de estas rayas, que el sistema de identificación detecta como "barras", son registrados de igual modo. Después de tener almacenadas todas la duraciones de los pulsos y valles de la señal es necesario discrimar cuales de éstas no pertenecen al código de barras, por lo que hay que eliminarlas antes de comenzar a decodificar la señal, de lo contrario se pueden generar errores en la decodificación.



Figuraura 4.2
Tarjeta oficial para comprobar la asistencia del trabajador.
El sentido de la flecha indica como va pasando la tarjeta por el lector óptico.

Para discriminar los valores de tiempos se obtuvo la media aritmética de todas las duraciones de los pulsos. Con la media se eliminan todos los datos que sean mayores al valor de la media más un valor de seguridad, hasta tener 130 datos consecutivos menores a dicho valor, con ésto se eliminan todos los valores que se encontraron antes del código de barras y se asegura tener el número de mediciones correspondiente a 13 caracteres (11 del R.F.C. y 2 de los caracteres de inicio y fin). Si existieran mediciones correspondientes a líneas posteriores al código de barras estas no se tomarían en cuenta debido a que sólo se toman 130 lecturas para decodificar una longitud fija en el mensaje. Las lecturas correspondientes al código de barras se almacenan en la RAM. La figura 4.3 muestra cómo se van acomodando los valores conforme se van leyendo las barras ó espacios. Al principio de la etiqueta se encuentra el código de inicio, luego viene el mensaje que para el caso de la figura 4.3 empieza con "C" y al terminar el mensaje viene el código de fin:



Figuraura 4.3
Modo en que se almacenan los valores correspondientes a cada barra y espacio.

Selección Automática de Rangos

Una vez que la duración de cada pulso de la señal (únicamente el correspondiente a las barras del R.F.C.) fue almacenada dentro de los registros de la RAM, se genera la interrupción por sobrecarga del temporizador con el objeto de hacer que el microprocesador lleve a cabo la decoficación de las señal sin que ocurra otra acción.

En el capítulo 1 se describieron las características del código 39 y se dijo que cada carácter o símbolo está compuesto por barras gruesas y barras delgadas, así mismo, que contiene espacios gruesos y espacios delgados. Otra característica importante es que consta de caracteres de inicio y fin de código.

Se les llamará **marcas** a los espacios y a las barras, gruesos o delgados, de manera indistinta. Al caracterizar la señal del lector óptico y que ésta hubiera sido una lectura correcta, se observó que las marcas tenían una duración dentro de un rango, por esta razón se trato de fijar un rango de valores para clasificar las marcas. Conforme se avanzó en el trabajo y con las diferentes muestras que se tomaron, resultó que las mediciones variaban según la forma en que se deslizaba la tarjeta por el lector, por lo que no era práctico fijar un rango de tiempos sino que éste fuera variable.

Se propuso entonces establecer cuatro rangos: para barras delgadas, barras gruesas, espacios delgados y espacios gruesos; basados en las mismas mediciones del R.F.C. leído. Esto se logró tomando las medidas de los códigos de inicio y fin de cada R.F.C., pues son las únicas marcas de las que se sabe su equivalencia y representan además, los valores máximos y mínimos de cada marca (esto se observó al tomar varias lecturas), debido a la aceleración de la tarjeta, de esta forma se obtienen los rangos para las marcas. La figura 4.4 muestra lo anterior.

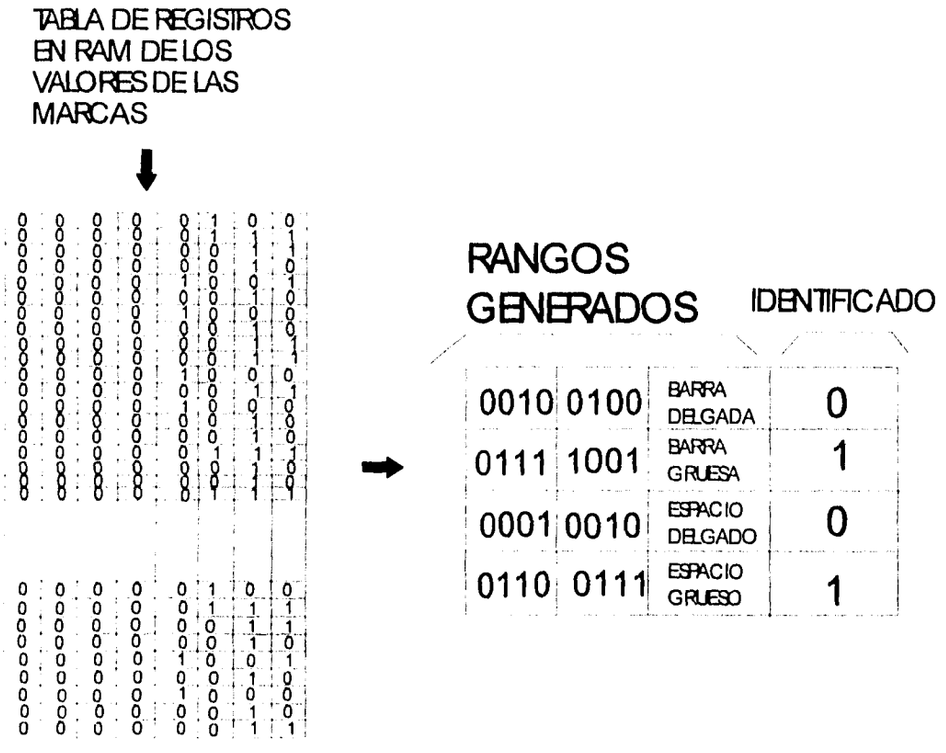


Figura 4.4
Generación de rangos a partir de la lectura de un código de barras.

Posteriormente se examinarán estos rangos para tratar de ampliarlos si es posible. En el caso de que estos rangos se traslapen la decodificación no se lleva acabo.

Así con la subrutina de decodificación empieza el análisis de los valores almacenados y la clasificación según el rango al que dicho valor pertenezca, sabiendo que se tiene necesariamente la secuencia: barra-espacio-barra-...-barra.

Una vez que se determinan los rangos se procede a identificar cada uno de los valores y se les asigna, un **1** si son gruesos o un **0** si son delgados, ya sean barras o espacios como se muestra en la figura 4.5, cabe hacer notar que hasta este momento el espacio entre caracteres se

maneja como un espacio delgado. Para guardar estos valores, en la figura 4.5 se ocupó otro grupo de registros en RAM y se les llamó registros de clasificación. Cada cero ó uno se va colocando como el bit menos significativo de un registro de clasificación, rotando este registro para almacenar el siguiente pulso de un carácter, aquí es donde se separan los espacios entre caracteres de la señal.

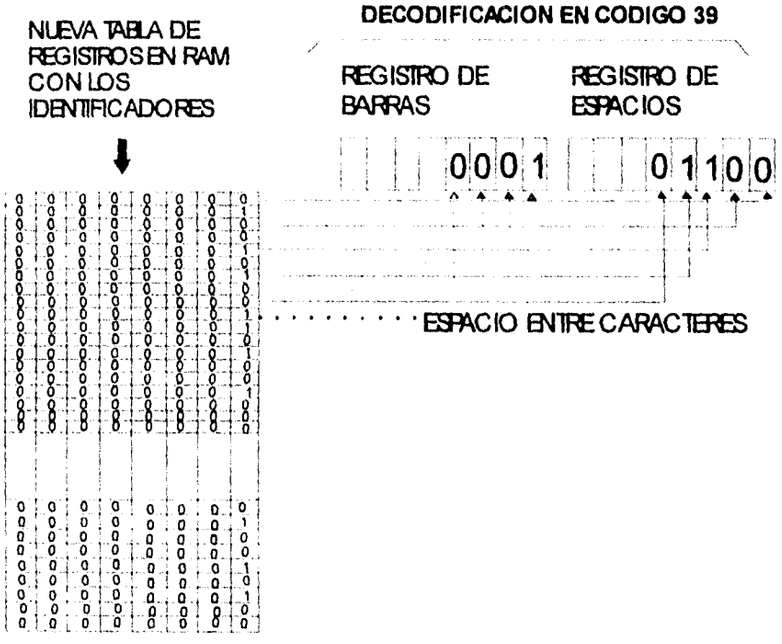


Figura 4.5
Clasificación de barras y espacios.

Por cada carácter del R.F.C. el microcontrolador maneja dos registros denominados registros de código 39. El primer registro se utiliza como la parte baja del carácter, almacena la clasificación de ceros ó unos de los espacios. El segundo registro se utiliza como la parte alta y contiene la clasificación de ceros ó unos de las barras. Los registros de código 39 se van llenando con los valores contenidos en los registros de clasificación, siguiendo la secuencia barra-espacio-barra-...-barra. Según el código 39, hay 9 marcas por cada cárácter, cinco son barras y cuatro son

espacios. El carácter debe contener 3 unos y 6 ceros, con esto se pueden detectar algunos errores de decodificación mediante una pequeña subrutina que cuenta el número de unos por carácter y en caso de no ser tres, manda un mensaje de error y anula la operación, lo que hace necesario que se vuelva a pasar la tarjeta.

Para desplegar el carácter en el *display*, la subrutina se vale del valor binario que se forma en los registros de código 39. Este valor lo utiliza como el número de la dirección en el que los valores ASCII de cada carácter se encuentran guardados dentro de la memoria ROM (figura 4.6), almacenando en RAM los caracteres ASCII del R.F.C..

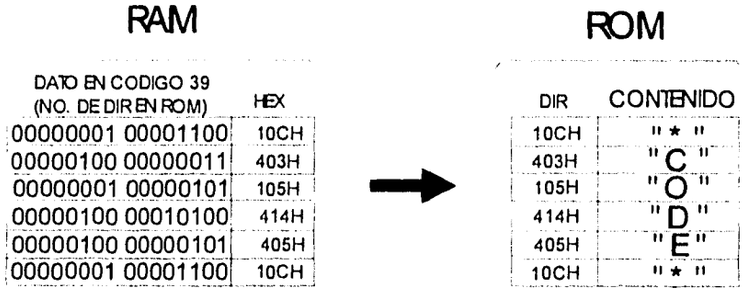


Figura 4.6
Equivalencia del código 39 al código ASCII.

Almacenamiento de los Datos en la Tarjeta IC

Una vez que el dato ha sido almacenado en RAM, se procede a pasar los datos a la tarjeta IC. La información que se almacena en la tarjeta es el R.F.C. del trabajador que se leyó en el código de barras y junto con éste, la hora, minutos y segundos, el día de la semana, el día del mes, y el año, que se toman de los registros del reloj de tiempo real (RTR)

Ya se mencionó que el lector óptico debe ir junto con el reloj checador mecánico, por lo que se ha instalado un adaptador entre el reloj mecánico y el lector óptico de manera que el trabajador deba pasar la tarjeta primero por el lector hacia el reloj mecánico. El diseño del adaptador permite que la tarjeta sea depositada en la parte superior al sistema y por medio de dos paredes paralelas, sea guiada dentro del reloj mecánico. El trabajador deberá accionar el pulsador de la misma forma como lo hacía antes y así sellar su tarjeta.

Al oprimir el pulsador, un interruptor colocado en el interior del reloj mecánico se activa y manda una señal al sistema, lo que permite a este último continuar con la rutina que graba la

información en la tarjeta IC. El sistema espera la señal del reloj mecánico para asegurar que la hora que se sella el reloj mecánico a la tarjeta es la misma que tiene el reloj de tiempo real en el sistema. El administrador debe sincronizar ambos relojes antes de poner en operación al sistema.

El manejo de la tarjeta es igual que para una RAM. La diferencia está en que hay que moverse primero a la página donde ésta trabaja. Los apuntadores del microcontrolador se cargan con el número de la direeción en donde empiezan los caracteres ASCII del R.F.C. leído. Después, se cargan otros apuntadores al inicio de las direeciones de la tarjeta IC.

Los datos contenidos en la memoria RAM son únicamente los correspondientes al R.F.C. del trabajador. El sistema, como ya se mencionó toma el dato de la direeción de la RAM y antes de llevarlo a la correspondiente dirección de la tarjeta IC, pregunta si la tarjeta ha llegado ya a su límite de capacidad. En caso de no haberlo hecho, graba entonces el dato en la tarjeta, y los contadores de la misma se quedan con un valor le indicará al microcontrolador en qué dirección de la tarjeta IC debe empezar a grabar cuando el siguiente R.F.C. sea leído.

Previamente, la subrutina graba los números de las direeciones en las que se quedó la tarjeta IC. en otra localidad de ésta, con el fin de hacer que cada que se encienda el sistema, los apuntadores no se inicialicen en cero, y de esta manera, se evita que el siguiente R.F.C. sea escrito encima del anterior antes de que haya sido vaciado en la lista de nómina.

Si los apuntadores de la tarjeta IC han llegado al número previamente definido como el límite de capacidad, entonces da inicio otra rutina en donde se pide al administrador revise la información contenida en la tarjeta y posteriormente la borre. A partir de este momento el sistema deja de adquirir información y espera a que la tarjeta sea borrada, será hasta entonces cuando el sistema reanude su operación cotidiana.

El administrador debe entonces vaciar la información contenida en la tarjeta a un archivo en la PC. Después de ello, debe borrar la tarjeta y regresarla al sistema. Más adelante se hará una descripción de la parte del programa en la computadora personal que recibe la información y genera dichos archivos. A continuación se describen las formas en que se puede borrar la tarjeta:

Existen dos opciones para borrar la tarjeta IC. La primera de ellas es sacar la tarjeta del sistema y llevarla al drive especial de la IC conectado a una PC. El programa que se hizo para la PC utiliza los comandos específicos que manejan al drive y borran la tarjeta IC. Estos comandos lo que hacen es llenar los registros de la tarjeta con **unos** lógicos, que equivalen al valor de **FF** hexadecimal, por registro.

La segunda opción para borrar la tarjeta es también por medio de un programa en la PC que se vale de la comunicación por el puerto serie. El programa de la PC interrumpe al microcontrolador y borra la tarjeta sin tener que sacarla del sistema. Para este efecto la

computadora personal manda un comando por el puerto serie al microcontrolador. Este está programado para generar una interrupción al momendo de recibir dicho comando. La interrupción en el microcontrolador conduce a una subrutina cuyo fin es el de llenar con el valor hexadecimal **FF** todos los registros de la tarjeta IC.

Una vez que la tarjeta ha sido borrada, el microcontrolador ejecuta otra subrutina que se encarga de verificar si efectivamente la tarjeta fue borrada y continuará con su operación.

Cabe mencionar que existen tres subrutinas que se utilizan para detectar si la tarjeta IC está presente, si no está puesta su proteeción contra escritura y si la batería mantiene el voltaje óptimo. Estas subrutinas entran en operación antes de que la tarjeta IC sea escrita, hasta que las tres sean ciertas el programa podrá avanzar.

El límite de la capacidad de la tarjeta está calculado para 2.5 meses aproximadamente (capítulo 2). La Facultad de Ingeniería genera semanalmente (en algunas ocaciones lo hace cada quince días) el reporte de asistencias, por lo que no será necesario que la tarjeta llegue a su límite de capacidad. Sin embargo, en caso de ocurrir, el sistema no permitirá realizar ninguna operación salvo la de vaciar y borrar la tarjeta.

Comunicación entre el Sistema de Identificación con una PC

La comunicación entre el sistema de identificación y una computadora personal se utiliza para ealizar las siguientes funciones:

- Actualizar la fecha y hora en el reloj de tiempo real.
- Adquirir la información contenida en el sistema de identificación sin necesidad de ir al lugar donde éste se encuentra.
- Cambiar la velocidad de transmisión.

Para realizar esta comunicación se programó el puerto serial del microcontrolador de forma que se puede optar por una velocidad de transmisión de 1200, 2400, 4800 y 9600 bauds (modo 3 de funcionamiento), esto es con el fin de que el usuario escoja la velocidad según la distancia a la que se encuentra la computadora del sistema de identificación, la velocidad de transmisión base que maneja el sistema de identificación es de 1200 bauds. El microcontrolador transmite 10 bits, uno de inicio, ocho de información y uno de paro.

Cuando se quiere entablar una comunicación entre el sistema de identificación y una PC, la PC verifica si el sistema de identificación está conectado al puerto serial, esto lo hace monitoreando la velocidad de transmisión a la cual está operando el sistema de identificación y se ajusta a ésta. Si la PC no puede acceasar al sistema inhabilita toda comunicación.

Para que estos dispositivos se puedan entender, la comunicación entre ellos tiene un formato especifico. La figura 4.7 muestra como llegan los datos del microcontrolador a la computadora personal (PC) y viceversa. Cuando se realiza una comunicación entre una PC y el sistema sin importar la dirección de ésta, se manda un carácter de inicio, un carácter de control, datos (si es que los hay), y un carácter de fin de transmisión "$".

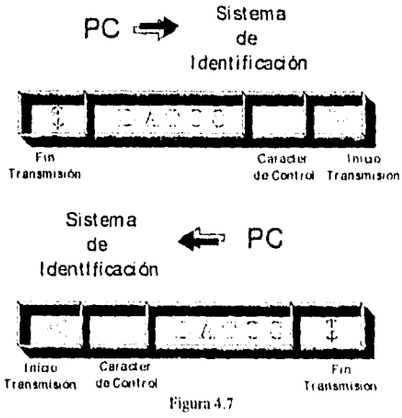


Figura 4.7

Tren de datos de la comunicación serial entre la Computadora Personal y el Microcontrolador.

El sistema de identificación al recibir un carácter de inicio de transmisión el µC comienza a almacenar toda la información que recibe, para después examinar el carácter de control, dependiendo del carácter de control realiza una acción o toma los siguientes caracteres para configurar el µC como se muestra en la tabla 4.2. Si el carácter de control recibido no es válido el µC ignora todos los caracteres recibidos.

Comunicación PC → Sistema de Identificación	
Carácter de Control	Acción
@	Actualiza el Reloj de Tiempo Real.
&	Accesa la infromación almacenada en la tarjeta IC-RAM.
#	Cambia la velocidad de transmisión.
V	Checa la velocidad de transmisión al iniciar la comunicación.
!	Borra el contenido de la tarjeta IC-RAM
L	Avisa el cambio de la velocidad de trasmisión de la PC para confrimar el del sistema.

Tabla 4.2

Caracteres de control que se fijaron para la comunicación serial.

La figura 4.8 muestra la pantalla que aparece en el programa para la computadora personal en donde se ingresan los datos de hora y fecha para actualizar el reloj de tiempo real del sistema del microcontrolador. Cuando la PC actualiza la información del reloj de tiempo real, además de mandar el carácter de control manda la información de la hora y fecha en código ASCII. Posteriormente el programa del microcontrolador toma los valores recibidos, convirtiéndolos a su correspondiente número en código BCD y ejecuta un programa de actualización del reloj de tiempo real, este programa está compuesto de dos subrutinas: la subrutina de actualización de hora y la subrutina de actualización de fecha. Esta forma de actualización se escogió porque si el usuario quiere sincronizar el reloj mecánico con el sistema de identificación generalmente sólo actualizará la hora de éste.

Figura 4.8
Pantalla para actualizar fecha y hora en el reloj de tiempo real del sistema.

La pantalla mostrada en la figura 4.9 corresponde a la opción en la PC para cambiar la velocidad de transmisión (*baud rate*). Al cambiar la velocidad de transmisión, la computadora, además de mandar el carácter de control, manda la velocidad de transmisión a la cual se quiere trabajar. Al recibir la velocidad de transmisión el sistema de identificación responde con la velocidad de transmisión mandada por la PC para confirmarla, después la PC envía un carácter de control (L) el cual avisa al microcontrolador que cambie su velocidad de transmisión, al recibir el sistema este carácter cambia su velocidad de transmisión, un tiempo después manda un carácter con la nueva velocidad de transmisión y con este carácter la PC sabe que el cambio de velocidad de transmisión fue exitoso. Si el cambio de velocidad no fue exitoso el µC permanece con la velocidad de transmisión anterior al cambio que se intento antes.

Figura 4.9
Pantalla para cambiar la velocidad de transmisión tanto en el microcontrolador como en la PC.

Manejo de Archivos en la Computadora Personal (PC)

En la sección que describe a la rutina que pasa los datos de la RAM a la tarjeta IC se habla de cómo borrar la tarjeta. Las opciones que se presentan son a la vez, parte de la opción general para manejar los datos de la tarjeta auxiliándose de un programa en PC.

La siguiente pantalla (figura 4.10) muestra el menú para poder manejar la información de la tarjeta IC:

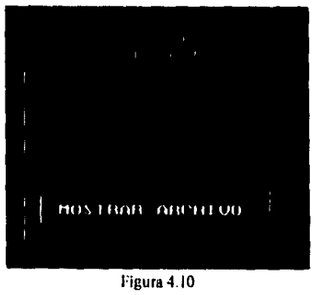
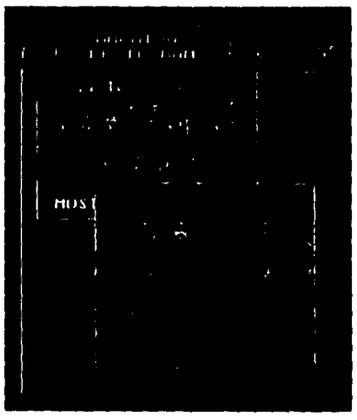


Figura 4.10
Pantalla del menú de opciones para manejar la información de la Tarjeta IC-RAM.

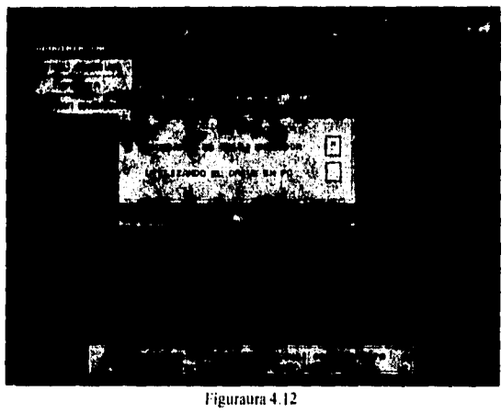
ARCHIVOS DE LA IC RAM. Es un menú exclusivo para manejar la información de la tarjeta IC y clasificarla en distintos archivos. Estos archivos se refieren a los reportes de asistencia, faltas, puntualidad y tiempos extras (Figura 4.11).

Cada opción es única, esceptuando la última **MOSTRAR ARCHIVO**, que se subdivide en otro menú, el cual se muestra también en la figura 4.11 y cuya función es la de desplegar en pantalla los reportes generados. A continuación se describirán las opciones de la figura.



Figuraura 4.11
Menús de visualización de información.

Para poder generar cualquier reporte es necesario antes adquirir la información contenida en la tarjeta IC. Cuando la PC accesa la información del sistema de identificación pide al usuario, como primera instancia, el nombre del archivo donde se guardará la información del sistema, siendo este nombre la referencia para los reportes que se generarán, modificandose sólo su extensión : "**nombre.extensión**". Posteriormente pregunta el medio por el cual se accesará a la infromación: puerto serie ó drive en PC como se muestra en la figura 4.12. Cuando el medio de acceso es el puerto serie, la PC manda el carácter de control para accesar la información en el sistema de identificación, éste responde con un carácter para que la PC sepa que el sistema comenzará a mandar la información, un tiempo después el sistema manda la información en bloques de 1 Kbyte (28 caracteres para el R.F.C. con información de hora y fecha de entrada), si exite más información, el sistema cambia el carácter de fin de transmisión por el carácter "|", con esto la PC sabe que el sistema mandará otro bloque de información, este proceso continúa hasta terminar de mandar la información.



Figuraura 4.12
Pantalla para escoger el medio por el cual se adquirirá la información.

Una vez que la información ha sido enviada a la PC, ésta última manda un carácter que le indica al sistema de identificación que debe borrar la información contenida en la IC-RAM (si es que el usuario así lo decide).

En la figura 4.13 se muestra la pantalla en la que el programa le pide al usuario dar un nombre de archivo, cuya extensión sea "**.ord**", éste puede ser el mismo nombre que se acaba de generar al adquirir la información ó uno que tenga la misma extensión y que ya haya sido definido anteriormente por el mismo programa. Este archivo contiene una lista preliminar ordenada alfabéticamente, la extensión permite identificar el tipo de archivo.

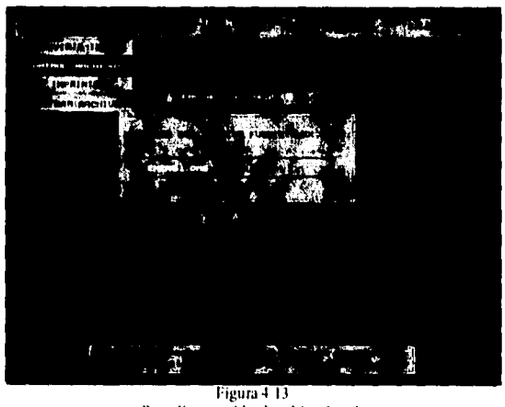


Figura 4.13
Pantalla que pide el archivo *.ord.

Después de pedir el nombre del archivo *.ord, en el mismo submenú se pide el rango de fechas que se desea analizar dentro de éste archivo así como el total de horas laboradas. La figura 4.14 muestra cómo es la pantalla.

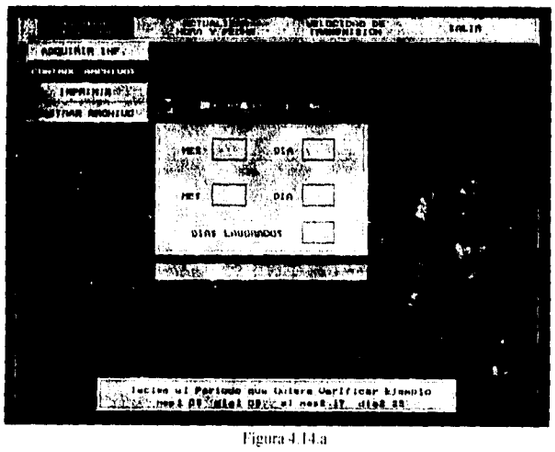


Figura 4.14.a
Pantalla para definir el periodo de tiempo a examinar

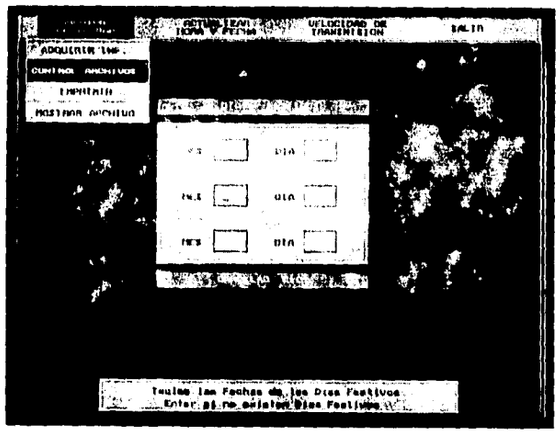


Figura 4.14.b
Pantallas donde se ingresan los días festivos

El programa está hecho de forma que el administrador obtenga los reportes en períodos desde un día hasta un mes. La administración de la Facultad de Ingeniería reliza sus reportes, por lo general, cada semana.

Lo anterior puede obtener la Lista Completa que se muestra en la pantalla de la figura 4.15.

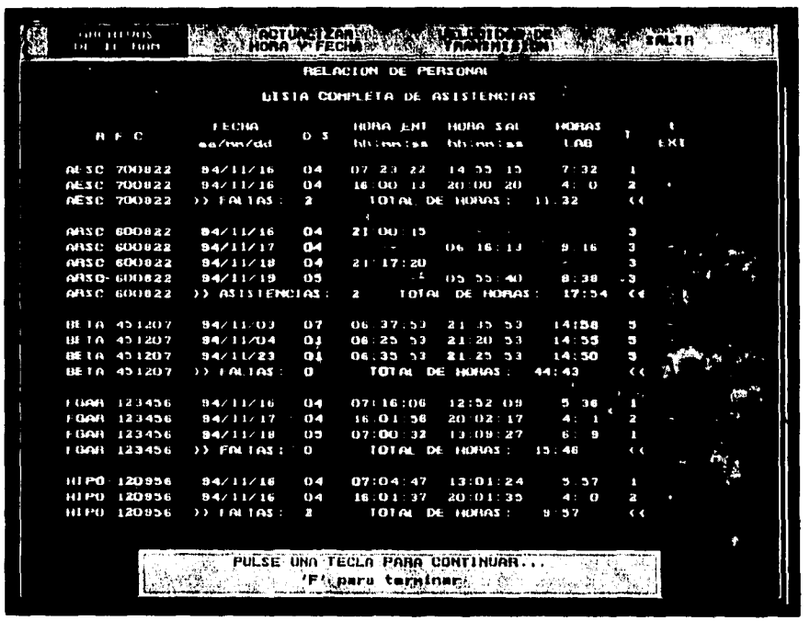


Figura 4.15
Pantalla que despliega el reporte de la lista completa

Además del período a analizar, se piden también las fechas de los días festivos que tuvieran lugar dentro del rango dado. Las listas de puntualidad, faltas y tiempos extras, se generan a partir de de la lista completa.

Para poder elaborar la lista de puntualidad, es necesario que el administrador proporcione los horarios que desea analizar, especificando la hora de entrada y la de salida como se muestra en la pantalla de la figura 4.16.

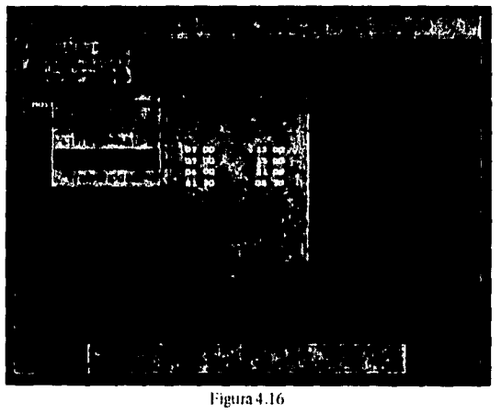


Figura 4.16
Pantalla para ingresar horarios.

El programa con esta referencia, busca todos los R.F.C. en *.ord que estén lo más cerca posible (con tolerancias de hasta 15 minutos antes de la hora de entrada y después de la hora de salida) de dichos horarios. Además analiza con ello los minutos de retraso del trabajador en la hora de entrada y de adelanto en la hora de salida y para generar posteriormetne un archivo que se despliega en una pantalla como la siguiente (Figura 4.17):

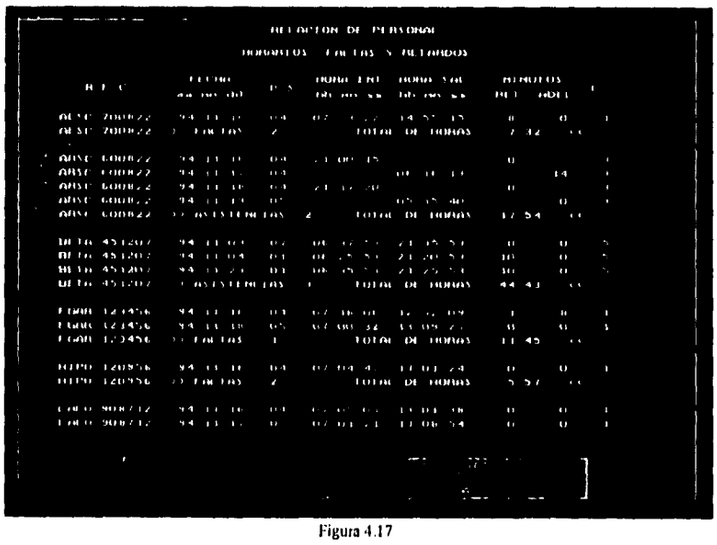


Figura 4.17
Pantalla del reporte de puntualidad.

La siguiente tabla es un resumen de lo que contiene cada uno de los reportes generados con el programa de la computadora personal. El nombre de todos será el mismo, sólo cambiará su extensión.

TIPO DE REPORTE	CONTENIDO	EXTENSION
LISTA COMPLETA	R.F.C., fecha, día de la semana, entrada, salidas horas laboradas por jornada, turno y posibles tiempos extras. Total de faltas y total de horas laboradas en el período.	*.CMP
PUNTUALIDAD	R.F.C., fecha, día de la semana,, entrada, salida, minutos de retardo y de aldelanto, turno, total de faltas, total de horas laboradas por período.	*.HOR
FALTAS	R.F.C., total de faltas, total de horas laboradas, nota: para el turno 3 se registran sólo asistencias	*.FAL
TIEMPOS EXTRAS	R.F.C., fecha, día de la semana, horas de entrada y salida, total de horas laboradas en la jornada, turno en que se realizó el tiempo extra	*.TEX

Tabla 4.3
Relación de nombres y extensiones de los reportes del programa en PC

Las horas laboradas se despliegan en cada lista. El programa hace un recuento de las horas que el trabajador laboró desde su entrada hasta su salida (jornada) y posteriormente, el recuento de las horas que laboró durante el período establecido por el administrador.

La lista de faltas se elabora en base al total de días laborados especificados por el administrador (figura 4.18). La lista proporciona el número de faltas para los turnos 1 y 2. En el caso de los turnos 3 y 5, sólo proporciona asistencias debido a que sus registros no son regulares con respecto a los dos primeros.

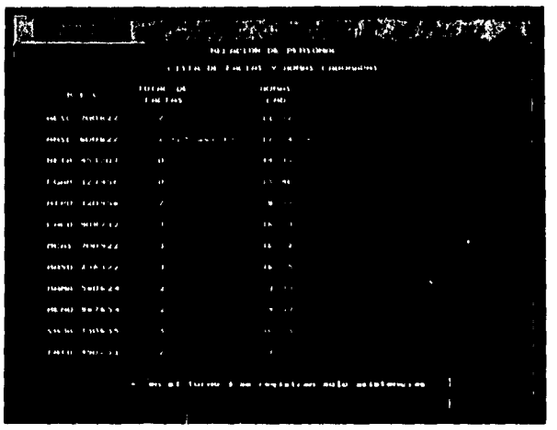


Figura 4.18
Lista de Faltas.

La administración de la Facultad de Ingeniería establece que el primer turno es de las 6:00 a las 13:30 horas, el segundo turno se especifica de las 13:30 horas y hasta las 21:30 horas. El turno 3 es el nocturno y tiene el horario de las 21:30 horas hasta las 6:00 horas del día siguiente. Existe un turno 5 ó especial que sólo trabaja los sábados, domingos y los días festivos, este turno tiene el horario de 6:00 a 21:30 horas. Los horarios aquí descritos no son muy específicos, es decir, pueden variar y no necesariamente todos los trabajadores entran y salen a las horas antes mencionadas, sino que puede suceder que sus horarios de entrada y salida estén dentro del rango correspondiente a dichos turnos, especialmente los que se encuentran en los turnos 1 y 2. El programa sólo analiza los horarios de entrada y salida en los que checó el trabajador y con ello, deduce el número de turno.

La razón por la que se piden el total de días laborados y las fechas de los días festivos es porque este programa no tiene acceso a la base de datos de la administración y se necesita un punto de referencia para que el programa proporcione los reportes deseados. Por la misma razón, se piden los horarios específicos de entrada-salida que se desean analizar dentro de la lista *.ord. Los tiempos extras se basan en los trabajadores que tienen registradas dos jornadas entrada-salida al día. La figura 4.19 muestra cómo se despliega esta pantalla. El programa no tiene forma de saber si el doble registro se refiere a un trabajador de plaza y media (que debe checar dos veces entrada-salida al día) ó efectivamente a un tiempo extra. La lista completa marca cuáles serían los trabajadores con posibles tiempos extras.

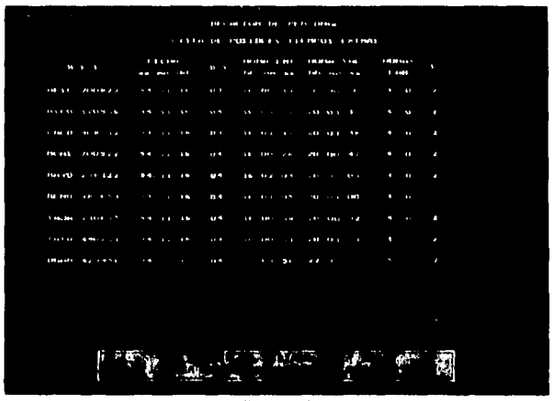


Figura 4.19
Pantalla de la lista de Tiempos-Extras

Una vez que se elaboraron los reportes, se pueden ver cada uno de ellos en la pantalla así como imprimirlos, por ello se les ha distinguido a cada uno con diferentes extensiones, lo que permitirán al administrador saber a qué se refiere cada archivo. El poder imprimir los diferentes reportes generados por el programa permite al administrador hacer uso de los listados e ingresarlos a su sistema de nómina. Si hubiera alguna irregularidad con los registros de checado de algún trabajador y éste no apareciera en alguno de los listados, entonces el submenú ofrece la búsqueda en la lista del archivo *.ord, el R.F.C. que se desee y desplegar en pantalla toda la información que se tenga acerca del mismo. Lo anterior se muestra en la figura 4.20.

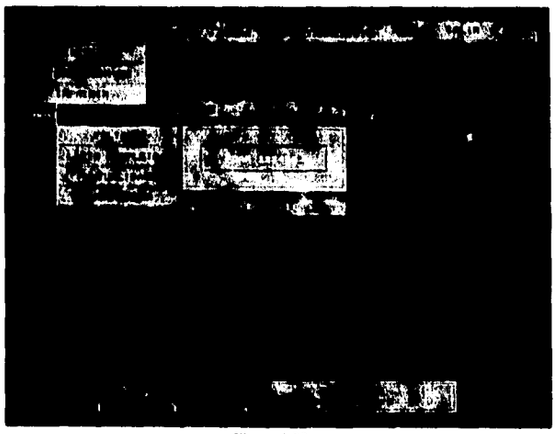


Figura 4.20 a
Pantalla que pide el RFC a buscar

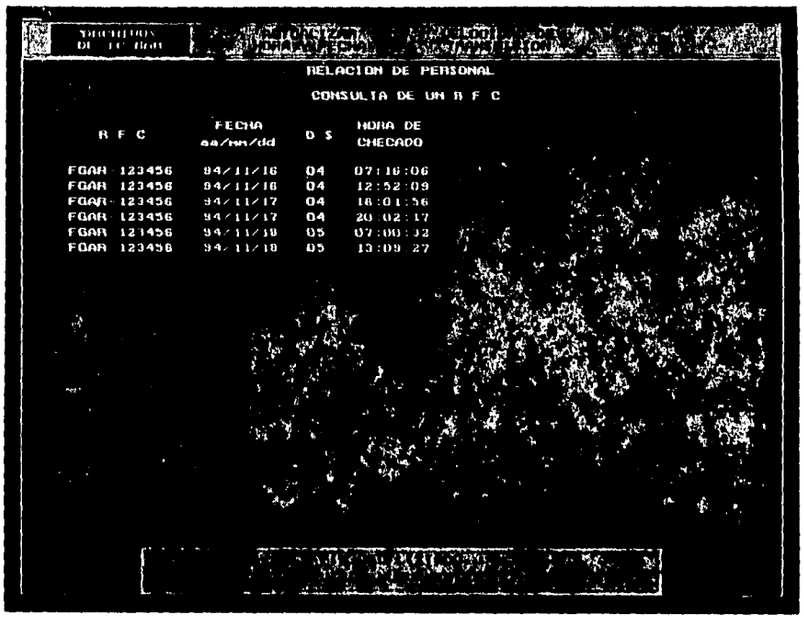


Figura 4.20
Pantalla para la Búsqueda por RFC.

Cabe mencionar que mientras no se tenga un archivo *.ord, el programa no podrá realizar ningún reporte. Se hace hincapié en que el formato de los archivos que elabora el programa de la PC es un formato particular y es sólo éste el formato que puede leer el programa tanto para generar los reportes como para desplegarlos en pantalla. Los archivos son grabados en formato ASCII ó Texto, por lo que se pueden leer también en los procesadores de texto compatibles con el editor del sistema operativo de la PC.

De esta forma se concluye con la programación del sistema de identificación y con el programa de procesamiento de la información de la computadora personal, con este capítulo y la información de los capítulos 1, 2, 3 se puede tener una conceptualización del sistema en conjunto.

Resultados y Conclusiones

La investigación previa al desarrollo de este proyecto fue una parte muy importante ya que se determinó con ella su factibilidad. La existencia de equipos comerciales puede dar lugar a pensar que no es necesario diseñar este tipo de dispositivos puesto que el comprarlos podría ahorrar tiempo. Sin embargo, la investigación dió como resultado que no existen equipos que empleen tarjetas de memoria (IC RAM) como la que se ocupó para este sistema y cuya utilidad representa ventajas en la forma de almacenar y transferir la información (almacenar una gran cantidad de información y permitir transportarla en una tarjeta para leerla en una PC sin necesidad de transmitirla, ofreciendo una manera alterna de accesar a la información). Por otro lado, las necesidades de la Facultad de Ingeniería son muy específicas como para encontrar un equipo comercial que reúna a todas ellas, los equipos que cubren estas necesidades debían realizar cambios en la forma de llevar el control administrativo (cambio de tarjetas de checado, programas de procesamiento de infromación y base de datos), además de ser costosos (haciendo un estudio de los equipos comerciales, todos éstos son extranjeros).

El desarrollo de un algoritmo de decodificación del código de barras con longitud fija, el cual atenúa las variaciones de velocidad al pasar la tarjeta por el lector óptico permite que el sistema de identificación desarrollado presente una razón de primeras lecturas de más del 90%, lo que resulta bastante aceptable. Presentando una razón de error de sustitución baja. Por lo anterior creemos que se cumplen con las espectativas que se marcaron al inicio de este desarrollo. Este algoritmo puede ser modificado para decodificar un código de longitud variable.

El haber desarrollado un programa de decodificación de código de barras, permite utilizarlo en otras aplicaciones como en el control de acceso; ajustando los programas a las necesidades que se tengan, esto justifica en gran medida el desarrollo de estos equipos ya que se genera tecnología propia, que se puede utilizar o modificar, aunque existan ya equipos comerciales

El sistema que se desarrolló en este proyecto permite a la Facultad de Ingeniería contar con un sistema adecuado a sus necesidades y además elaborado con tecnología mexicana diseñada por alumnos de la misma escuela, lo cual disminuye su costo y motiva a seguir creando.

Las restricciones impuestas por la Facultad de Ingeniería no permiten que el sistema sea totalmente automático. Como se mencionó antes, la tarjeta del trabajador es un comprobante de asistencia legal y debe ser sellado al momento de registrar la entrada y la salida del trabajador y al no tener un medio de impresión para el formato de la tarjeta, fue necesario conservar el reloj mecánico que ya se tenía. En este proyecto se soluciona el problema adaptando el lector óptico al reloj checador mecánico de forma que, una vez leído el código de barras, se registra la hora en la tarjeta de checado y en el sistema de identificación (a este último se anexa el R.F.C.), simultáneamente . Lo anterior se logra con una señal eléctrica (SRM) que se origina al accionar el pulsador del reloj mecánico para imprimir la hora y fecha y que va directamente al sistema de identificación. Otra restricción fue hacer que el programa que genera los reportes fuera independiente de la base de datos de la nómina. El problema se resolvió realizando comparaciones entre los datos de los registros y los datos que se le piden al administrador.

Por lo anterior, es aceptable proponer modificaciones al sistema que permitan más versatilidad y crear un módulo totalmente automático. Se han considerado dos modificaciones muy importantes que podrían llevarse a cabo posteriormente, la primera de ellas es crear un sólo módulo independiente que contenga al sistema de identificación (lector óptico y decodificador), la base para la tarjeta de memoria IC y una impresora que permita sellar las tarjetas de los trabajadores. La segunda modificación consistiría en hacer que el sistema tenga acceso a una base de datos de manera que se automatice aún más el manejo de la información.

Una tercera modificación sería el conectar una red de sistemas de identificación que concentren la información en una sola tarjeta IC. Estos sistemas de identificación, pueden tener la capacidad de comunicarse entre sí, lo que permitiría a los trabajadores checar en cualquier

reloj. Posteriormente se obtendría la información de la tarjeta IC utilizando una computadora personal.

A continuación se muestra una fotografía del sistema final.

Sistema automático de registro de asistencias.

Bibliografía

Brian N. Kernighan., Dennis M. Ritchie. *The C Programming Language.* N. Jersey E.U.: Prentice Hall, 1987.

Bruce Ekkel. *Using C++.* California: Osborne/Mc Graw-Hill,1987.

Data Logic. *Automatic Identification, Bar Code Identification, Product Catalog.* 1993-94.

David Kruglisnki. *Guía de las Comunicaciones del IBM/PC.* México: Mc Graw-Hill, 1985.

D. E. Pippenger., E. J. Tobaben. *Linear & Interface Circuits Applications.* Mc Graw-Hill, 1988.

Harry E. Burke. *Automating Managment Information Systems Vol. 2 Bar Code Engineering and implementation.* New York: Van Nostrand Reinold.

Herbert Shield. *Turbo C Manual de Bolsillo.* México: Osborne/Mc Graw-Hill, 1989.

Herbert Shield. *Turbo C Second Edition.* California, E.U.: Borland Osborne/Mc Graw-Hill, 1989.

Intell. *Embedded Aplications.*

Intell. *Embedded Microcontrollers and Processors Volume II.* Illinois, EUA. 1992

James L. Pinson. *Designing Screen Interfaces in C.* Englewood Clifs: Prentice Hall, 1991.

M. Morris Mano. *Diseño Digital.* México: Prentice Hall, 1987.

Markus A. Levy. *Designing with Flash Memory.* Review .December '90-January '91, 50-58.

Motorola Data Book. *Fast & LS TTL Data.* 1992.

National Semiconductor Corporation. *CMOS Logic Data Book.* California, USA. 1988.

National Semiconductor Corporation. *Linear Data Book.* California, USA. 1988.

National Semiconductor Corporation. *Memory Data Book.* California, USA. 1990.

National Semiconductor Corporation. *Transistor Data Book, Field Effect/Power/Small Signal.* California, USA. 1992.

Paul Harowitz. Winfield Hill. The Art of Electronics. E. U. A.: Cambridge Press, 1982

Robert Bojlestad. Louis Nashelsky. *Electrónica, Teroía de Circuitos.* México: Prentice Hall, 1989.

Rodrigo Cordoba. *El Código de Barras, ¿Hasta Donde LLega su Control?.* Computer/ World/México. Marzo 1987, 28, 34.

Rodrigo Cordoba. *Lectores de Código de Barras para la Industria.* Computer/ World/México. Abril 13, 1987, 8, 39.

Rodrigo Cordoba. *Terminologia Basica*. Computer/ World/México. Abril 29, 1987, 6-8.

Rodrigo Cordoba. *Código de Barras , ¿Donde se esta Utilizando?*. Computer/ World/México. Mayo, 1987, 8, 9.

Roger C. Palmer. *The Bar Code Book Reading, Printing and Specification of Bar Code Systems*. New Hampsire, USA:

Thremblay Sorervas. *An Introduction to Data Structures with Aplications*. Singapore: Mc Graw-Hill, 1985.

Apéndice A

Hojas de Especificaciones

MICROCONTROLADOR 80C552

TC-RAM RBJ-1000

RELOJ DE TIEMPO REAL MC146818

TESIS SIN PAGINACION

COMPLETA LA INFORMACION

Single-chip 8-bit microcontroller with 10-bit A/D, capture/compare timer, high-speed outputs, PWM

DESCRIPTION

The 80C552/83C552/87C552 (hereafter generically referred to as 8XC552) Single-Chip 8-Bit Microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The 8XC552 has the same instruction set as the 80C51. Three versions of the derivative exist:

- 83C562 — 8k bytes mask programmable ROM
- 80C562 — ROMless version of the 83C562
- 87C552 — 8k bytes EPROM

The 8XC552 contains a non-volatile 8k × 8 read-only program memory (83C552 EPROM (87C562)), a volatile 256 × 8 read/write data memory, five 8-bit I/O ports, one 8-bit input port, two 16-bit timer/event counters (identical to the timers of the 80C51), an additional 16-bit timer coupled to capture and compare latches, a 15-source two-priority-level, nested interrupt structure, an 8-input A/D, a dual DAC pulse width modulated interface, two serial interfaces (UART and I²C-bus), a "watchdog" timer and on-chip oscillator and timing circuits. For systems that require extra capability, the 8XC552 can be expanded using standard TTL compatible memories and logic.

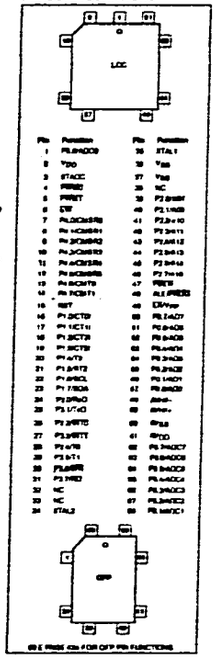
In addition, the 8XC562 has two software selectable modes of power reduction — idle mode and power-down mode. The idle mode freezes the CPU while allowing the RAM, timers, serial ports, and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator causing all other chip functions to be inoperative.

The device also functions as an arithmetic processor having facilities for both binary and BCD arithmetic plus bit handling capabilities. The instruction set consists of over 100 instructions: 49 one-byte, 45 two byte, and 17 three-byte. With a 16MHz (24MHz) crystal, 58% of the instructions are executed in 0.75μs (0.5μs) and 40% in 1.5μs (1μs). Multiply and divide instructions require 3μs (2μs).

FEATURES

- 80C51 central processing unit
- 8k × 8 ROM expandable externally to 64k bytes
- An additional 16-bit timer/counter coupled to four capture registers and three compare registers
- Two standard 16-bit timer/counters
- 256 × 8 RAM, expandable externally to 64k bytes
- Capable of producing eight synchronized, timed outputs
- A 10-bit ADC with eight multiplexed analog inputs
- Two 8-bit resolution, pulse width modulation outputs
- Five 8-bit I/O ports plus one 8-bit input port shared with analog inputs
- I²C-bus serial I/O port with byte oriented master and slave functions
- Full-duplex UART compatible with the standard 80C51
- On-chip watchdog timer
- Three speed ranges
  - 16MHz
  - 24MHz
  - 30MHz (in preparation)
- Extended temperature ranges
- OTP package available

PIN CONFIGURATIONS



Pin	Function	Pin	Function
1	P5.0/ADC0	39	XTAL1
2	$T2_D$	38	V_{SS}
3	ST/$\overline{ACC}$	37	V_{DD}
4	$\overline{PSEN}$	36	NC
5	$\overline{PRET}$	35	P3.0/$\overline{RD}$
6	$\overline{EW}$	34	P3.1/$\overline{WR}$
7	P5.2/ADC2/$\overline{INT0}$	33	P3.2/$\overline{INT0}$
8	P5.3/ADC3/$\overline{INT1}$	32	P3.3/$\overline{INT1}$
9	P5.4/ADC4/T2	31	P3.4/T0
10	P5.5/ADC5/$\overline{T2}$	30	P3.5/T1
11	P5.6/ADC6/$\overline{T2EX}$	29	P3.6/$\overline{WR}$
12	P5.7/ADC7/$\overline{T2EX}$	28	P3.7/$\overline{RD}$
13	P4.0/CMT0	27	XTAL2
14	P4.1/CMT1	26	RST
15	$\overline{RET}$	25	ALE/$\overline{PROG}$
16	P1.0/CT0I	24	$\overline{EA}$/Vpp
17	P1.1/CT1I	23	P0.0/AD0
18	P1.2/CT2I	22	P0.1/AD1
19	P1.3/CT3I	21	P0.2/AD2
20	P1.4/T3	20	P0.3/AD3
21	P1.5/$\overline{RT2}$	19	P0.4/AD4
22	P1.6/SCL	18	P0.5/AD5
23	P1.7/SDA	17	P0.6/AD6
24	P2.0/PWM0	16	P0.7/AD7
25	P2.1/PWM1	15	$\overline{CE}$
26	P2.2/$\overline{INT6}$	14	P2.7/A15
27	P2.3/$\overline{INT7}$	13	P2.6/A14
28	P2.4/$\overline{INT8}$	12	P2.5/A13
29	P2.5/$\overline{INT9}$	11	P2.4/A12
30	P2.6/$\overline{INT10}$	10	P2.3/A11
31	P2.7/$\overline{INT11}$	9	P2.2/A10
32	NC	8	P2.1/A9
33	NC	7	P2.0/A8
34	XTAL3	6	P5.1/ADC1

68 E PINS +2x FOR OTP PIN FUNCTIONS

DC ELECTRICAL CHARACTERISTICS

V_{SS}, AV_{SS} = 0V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS MIN	LIMITS MAX	UNIT
I_{DD}	Supply current operating	See notes 1 and 2			
	PCB8XC552 5-16	f_{osc} = 16MHz		45	mA
	PCF8XC552 5 16	f_{osc} = 16MHz		45	mA
	PCA8XC562 5-16	f_{osc} = 16MHz		40	mA
	87C552	f_{osc} = 16MHz		35	mA
	PCB8XC552 5 24	f_{osc} = 24MHz		55	mA
	PCF8XC562 5 24	f_{osc} = 24MHz		55	mA
I_{ID}	Idle mode	See notes 1 and 3			
	PCB8XC552 5 16	f_{osc} = 16MHz		10	mA
	PCF8XC552 5 16	f_{osc} = 16MHz		10	mA
	PCA8XC562 5 16	f_{osc} = 16MHz		9	mA
	87C552	f_{osc} = 16MHz		7	mA
	PCB8XC552 5 24	f_{osc} = 24MHz		12.5	mA
	PCF8XC552 5 24	f_{osc} = 24MHz		12.5	mA
I_{PD}	Power-down current	See notes 1 and 4 $2V < V_{PD} < V_{DD}$ max			
	PCB8XC552			50	μA
	PCF8XC552			50	μA
	PCA8XC562			150	μA
	87C552			50	μA
Inputs					
V_{IL}	Input low voltage, except EA, P1.6, P1.7		-0.5	$0.2V_{DD}$-0.1	V
V_{IL1}	Input low voltage to EA		0.5	$0.2V_{DD}$-0.3	V
V_{IL2}	Input low voltage to P1.6/SCL, P1.7/SDA[8]		-0.5	$0.3V_{DD}$	V
V_{IH}	Input high voltage, except XTAL1, RST		$0.2V_{DD}$+0.9	V_{DD}+0.5	V
V_{IH1}	Input high voltage, XTAL1, RST		$0.7V_{DD}$	V_{DD}+0.5	V
V_{IH2}	Input high voltage, P1.6/SCL, P1.7/SDA[8]		$0.7V_{DD}$	6.0	V
I_{IL}	Logical 0 input current, ports 1, 2, 3, 4 except P1.6, P1.7	V_{in} = 0.45V		-50	μA
I_{TL}	Logical 1 to-0 transition current, ports 1, 2, 3, 4 except P1.6, P1.7	See note 6		-650	μA
I_{LI}	Input leakage current, port 0, EA, STADC, EW	0.45V < V_I < V_{DD}		10	μA
I_{LI2}	Input leakage current, P1.6/SCL, P1.7/SDA	0V < V_I < 6V 0V < V_{DD} < 5.5V		10	μA
I_{LI3}	Input leakage current, port 5	0.45V < V_I < V_{DD}		1	μA
Outputs					
V_{OL}	Output low voltage, ports 1, 2, 3, 4 except P1.6, P1.7	I_{OL} = 1.6mA[7]		0.45	V
V_{OL1}	Output low voltage, port 0, ALE, $\overline{PSEN}$, PWM0, PWM1	I_{OL} = 3.2mA[7]		0.45	V
V_{OL2}	Output low voltage, P1.6/SCL, P1.7/SDA	I_{OL} = 3.0mA[7]		0.4	V
V_{OH}	Output high voltage, ports 1, 2, 3, 4 except P1.6/SCL, P1.7/SDA	$-I_{OH}$ = 60μA $-I_{OH}$ = 25μA $-I_{OH}$ = 10μA	2.4 $0.75V_{DD}$ $0.9V_{DD}$		V V V
V_{OH1}	Output high voltage (port 0 in external bus mode, ALE, $\overline{PSEN}$, PWM0, PWM1)[9]	$-I_{OH}$ = 400μA $-I_{OH}$ = 150μA $-I_{OH}$ = 40μA	2.4 $0.75V_{DD}$ $0.9V_{DD}$		V V V
V_{OH2}	Output high voltage (RST)	$-I_{OH}$ = 400μA $-I_{OH}$ = 120μA	2.4 $0.9V_{DD}$		V V

PIN DESCRIPTION

MNEMONIC	PIN NO. PLCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{DD}	2	72	I	Digital Power Supply: +5V power supply pin during normal operation, idle and power-down mode
STADC	3	74	I	Start ADC Operation: Input starting analog to digital conversion (ADC operation can also be started by software)
PWM0	4	75	O	Pulse Width Modulation: Output 0
PWM1	5	76	O	Pulse Width Modulation: Output 1
EW	6	77	I	Enable Watchdog Timer: Enables the T3 watchdog timer and disables power-down mode
P0.0-P0.7	57-50	58-51	I/O	Port 0: Port 0 is an 8-bit open-drain bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application it uses strong internal pull-ups when emitting 1s. Port 0 is also used to input the code byte during programming and to output the code byte during verification
P1.0-P1.7	16-23	10-17	I/O	Port 1: 8-bit I/O port. Alternate functions include
	16-21	10-15	I/O	(P1.0-P1.5): Quasi-bidirectional port pins
	22,23	16-17	I/O	(P1.6, P1.7): Open drain port pins
	16-18	10-13	I	CT0I-CT3I (P1.0-P1.3): Capture timer input signals for timer T2
	20	14	I	T2 (P1.4): T2 event input
	21	15	I	RT2 (P1.5): T2 timer reset signal. Rising edge triggered.
	22	16	I/O	SCL (P1.6): Serial port clock line I^2C-bus
	23	17	I/O	SDA (P1.7): Serial port data line I^2C-bus
				Port 1 is also used to input the lower order address byte during EPROM programming and verification. Ad is on P1.0 etc.
P2.0-P2.7	39-46	36-42 43-47	I/O	Port 2: 8-bit quasi-bidirectional I/O port. Alternate function: High-order address byte for external memory (A08-A15). Port 2 is also used to input the upper order address during EPROM programming and verification. A8 is on P2.0, A9 on P2.1, through A12 on P2.4
P3.0-P3.7	24-31	18,20 23-27	I/O	Port 3: 8-bit quasi-bidirectional I/O port. Alternate functions include
	24	18		RxD (P3.0): Serial input port
	25	19		TxD (P3.1): Serial output port
	26	20		INT0 (P3.2): External interrupt
	27	23		INT1 (P3.3): External interrupt
	28	24		T0 (P3.4): Timer 0 external input
	29	25		T1 (P3.5): Timer 1 external input
	30	26		WR (P3.6): External data memory write strobe
	31	27		RD (P3.7): External data memory read strobe
P4.0-P4.7	7-14	80, 1,2 4-8	I/O	Port 4: 8-bit quasi-bidirectional I/O port. Alternate functions include
	7-12	80, 1,2 4-6	O	CM5R0-CM5R6 (P4.0-P4.6): Timer T2 compare and set/reset outputs on a match with timer T2
	13, 14	7, 8	O	CMT0, CMT1 (P4.6, P4.7): Timer T2 compare and toggle outputs on a match with timer T2
P5.0-P5.7	68,62 1	71-64	I	Port 5: 8-bit input port. ADC0- ADC7 (P5.0-P5.7): Alternate function. Eight input channels to ADC
RST	15	9	I/O	Reset: Input to reset the 83C552. It also provides a reset pulse as output when timer T3 overflows
XTAL1	35	32	I	Crystal Input 1: Input to the inverting amplifier that forms the oscillator, and input to the internal clock generator. Receives the external clock signal when an external oscillator is used
XTAL2	34	31	O	Crystal Input 2: Output of the inverting amplifier that forms the oscillator. Left open-circuit when an external clock is used

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO. PLCC	PIN NO. QFP	TYPE	NAME AND FUNCTION
V_{SS}	36, 37	34-38	I	Digital ground.
PSEN	47	48	O	Program Store Enable: Active-low read strobe to external program memory
ALE/PROG	48	49	O	Address Latch Enable: Latches the low byte of the address during accesses to external memory. It is activated every six oscillator periods. During an external data memory access, one ALE pulse is skipped. ALE can drive up to eight LS TTL inputs and handles CMOS inputs without an external pull-up. This pin is also the program pulse input (PROG) during EPROM programming
EA/V_{PP}	49	50	I	External Access: When EA is held at TTL level high, the CPU executes out of the internal program ROM provided the program counter is less than 8192. When EA is held at TTL low level, the CPU executes out of external program memory. EA is not allowed to float. This pin also receives the 12.75V programming supply voltage (V_{PP}) during EPROM programming
AV_{REF-}	58	59	I	Analog to Digital Conversion Reference Resistor: Low-end
AV_{REF+}	59	60	I	Analog to Digital Conversion Reference Resistor: High-end
AV_{SS}	60	61	I	Analog Ground
AV_{DD}	61	63	I	Analog Power Supply

NOTE:
1. To avoid "latch-up" effect at power-on, the voltage on any pin at any time must not be higher or lower than V_{DD} + 0.5V or V_{SS} – 0.5V, respectively.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier. The pins can be configured for use as an on-chip oscillator, as shown in the logic symbol, page 424.

To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected. There are no requirements on the duty cycle of the external clock signal, because the input to the internal clock circuitry is through a divide by two flip-flop. However, minimum and maximum high and low times specified in the data sheet must be observed.

RESET

A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. To insure a good power-on reset, the RST pin must be high long enough to allow the oscillator time to start up (normally a few milliseconds) plus two machine cycles. At power-on, the voltage on V_{DD} and RST must come up at the same time for a proper start-up.

IDLE MODE

In the idle mode, the CPU puts itself to sleep while all of the on-chip peripherals stay active. The instruction to invoke the idle mode is the last instruction executed in the normal operating mode before the idle mode is activated. The CPU contents, the on-chip RAM, and all of the special function registers remain intact during this mode. The idle mode can be terminated either by any enabled interrupt (at which time the process is picked up at the interrupt service routine and continued), or by a hardware reset which starts the processor in the same manner as a power-on reset.

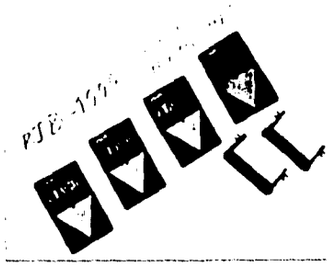
POWER-DOWN MODE

In the power-down mode, the oscillator is stopped and the instruction to invoke power-down is the last instruction executed. Only the contents of the on-chip RAM are preserved. A hardware reset is the only way to terminate the power-down mode. The control bits for the reduced power modes are in the special function register PCON. Table 1 shows the state of the I/O ports during low current operating modes.

Table 1. External Pin Status During Idle and Power-Down Modes

MODE	PROGRAM MEMORY	ALE	PSEN	PORT 0	PORT 1	PORT 2	PORT 3	PORT 4	PWM0/ PWM1
Idle	Internal	1	1	Data	Data	Data	Data	Data	High
Idle	External	1	1	Float	Data	Address	Data	Data	High
Power-down	Internal	0	0	Data	Data	Data	Data	Data	High
Power-down	External	0	0	Float	Data	Data	Data	Data	High

IC MEMORY CARDS



Card Edge Type Cards (with shutter)

40-pin cards with 8 bit wide interface

Device Type	Density
SRAMs	8 K × 8 to 1 M × 8
OTP ROMs	32 K × 8 to 1 M × 8
EEPROMs	8 K × 8 to 32 K × 8
MASK ROMs	128 K × 8 to 8 M × 8
FLASH EEPROMs	32 K × 8 to 1 M × 8

50-pin cards with 16 bit wide interface

Device Type	Density
SRAMs	32 K × 16 to 512 K × 16
OTP ROMs	64 K × 16 to 512 K × 16
MASK ROMs	128 K × 16 to 2 M × 16
FLASH EEPROMs	32 K × 16 to 512 K × 16

Two-piece Type Cards

JEIDA Ver.4 standard 68-pin two-piece cards with 16 bit wide interface

Device Type	Common Memory Density	Attribute Memory Density
SRAMs	32 K × 16 to 1 M × 16	2 Kbytes / 16 bytes / Nothing
OTP ROMs	128 K × 16 to 512 K × 16	2 Kbytes / Nothing
MASK ROMs	128 K × 16 to 2 M × 16	2 Kbytes / Nothing
FLASH EEPROM	64 K × 16 to 512 × 16	2 Kbytes / Nothing

Features

[redacted]

Card Edge Type Cards (with shutter)

• Precision shutter mechanism

The shutter mechanism opens automatically when the card is inserted and closes again when it is removed, protecting the card contacts and electronics from dust, dirt and static electricity. The shutter mechanism has a guaranteed insertion rate of 10,000 times.

• Front and rear stainless panels

In addition to improving mechanical strength, these panels also protect the card from the effects of electrical noise and shock.

• Card to connector polarization

The card and the connector are keyed, preventing incorrect insertion. This eliminates electrical malfunctions and prevents data corruption.

• Write protect switch (SRAM cards only)

When set, data in the card is protected from accidental overwriting. The switch status can be noted visually or by an output signal from the card.

• Battery replacement (SRAM cards only)

Data stored in the card is retained using an internal back-up battery (CR2016) while the card is unplugged.

The battery can be replaced only when the card is plugged into V_{cc}.

[redacted]

Two-piece type cards

• JEIDA standard [redacted]

Cards comply with Version 4 and Version 3 of the JEIDA specifications for IC memory cards.

• Resister function

The register area holds data on the type, density and access speed of the memory devices used in the card.

• Power supply and signal I/O control.

The main unit power can be turned on and off during insertion and also, the card can be inserted or removed with main unit power turned on without hazard to the card electronics or stored data.

• Card to connector polarization

The card and the connector are keyed, preventing incorrect insertion. This eliminates electrical malfunctions and prevents data corruption.

• Write protect switch (SRAM cards only)

When set, data in the card is protected from accidental overwriting. The switch status can be noted visually or by an output signal from the card.

• Battery replacement (SRAM cards only)

Data stored in the card is retained using an internal back-up battery (CR2025) while the card is unplugged. The battery can be replaced only when the card is plugged into V_{cc}.

• Battery holder lock (SRAM cards only)

This prevents the battery from loosening or accidentally falling out of the card.

• 60 pin type DRAM cards

• 16 bit wide interface

[redacted]

Typical Applications

• Memory expansion for laptop or hand-held computers

• Printer font cards

• Software cards for electronic organizers, and scientific equipment etc.

Two-piece Type Cards

JEIDA Ver.4 standard 68-pin 16 bit wide interface

Part Number	Memory Device	Density	Access Time (ns)	Power Consumption (mA)		Operating Temperature (deg. C)	Storage Temperature (deg. C)	Data Retention Period (years) See note 1.	Erasable Unit
				Active	Standby				
AWB065 SD20/30/40	Mixed CMOS SRAM	32 K × 16	200	120	1.5	0 to 60	-20 to 60	10 (TYP.)	Chip
AWB129 SD20/30/40		64 K × 16						8 (TYP.)	
AWB257 SD20/30/40		128 K × 16						8 (TYP.)	
AWB513 SD20/30/40		256 K × 16						5 (TYP.)	
AWB101 SD20/30/40		512 K × 16						2.5 (TYP.)	
AWB201 SD20/30/40		1 M × 16						1 (TYP.)	
OWB257 SDX0/Y0	CMOS PROM	128 K × 16	250	60	1	0 to 60	-20 to 60	—	
OWB513 SDX0/Y0		256 K × 16							
OWB101 SDX0/Y0		512 K × 16							
KWB257 SDX0/Y0	CMOS mask ROM	128 K × 16		60	1	0 to 60	20 to 60	—	
KWB513 SDX0/Y0		256 K × 16							
KWB101 SDX0/Y0		512 K × 16							
KWB201 SDX0/Y0		1 M × 16							
KWB401 SDX0/Y0		2 M × 16							
HWB065 SDX0/Y0	CMOS flash EEPROM	64 K × 16	250	60	1	0 to 60	-20 to 60	—	Chip
HWB257 SDX0/Y0		128 K × 16							
HWB513 SDX0/Y0		256 K × 16							
HWB101 SDX0/Y0		512 K × 16							

Attribute Memory Type

Series	Part Number	Attribute Memory Density	Description
AWB series	SD20	2 Kbytes	EEPROM
	SD30	16 bytes	Battery backup
	SD40		No attribute memory
OWB/KWB/HWB series	SD*0	2 Kbytes	EEPROM
	SD*0		No attribute memory

Note:
1. Data retention periods are those obtained when using a CR2025 type battery at Ta = 25°C (77°F) and are for reference only. Note that in order to retain stored data, batteries should only be replaced while the card is inserted in the main unit or attached to an alternate power source.

Two-piece Type Cards

JEIDA Ver.4 standard 68-pin 16 bit wide interface

Pin Number	Pin Name	Description
1	GND	Ground
2	D3	Data input output
3	D4	Data input output
4	D5	Data input output
5	D6	Data input output
6	D7	Data input output
7	$\overline{CE1}$	Chip enable input $\overline{CE1}$ even byte (D0 to D7)
8	A10	Address line
9	$\overline{OE}$	Output enable input
10	A11	Address line
11	A9	Address line
12	A8	Address line
13	A13	Address line
14	A14	Address line
15	WE PGM	Write enable program when PROM
16	RDY BSY	(EEPROM)
17	V_{cc}	Power supply input
18	V_{pp}	PROM writing Power supply input Power supply input
19	A16	Address line
20	A15	Address line
21	A12	Address line
22	A7	Address line
23	A6	Address line
24	A5	Address line
25	A4	Address line
26	A3	Address line
27	A2	Address line
28	A1	Address line
29	A0	Address line
30	D0	Data input output
31	D1	Data input output
32	D2	Data input output
33	WP	Write protect output Write is enabled when LOW disabled when HIGH
34	GND	Ground
35	GND	Ground

Pin Number	Pin Name	Description
36	$\overline{CD1}$	Card detect output Connected to ground internally
37	D11	Data input output
38	D12	Data input output
39	D13	Data input output
40	D14	Data input output
41	D15	Data input output
42	$\overline{CE2}$	Chip enable input $\overline{CE2}$ odd byte (D8 to D15)
43	RFSH	(PSRAM)
44	RFU	No connect
45	RFU	No connect
46	A17	Address line
47	A18	Address line
48	A19	Address line
49	A20	Address line
50	A21	Address line
51	V_{cc}	Power supply input
52	V_{pp}	PROM writing Power supply input Power supply input
53	A22	Address line
54	A23	Address line
55	A24	Address line
56	A25	Address line
57	RFU	No connect
58	RFU	No connect
59	RFU	No connect
60	RFU	No connect
61	$\overline{REG}$	Register select input
62	BV_2	Battery voltage detect 2
63	BV_1	Battery voltage detect 1
64	D8	Data input output
65	D9	Data input output
66	D10	Data input output
67	$\overline{CD2}$	Card detect output Connected to ground internally
68	GND	Ground

MOTOROLA

MC146818 Addendum

Advance Information

REAL-TIME CLOCK PLUS RAM (RTC)
Advance Information Data Sheet
ADI-856-R1

The following information is an addition to POWER DOWN CONSIDERATIONS found on page 11 of the MC146818 Advance Information Data Sheet (ADI-856-R1).

MC146818s with the date code of 8N4Xxxxx and GC6xxxx require a synchronization of the $\overline{CE}$ pin with address strobe. The following circuit will satisfy that condition, and also show a typical application of power down circuitry.

If $\overline{CE}$ is grounded at all times (no power down required) the following circuit need not be used.

MOTOROLA

MC146818

Advance Information

REAL-TIME CLOCK PLUS RAM (RTC)

The MC146818 Real-Time Clock plus RAM is a peripheral device which includes the unique MOTEL concept for use with various microprocessors, microcomputers, and larger computers. This part contains many unique features in a complete time-of-day clock with alarm and one hundred year calendar, a programmable periodic interrupt and square wave generator, and 50 bytes of low power static RAM. The MC146818 uses high speed CMOS technology to interface with 1 MHz processor buses, while consuming very little power.

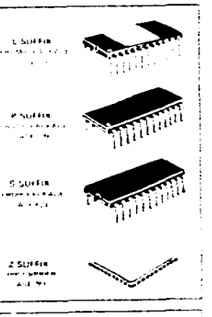
The real-time clock plus RAM has two distinct uses. First, it is designed as a battery powered CMOS part in an otherwise NMOS or TTL system, including all the common battery backed-up functions such as RAM, time and calendar. Secondly, the MC146818 may be used with a CMOS microprocessor to replace the software of the timekeeping, interrupt and 50 bytes of the available RAM of an MCU such as the MC146805E2.

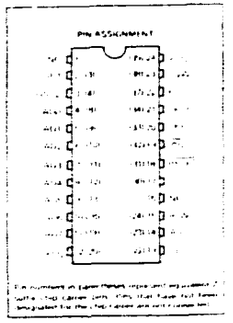
- Low Power, High Speed, Hot Remote CMOS
- Internal Time Base and Oscillator
- Counts Seconds, Minutes, and Hours of the Day
- Counts Days of the Week, Date, Month, and Year
- 3 V to 6 V Operation
- Time Base Input Options: 4.194304 MHz, 1.048576 MHz, or 32.768 kHz
- Time Base Oscillator for Parallel Resonant Crystals
- 10 to 230 μW Typical Operating Power at Low Frequency Time Base
- 4.0 to 20 mW Typical Operating Power at High Frequency Time Base
- Binary or BCD Representation of Time, Calendar, and Alarm
- 12- or 24-Hour Clock with AM and PM in 12-Hour Mode
- Daylight Savings Time Option
- Automatic End-of-Month Recognition
- Automatic Leap-Year Compensation
- Microprocessor Bus Compatible
- MOTEL Circuit for Bus Universality
- Multiplexed Bus for Pin Efficiency
- Interfaced with Software as 64 RAM Locations
- 14 Bytes of Clock and Control Registers
- 50 Bytes of General-Purpose RAM
- Status Bit Indicates Data Integrity
- Bus Compatible Interrupt Signals ($\overline{IRQ}$)
- Three Interrupts are Separately Software Maskable and Testable
  - Time of Day Alarm, Once per Second to Once per Day
  - Periodic Rates from 61 μs to 500 ms
  - End-of-Clock Update Cycle
- Programmable Square-Wave Output Signal
- Clock Output May Be Used as Microprocessor Clock Input
  - At Time Base Frequency of 4.194304
- 24 Pin Dual-In-Line Package
- Chip Carrier Also Available

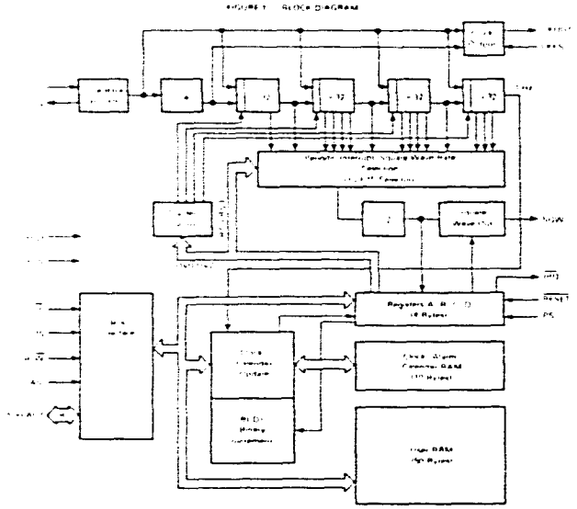
CMOS

HIGH PERFORMANCE SILICON-GATE COMPLEMENTARY MOS

REAL TIME CLOCK PLUS RAM







MAXIMUM RATINGS (Voltages referenced to V_{SS})

Rating	Symbol	Value	Unit
	V_{DD}		V
	V_{in}		V
	I		mA
	T_A		°C
	T_{stg}		°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
	θ_{JA}		°C/W

DC ELECTRICAL CHARACTERISTICS (V_{DD} = 5 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H, unless otherwise noted)

Characteristic	Symbol	Min	Max	Unit
Frequency of Oscillation	f_{osc}	32.768	32.768	kHz
Output Voltage	V_{OL}		0.4	V
I_{Load} = 1.0 mA	V_{OH}	V_{DD} - 0.7		
V_{DD} Supply Current C_L = 0 pF, f_{osc} = 32.768 kHz, C_L = 15 pF, SQW Disabled, $\overline{CE}$ = V_{DD} - 0.2, C_L (OSC2) = 10 pF	I_{DD3}		50	µA
V_{DD} Quiescent f_{osc} = DC, (OSC1 = DC All Other Inputs = V_{DD} - 0.2 V No Clocks	I_{DD4}		50	µA
Output High Voltage I_{Load} = -0.25 mA, All Outputs	V_{OH}	2.7		V
Output Low Voltage I_{Load} = 0.25 mA, All Outputs	V_{OL}		0.3	V
Input High Voltage AD0-AD7, DS, AS, R/W, CE RESET, CS0, PS, OSC1	V_{IH}	2.1 2.4	V_{DD} V_{DD}	V
Input Low Voltage (All Inputs)	V_{IL}	V_{SS}	0.8	V
Input Current All Inputs	I_{in}		±1	µA
Three-State Leakage $\overline{IRQ}$, AD0-AD7	I_{TSL}		±10	µA

DC ELECTRICAL CHARACTERISTICS (V_{DD} = 5 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H, unless otherwise noted)

Characteristic	Symbol	Min	Max	Unit
Frequency of Oscillation	f_{osc}	32.768	4194.304	kHz
Output Voltage	V_{OL}		0.4	V
I_{Load} = 1.0 mA	V_{OH}	V_{DD} - 0.7		
V_{DD} Supply Current C_L = 0 pF, f_{osc} = 4.194304 MHz, C_L = 15 pF, SQW Disabled, $\overline{CE}$ = V_{DD} - 0.2, C_L (OSC2) = 10 pF f_{osc} = 1.048576 MHz f_{osc} = 32.768 kHz	I_{DD1} I_{DD2} I_{DD3}		3 800 50	mA µA µA
V_{DD} Quiescent f_{osc} = DC, (OSC1 = DC All Other Inputs = V_{DD} - 0.2 V No Clocks	I_{DD4}		50	µA
Output High Voltage I_{Load} = -1.6 mA, AD0-AD7, CEOUT) I_{Load} = -1.0 mA, SQW)	V_{OH}	4.1		V
Output Low Voltage I_{Load} = 3.6 mA, AD0-AD7, CEOUT) I_{Load} = 1.0 mA, $\overline{IRQ}$ and SQW)	V_{OL}		0.4	V
Input High Voltage CE, CS0, ±10, AD7, DS, AS, R/W, CE, PS RESET OSC1	V_{IH}	V_{DD} - 2.0 V_{DD} - 0.8 V_{DD} - 1.0	V_{DD} V_{DD} V_{DD}	V
Input Low Voltage AD0-AD7, DS, AS, R/W, CE, PS CS0, PS, RESET OSC1	V_{IL}	V_{SS} V_{SS} V_{SS}	0.8 0.8 0.8	V
Input Current All Inputs	I_{in}		±1	µA
Three-State Leakage $\overline{IRQ}$, AD0-AD7	I_{TSL}		±10	µA

Apéndice B

Listado de Programas

MICROCONTROLADOR

COMPUTADORA PERSONAL

PAGINACION VARIA

COMPLETA LA INFORMACION

```
; ****** PROGRAMA PARA DECODIFICAR     **********
; *****   CODIGO DE BARRAS   CODIGO 39    **********

; ******************** DEFINICION DE REGISTROS
********************    PARA EL TIMER 2

IENI            EQU 0E8H
IEN0            EQU 0A8H
TM2CON          EQU 0EAH
TM2IR           EQU 0C8H
CTCON           EQU 0EBH
CT0H            EQU 0CCH
CT0L            EQU 0ACH
P4              EQU 0C0H

; ******************** VALORES Y REGISRSTROS
;********************      USADOS

NUMDATOS        EQU 130
NCARACT         EQU 028H

; ******************** DEFINICION DE REGISTROS ;
******************** PARA LOS RANGOS DE CONV
PDL             EQU 020H
PDH             EQU 021H
PGL             EQU 022H
PGH             EQU 023H
EDL             EQU 024H
EDH             EQU 025H
EGL             EQU 026H
EGH             EQU 027H


; ********************** REGISTROS PARA EL RELOJ ;
********************** DE TIEMPO REAL
REGA            EQU 200AH
REGB            EQU 200BH
REGC            EQU 200CH
REGD            EQU 200DH
RTR             EQU 1H
SEG             EQU 2000H
MIN             EQU 2002H
HORAS           EQU 2004H
DIASEM          EQU 2006H
DIAMES          EQU 2007H
MES             EQU 2008H
YEAR            EQU 2009H


; ******************** TABALA DE CARACTERES DEL ;
********************   C O D I G O  39
            ORG 1007H
            DB '*'
            ORG 100EH
            DB '"'
            ORG 1103H
            DB 'Q',0FFH,'N','T',0FFH,0FFH,'L','S',0FFH,'P'
            ORG 1111H
            DB 'K','R',0FFH,'O',0FFH,0FFH,0FFH,'M'
            ORG 1203H
            DB 'G',0FFH,'D','J',0FFH,0FFH,'B','I',0FFH,'F'
            ORG 121H
            DB 'A','H',0FFH,'E',0FFH,0FFH,0FFH,'C'
            ORG 1403H
            DB '7',0FFH,'4','0',0FFH,0FFH,'2','9',0FFH,'6'
            ORG 141H
            DB '1','8',0FFH,'5',0FFH,0FFH,0FFH,'3'
            ORG 1803H
            DB '-',0FFH,'X','*',0FFH,0FFH,'V',' ',0FFH,'Z'
            ORG 181H
            DB 'U','.',0FFH,'Y',0FFH,0FFH,0FFH,'W'
            ORG 1700H
            DB '%'
            ORG 1B00H
            DB '+'
            ORG 1D00H
            DB '/'
            ORG 1E00H
            DB '$'

;
*****************************************************
**; ********** PROGRAMA PRINCIPAL

            ORG 0000H
            AJMP 80H

            ORG 33H
            AJMP MEASURE
SALIDA33    RETI

            ORG 4DH
            AJMP OUTCARD
SALIDA4B    RETI

            ORG 73H
            AJMP VERIFLEC
SALIDA73    RETI

; ******************** INICIALIZACION DE REGISTROS

            ORG 80H
            MOV P4,#00011111B
            LCALL RETRA25
            LCALL INIDISP
;           LCALL DISPCLR
            MOV NUMMENH,#1DH
            MOV RENGLON,#80H
            MOV NUMMEN,#90H
            LCALL LEEROM            ;
            LCALL RETRA25
            LCALL AVISOSIC
            LCALL WRENABLE
            LCALL BAT
            MOV TM2IR,#00H
            CLR ET0                 ;
            CLR ET1                 ;
            LCALL LEETCRAM
            MOV P4,#00011111B
```

```
                LCALL WATCHDOG
                MOV NUMMENH,#1EH
                MOV R0,#00H
                MOV SORTIR,#00H          ;
                MOV CONT4,#10H
                MOV CONT5,#00
                MOV CONT6,#00
                MOV DPTR,#0900H          ;
                MOV P4,#00010000B
                MOV A,#0FFH
                MOVX @DPTR,A             ;
                MOV ERRORES,#00
                MOV BANDERA,#10100000B
                MOV TH1,#0E8H
                MOV PCON,#00H
                MOV TMOD,#00100000H
                MOV SCON,#01010000B      ;
                MOV TCON,#01000000B
                MOV IEN0,#10010101B      ;
                MOV IEN1,#00000001B
                MOV TM2CON,#00000000B
                MOV CTCON,#00000011B     ;

********************************* POLEO

OTRA    LCALL WATCHDOG
        MOV P4,#0FFH             ;
        MOV R7,BANDERA           ;

;-----------------------------------------------------------------------
                CJNE R7,#11000000B,PAI
                AJMP PBI
PAI             LJMP POL
PBI             MOV BANDERA,#10100000B
                LCALL DISPCLR
                LCALL AVISOSIC
                LCALL BAT

                MOV RENGLON,#80H
                MOV NUMMEN,#0D0H
                LCALL LEEROM             ;
                 MOV RENGLON,#0C0H
                MOV NUMMEN,#0E0H
                LCALL LEEROM             ;
                LCALL RET5MS
                LCALL RET5MS
                LCALL RET5MS
                MOV IEN0,#00
                MOV P4,#00011111B
                MOV CONT7,#00
                MOV R5,#20
IN1              MOV SBUF,#3EH
SIGIN1          JNB TI,SIGIN1
                CLR TI                   ;  >
                DJNZ R5,IN1              ;
                 MOV P4,CONT6            ;
                MOV DPL,CONT4            ;
                MOV DPH,CONT5            ;
                MOVX A,@DPTR
                CJNE A,#24H,RFCINC
                INC CONT7
                AJMP KGROWING
RFCINC          CJNE A,#0FFH,KGROWING
                MOV SBUF,#'!'
FTRANS8         JNB TI,FTRANS8           ;
                CLR TI                   ;
                CLR P1.2
                MOV P4,#00011111B
                LCALL RETRASO2
                SETB P1.2
                MOV CONT4,#10H           ;
                MOV CONT5,#00H           ;
                MOV CONT6,#00H           ;
                AJMP FINT

KGROWING        MOV R7,CONT4
                CJNE R7,#0FFH,MASL
                AJMP MASH
MASL            INC CONT4
                AJMP AJUERITA
MASH             MOV CONT4,#00H
                MOV R7,CONT5
                CJNE R7,#0FFH,MASLI
                AJMP INCPAGI
MASLI            INC CONT5
                AJMP AJUERITA
INCPAGI         MOV CONT5,#00H
                INC CONT6
AJUERITA         MOV P4,#00011111B
                MOV SBUF,A
FINTRANS        JNB TI,FINTRANS
                CLR TI
                LCALL WATCHDOG
                MOV R7,CONT7
                CJNE R7,#36,SIGTRAN
                MOV SBUF,#'&'
FTRANS          JNB TI,FTRANS
                CLR TI
FINT            MOV P4,#00011111B
                MOV IEN0,#10010101B

;-----------------------------------------------------------------------;
-----------------------------------------------------------------------

POL             CJNE R7,#10100000B,POL1
                MOV BANDERA,#10001000B
                 MOV R5,#100
EEE1            LCALL WATCHDOG
                LCALL DISPCLR
                 MOV RENGLON,#80H
                MOV NUMMEN,#10H
                LCALL LEEROM             ;
                 MOV RENGLON,#0C0H
                MOV NUMMEN,#20H
                LCALL LEEROM             ;
                 LCALL RET5MS
                LCALL RET5MS
                 DJNZ R5,EEE1

;-----------------------------------------------------------------------;
-----------------------------------------------------------------------
```

```
POL1        CJNE R7,#10001000B,POL2
            DESPLIEGA FECHA Y
            LJMP PL2A
POL2        LJMP POL3
PL2A        LCALL WATCHDOG
            MOV BANDERA,#10100000B
            MOV R7,#0CH
            MOV P4,#00010000B
ACTUAL      LCALL DISPCLR
             MOV P4,#00011111B
            LCALL BUSYDISP          ;
            MOV DPTR,#IR
            MOV A,#080H
            MOVX @DPTR,A
             LCALL RTRACC
            MOV P4,#00011111B       ;
            MOV DPTR,#DIASEM
            MOVX A,@DPTR            ;
            MOV AUXI,A
            LCALL BUSYDISP          ;
             MOV DPH,#1BH
            MOV DPL,AUXI
            CLR A
            MOVC A,@A+DPTR
            MOV DPTR,#DRW           ;
            MOVX @DPTR,A            ;
            MOV A,AUXI
            ADD A,#7
            MOV AUXI,A
            LCALL BUSYDISP
            MOV DPH,#1BH
            MOV DPL,AUXI
            CLR A
            MOVC A,@A+DPTR
            MOV DPTR,#DRW           ;
             MOVX @DPTR,A
            MOV P4,#00011111B       ;
            LCALL BUSYDISP
            MOV DPTR,#IR
            MOV A,#085H
            MOVX @DPTR,A
             MOV CONTI,#00
            MOV DPTR,#MES
            MOVX A,@DPTR            ;
            ADD A,#10H
            MOV DPL,A
            MOV DPH,#1BH
            CLR A
            MOVC A,@A+DPTR
            MOV AUXI,A
            LCALL BUSYDISP
            MOV DPH,#1BH
            MOV DPL,AUXI
            CLR A
            MOVC A,@A+DPTR
            MOV DPTR,#DRW           ;
            MOVX @DPTR,A            ;
            INC AUXI
            INC CONTI
            MOV R6,CONTI
            CJNE R6,#03,LLENI
            MOV P4,#00011111B       ;
            LCALL BUSYDISP
            MOV DPTR,#IR
            MOV A,#089H
            MOVX @DPTR,A
            MOV DPTR,#DIAMES
            MOVX A,@DPTR
            LCALL LEERTR
            LCALL BUSYDISP
            MOV DPTR,#DRW
            MOV A,#2CH
            MOVX @DPTR,A            ;
            LCALL MILCEN
             MOV DPTR,#YEAR
            LCALL LEERTR
            LCALL WATCHDOG
            MOV P4,#00011111B       ;
            LCALL BUSYDISP
            MOV DPTR,#IR
            MOV A,#0C4H
            MOVX @DPTR,A
            LCALL RTRACC
            MOV DPTR,#HORAS
            MOVX A,@DPTR
            LCALL LEERTR
            LCALL DOSPUN
             MOV DPTR,#MIN
            MOVX A,@DPTR
            LCALL LEERTR
            LCALL DOSPUN            ;
             MOV DPTR,#SEG
            MOVX A,@DPTR
            LCALL LEERTR            ;
            MOV R5,#20H
ESPERA      LCALL RET5MS
             MOV P4,#00011111B
            DJNZ R5,ESPERA
            DJNZ R7,OTRA5
            AJMP POL3
OTRA5       LJMP ACTUAL
;-------------------------------------------------------------------;
;-------------------------------------------------------------------
POL3        CJNE R7,#00001111B,POL4
            LCALL INIRTR
; ------------------------------------------------------------------;
;-------------------------------------------------------------------
POL4        CJNE R7,#11000011B,POL5

             LCALL WATCHDOG
            MOV BANDERA,#10100000B

            MOV P4,#00011111B
            MOV TCON,#00
             CLR PL2
            LCALL RETRASO2
            SETB PL2
            MOV TH1,VTRANS
            MOV PCON,#00H
            MOV TMOD,#00100000B
```

```
                MOV SCON,#01010000B
                MOV TCON,#01000000B
                MOV TCON,#01000000B
                MOV R5,#2
INI8            MOV SBUF,#3EH
SIGINI8         JNB TI,SIGINI8
                CLR TI
                DJNZ R5,INI8
                MOV SBUF,#46H
FINTRAN8        JNB TI,FINTRAN8          ;
                CLR TI
                MOV SBUF,#24H
FTRAN8          JNB TI,FTRAN8
                CLR TI
                MOV R7,#1
                MOV R5,#5

POL5            LCALL WATCHDOG
                 LJMP OTRA

; ********** DESPLIEGA EL CONTENIDO DE LA RAM

MUESTRA         ACALL RETRASO2
                CJNE R0,#NUMDATOS,SAL
                MOV P4,#10011111B
                ACALL RETRASO2
                MOV P4,R0
                ACALL RETRASO2
                MOV DPTR,#0000H
                MOV A,R4
                MOV R1,A
MUES            MOV P4,#0FFH
                ACALL RETRASO2
                MOV P4,DPL
                ACALL RETRASO2
                MOV P4,#10010010B
                ACALL RETRASO2
                MOV DPH,#08H
                MOVX A,@DPTR
                MOV P4,A
                ACALL RETRASO2
                MOV DPH,#00
                INC DPTR
                DJNZ R1,MUES
SAL             RET

************************************************
**; ************** INICIALIZACION DE RELOJ DE T R

INIRTR          MOV P4,#RTR
                MOV DPTR,#REGB
                MOV A,#86H
                 MOVX @DPTR,A
                CLR A
                MOV DPTR,#REGA
                MOVX @DPTR,A
                MOV DPTR,#SEG
                MOVX @DPTR,A
                MOV DPTR,#MIN
                MOVX @DPTR,A
                MOV DPTR,#HORAS
                MOVX @DPTR,A
                MOV A,#02H
                MOV DPTR,#DIASEM
                MOVX @DPTR,A
                MOV DPTR,#MES
                MOVX @DPTR,A
                MOV A,#13H
                MOV DPTR,#DIAMES
                MOVX @DPTR,A
                MOV A,#5EH
                MOV DPTR,#YEAR
                MOVX @DPTR,A
                MOV A,#06H
                MOV DPTR,#REGB
                MOVX @DPTR,A
                RET

;
************************************************
**; ************ LIMPIA EL TIMER 2

RESTT2          MOV TM2CON,#00100000B
                MOV P4,#00H
                NOP
                NOP
                MOV P4,#0FFH
                NOP
                NOP
                NOP
                MOV IEN1,#10000001B
                MOV CTCON,#00000011B
                MOV TM2CON,#10000101B
                RET

;************************************************
**;******************** RETRASO2 DE 5 SEG

RETRASO2        MOV R2,#0FFH
CIC1:           DEC R2
                MOV R3,#0FFH
CIC2:           DEC R3
                MOV R5,#0AH
                ACC con 80
CIC3:           DEC R5
                CJNE R5,#00H,CIC3
                CJNE R3,#00H,CIC2
                CJNE R2,#00H,CIC1
                RET
************************************************
**; ******************** OTRA LECT

VERIFLEC        MOV TM2CON,#00000000B
                 MOV TM2IR,#00H
                MOV IEN1,#00H
                CJNE R0,#NUMDATOS,ERRORLEC
                AJMP RANGOS
SRANGOS         AJMP ASIGNA
SASIGNA         AJMP DECOD
SDECOD          AJMP TABLA
STABLA          AJMP HORA
```

```
SHORA   MOV IEN1,#00001000B
        MOV CTCON,#11000000B
        MOV P3,#10000001B
        ACALL RETRASO2
        MOV DPTR,#00H
        AJMP SALIDA73

ERRORLEC MOV P4,#10111001B
        ACALL RETRASO2
SALI    MOV IEN1,#10000001B
        MOV TM2CON,#10000100B
        MOV CTCON,#00000011B
        MOV DPTR,#0000
        MOV R0,#00
        AJMP SALIDA73

;
;*********************************************************
;*********************** OUT CARD

OUTCARD MOV IEN1,#00000000B
        MOV TM2IR,#00H
        MOV P4,#01111110B
        ACALL RETRASO2
        ACALL RETRASO2
        MOV IEN1,#00000001B
        MOV TM2CON,#00000100B
        MOV CTCON,#00000011B
        MOV R0,#00
        AJMP SALIDA4B

;********************** SELECCION AUTOMATICA
;************************** DE RANGOS

RANGOS  MOV DPTR,#00H
        MOVX A,@DPTR
        MOV PDH,A
        MOV DPTR,#2
        MOVX A,@DPTR
        CJNE A,PDH,DIF
DIF     JC MENOR
        MOV R5,PDH
        MOV PDH,A
        MOV PDL,R5
        AJMP S
MENOR   MOV PDL,A
S       MOV DPTR,#08H
        MOVX A,@DPTR
        CJNE A,PDH,DIFA
        AJMP SA
DIFA    JC MENORA
        MOV PDH,A
        AJMP SA
MENORA  CJNE A,PDL,DIFB
        AJMP SA
DIFB    JC MENORB
        AJMP SA
MENORB  MOV PDL,A
SA      MOV DPTR,#120
        MOVX A,@DPTR
        CJNE A,PDH,DIFC
        AJMP SB
DIFC    JC MENORC
        MOV PDH,A
        AJMP SB
MENORC  CJNE A,PDL,DIFD
        AJMP SB
DIFD    JC MENORD
        AJMP SB
MENORD  MOV PDL,A
SB      MOV DPTR,#122
        MOVX A,@DPTR
        CJNE A,PDH,DIFE
        AJMP SC
DIFE    JC MENORE
        MOV PDH,A
        AJMP SC
MENORE  CJNE A,PDL,DIFF
        AJMP SC
DIFF    JC MENORF
        AJMP SC
MENORF  MOV PDL,A
SC      MOV DPTR,#128
        MOVX A,@DPTR
        CJNE A,PDH,DIFG
        AJMP SD
DIFG    JC MENORG
        MOV PDH,A
        AJMP SD
MENORG  CJNE A,PDL,DIFH
        AJMP SD
DIFH    JC MENORH
        AJMP SD
MENORH  MOV PDL,A

;********************** PULSOS GRUESOS
**********************

SD      MOV DPTR,#04
        MOVX A,@DPTR
        MOV PGH,A
        MOV DPTR,#6
        MOVX A,@DPTR
        CJNE A,PGH,DIFI
DIFI    JC MENORI
        MOV R5,PGH
        MOV PGH,A
        MOV PGL,R5
        AJMP SE
MENORI  MOV PGL,A
SE      MOV DPTR,#124
        MOVX A,@DPTR
        CJNE A,PGH,DIFJ
        AJMP SF
DIFJ    JC MENORJ
        MOV PGH,A
        AJMP SF
MENORJ  CJNE A,PGL,DIFK
        AJMP SF
DIFK    JC MENORK
        AJMP SF
```

```
MENORK      MOV PGL,A
SF          MOV DPTR,#126
            MOVX A,@DPTR
            CJNE A,PGH,DIFL
            AJMP SG
DIFL        JC MENORL
            MOV PGH,A
            AJMP SG
MENORL      CJNE A,PGL,DIFM
            AJMP SG
DIFM        JC MENORM
            AJMP SG
MENORM      MOV PGL,A

;********************** ESPACIOS DELGADOS **********************

SG          MOV DPTR,#031H
            MOVX A,@DPTR
            MOV EDH,A
            MOV DPTR,#5
            MOVX A,@DPTR
            CJNE A,EDH,DIFN
DIFN        JC MENORN
            MOV R5,EDH
            MOV EDH,A
            MOV EDL,R5
            AJMP SH
MENORN      MOV EDL,A
SH          MOV DPTR,#07
            MOVX A,@DPTR
            CJNE A,EDH,DIFNN
            AJMP SI
DIFNN       JC MENORNN
            MOV EDH,A
            AJMP SI
MENORNN     CJNE A,EDL,DIFO
            AJMP SI
DIFO        JC MENORO
            AJMP SI
MENORO      MOV EDL,A
SI          MOV DPTR,#9
            MOVX A,@DPTR
            CJNE A,EDH,DIFP
            AJMP SJ
DIFP        JC MENORP
            MOV EDH,A
            AJMP SJ
MENORP      CJNE A,EDL,DIFQ
            AJMP SJ
DIFQ        JC MENORQ
            AJMP SJ
MENORQ      MOV EDL,A
SJ          MOV DPTR,#123
            MOVX A,@DPTR
            CJNE A,EDH,DIFR
            AJMP SK
DIFR        JC MENORR
            MOV EDH,A
            AJMP SK
MENORR      CJNE A,EDL,DIFS
            AJMP SK
DIFS        JC MENORS
            AJMP SK
MENORS      MOV EDL,A
SK          MOV DPTR,#125
            MOVX A,@DPTR
            CJNE A,EDH,DIFT
            AJMP SL
DIFT        JC MENORT
            MOV EDH,A
            AJMP SL
MENORT      CJNE A,EDL,DIFU
            AJMP SL
DIFU        JC MENORU
            AJMP SL
MENORU      MOV EDL,A
SL          MOV DPTR,#127
            MOVX A,@DPTR
            CJNE A,EDH,DIFV
            AJMP SM
DIFV        JC MENORV
            MOV EDH,A
            AJMP SM
MENORV      CJNE A,EDL,DIFW
            AJMP SM
DIFW        JC MENORW
            AJMP SM
MENORW      MOV EDL,A
SM          MOV DPTR,#119
            MOVX A,@DPTR
            CJNE A,EDH,DIFX
            AJMP SN
DIFX        JC MENORX
            MOV EDH,A
            AJMP SN
MENORX      CJNE A,EDL,DIFY
            AJMP SN
DIFY        JC MENORY
            AJMP SN
MENORY      MOV EDL,A

;********************** ESPACIOS GRUESOS **********************

SN          MOV DPTR,#01H
            MOVX A,@DPTR
            MOV EGH,A
            MOV DPTR,#121
            MOVX A,@DPTR
            CJNE A,EGH,DIFZ
DIFZ        JC MENORZ
            MOV R5,EGH
            MOV EGH,A
            MOV EGL,R5
            AJMP SO
MENORZ      MOV EGL,A

;********************** REAJUSTA LOS RANGOS **********************

SO          MOV A,PDL
            CJNE A,#00,RESTP
```

```
                AJMP SPGL
RESTP           DEC PDL
SPGL            MOV A,PGL
                SUBB A,PDH
                JZ ERRANT
                AJMP SIGT
ERRANT          AJMP ERRAN
SIGT            CJNE A,#01,DOSP
                AJMP REGALP
DOSP            CJNE A,#02H,MASDOSP
                DEC PGL
                AJMP REGALP
MASDOSP         CJNE A,#03,MASTRESP
                DEC PGL
                DEC PGL
                AJMP REGALP
MASTRESP        DEC PGL
                DEC PGL
                DEC PGL
REGALP          INC PGH
                INC PGH
                INC PGH
                INC PGH
                MOV A,EDL
                CJNE A,#00,RESTE
                AJMP SEGL
RESTE           DEC EDL
SEGL            MOV A,EGL
                SUBB A,EDH
                JZ ERRAN
                CJNE A,#01,DOSE
                AJMP REGALE
DOSE            CJNE A,#02H,MASDOSE
                DEC EGL
                AJMP REGALE
MASDOSE         CJNE A,#03,MASTRESE
                DEC EGL
                DEC EGL
                JMP REGALE
MASTRESE        DEC EGL
                DEC EGL
                DEC EGL
REGALE          INC EGH
                INC EGH
                INC EGH
                INC EGH
                AJMP SRANGOS
ERRAN           MOV P4,#10110000B
                ACALL RETRASO2
                AJMP SALI

;*************************************************
;*****************  CLASIFICA ESP Y PUL CON BITS

ASIGNA          MOV DPTR,#0000
                MOV R6,#00
CONVERT         MOV DPL,R6
                MOV P4,#00010000B
                MOVX A, @DPTR
                MOV R7,A
```

```
;*********  VERIFICA SI ES PULSO O ESPACIO
*****************************

ELECCION        MOV A,#01H
                ANL A,R6
                JZ PULSO

;****************  MIDE ESPACIOS
*************************************

ESPACIO         MOV A,R7
                CJNE A,LEDL,LEDL
                MOV A,#00H
                AJMP SALIR
LEDL            JC OUTRANG
                CJNE A,EDH,INRANG3
                MOV A,#00H
                AJMP SALIR
INRANG3         JC GUARDA3
                AJMP NEED
GUARDA3         MOV A,#00H
                AJMP SALIR
NEED            CJNE A,LEGL,LEGA
                MOV A,#01H
                AJMP SALIR
LEGA            JC UMBRAL
                CJNE A,EGH,INRANG04
                MOV A,#01H
                AJMP SALIR
INRANG04        JC GUARDA4
                AJMP OUTRANG
GUARDA4         MOV A,#01H
                AJMP SALIR

;*****************  MIDE PULSOS

PULSO           MOV A,R7
                CJNE A,PDL,LPDL
                MOV A,#00H
                AJMP SALIR
LPDL            JC OUTRANG
                CJNE A,PDH,INRANG
                MOV A,#00H
                AJMP SALIR
INRANG          JC GUARDA
                AJMP NEPD
GUARDA          MOV A,#00H
                AJMP SALIR
NEPD            CJNE A,PGL,LPGL
                MOV A,#01H
                AJMP SALIR
LPGL            JC UMBRAL
                CJNE A,PGH,INRANG02
                MOV A,#01H
                AJMP SALIR
INRANG02        JC GUARDA2
                AJMP OUTRANG
GUARDA2         MOV A,#01H
                AJMP SALIR
SALIR           MOV DPH,#02H
                MOV DPL,R6
```

```
                MOVX @DPTR,A
                INC R6
                MOV DPH,#00
                CJNE R6,#129,CONVERT
                AJMP SASIGNA
UMBRAL          MOV IEN1,#00H
                MOV P4,#11110000B
                ACALL RETRASO2
                MOV P4,R7
                ACALL RETRASO2
                MOV P4,R0
                ACALL RETRASO2
                AJMP ERRORLEC
OUTRANG         MOV IEN1,#00H
                MOV P4,#00001111B
                ACALL RETRASO2
                MOV P4,R6
                ACALL RETRASO2
                MOV P4,R7
                ACALL RETRASO2
                AJMP ERRORLEC

;****************** GUARDA EN BYTES EL TREN DE ;
***************** PULSOSO DEL DECODIFICADOR

DECOD           MOV R6,#00H
                MOV R5,#00H
                MOV R3,#00H
                MOV R1,#00H
                MOV R4,#00H
                MOV R2,#NUMDATOS
                MOV R7,#00
CONTINUE        MOV DPH,#02H
                MOV DPL,R4
                MOV P4,#00010000B
                MOVX A,@DPTR
                  JZ NFUE1
                INC R7
NFUE1           ORL A,R5
                RL A
                MOV R5,A
                INC R4
                 CJNE R1,#08H,SIGUESP
                AJMP ESPCAR
SIGUESP         INC R1
                MOV DPL,R4
                MOV P4,#00010000B
                MOVX A,@DPTR
                JZ NOFUE1
                INC R7
NOFUE1          ORL A,R6
                RL A
                MOV R6,A
                INC R4
                INC R1
                DEC R2
                DEC R2
                CJNE R2,#00H,CONTINUE
                AJMP SIDECOD
                INC R4
                CJNE R3,#03,ERRORX1
                  MOV R7,#00
                MOV DPH,#04H
                MOV DPL,R3
                MOV A,R5
                RR A
                  MOV P4,#00010000B
                MOVX @DPTR,A
                INC R3
                MOV DPL,R3
                MOV A,R6
                  RR A
                MOV P4,#00010000B
                MOVX @DPTR,A
                INC R3
                 MOV R1,#00H
                MOV R6,#00H
                MOV R5,#00H
                AJMP CONTINUE
ERRORX1         MOV P4,#10110110B
                ACALL RETRASO2
                AJMP SALI

; ********** GUARDA EL VALOR ASCII DEL CARACTER ;
************** EN RAM APARTIR DE LA DIR 800

TABLA           MOV R3,#00
                  MOV R4,#00
                  MOV A,R0
                MOV B,#10
                DIV AB
                MOV NCARACT,A           ;
SIGCARAC        MOV DPH,#04             ;
                MOV DPL,R3              ;
                MOV P4,#00010000B
                MOVX A,@DPTR            ;
                  MOV R5,A
                INC R3                  ;
                  MOV DPL,R3            ;
                MOVX  A,@DPTR           ;
                ORL A,#00010000B
                MOV R6,A
                INC R3
                  MOV DPL,R5            ;
                  MOV DPH,R6
                MOV A,#00
                MOVC A,@A+DPTR          ;
                  JZ ERRORXDC
                  MOV DPH,#08           ;
                MOV DPL,R4              ;
                MOVX @DPTR,A
                INC R4                  ;
                  DJNZ NCARACT,SIGCARAC
                AJMP STABLA
ERRORXDC        MOV P4,#01111010B
                ACALL RETRASO2
                AJMP SALI
;
*****************************************************
**, ********************** LECTURA DEL RELOJ

HORA            MOV R7,#00
```

```
OCUPADO     MOV DPTR,#REGB
            MOVX A,@DPTR
            ANL A,#80H
            CJNE A,#00H,OCUPADO
             MOV DPTR,#SEG
            MOVX A,@DPTR
              MOV DPH,#08
            MOV DPL,R4
              INC R4
              MOVX @DPTR,A
              MOV DPTR,#MIN
            MOVX A,@DPTR
              MOV DPH,#08
            MOV DPL,R4
              INC R4
            MOVX @DPTR,A
            MOV DPTR,#HORAS
            MOVX A,@DPTR
              MOV DPH,#08
              MOV DPL,R4
            INC R4
            MOVX @DPTR,A
            MOV DPTR,#DIASEM
            MOVX A,@DPTR
            MOV DPH,#08
            MOV DPL,R4
            INC R4
            MOVX @DPTR,A
            MOV DPTR,#DIAMES
            MOVX A,@DPTR
            MOV DPH,#08
            MOV DPL,R4
            INC R4
            MOVX @DPTR,A
            MOV DPTR,#MES
            MOVX A,@DPTR
            MOV DPH,#08
            MOV DPL,R4
            INC R4
            MOVX @DPTR,A
            MOV DPTR,#YEAR
            MOVX A,@DPTR
            MOV DPH,#08
            MOV DPL,R4
            MOVX @DPTR,A
            AJMP SHORA

;
;*****************************************************
;********** COMUNICACION PUERTO SERIE
PTOSER      LCALL WATCHDOG
            CLR RI
            MOV A,SBUF
            CJNE A,#3CH,XXX
            MOV RTRANS,#00
            AJMP NFIN1
XXX         MOV DPH,#09H
            MOV DPL,RTRANS
            MOV P4,#00010000B
            MOVX @DPTR,A
            INC RTRANS
            CJNE A,#24H,NFIN1
            MOV IEN0,#00000000B
            MOV DPTR,#0900H
            MOV P4,#00011111B
            MOVX A,@DPTR
            CJNE A,#40H,PTOSER1
            LJMP ACTRTR
PTOSER1     CJNE A,#26H,PTOSER2
            LJMP TRANS
PTOSER2     CJNE A,#23H,PTOSER3
            LJMP VELTRANS
PTOSER3     CJNE A,#4CH,PTOSER4
            LJMP LISTO
PTOSER4     CJNE A,#46H,PTOSER5          PTOSER5
            CJNE A,#'V',PTOSER6
            LJMP MATCH
PTOSER6     CJNE A,#'',NFIN1
            LJMP NETTOIC
NFIN1       MOV IEN0,#10010101B
NFIN2       LCALL WATCHDOG
            MOV DPTR,#0900H
            MOV P4,#00011111B
            MOV A,#0FFH
            MOVX A,@DPTR
            LJMP SALIDA23

;*****************************************************
;**;******* ACTUALIZA EL RELOJ DE TIEMPO REAL POR
PC

ACTRTR      LCALL WATCHDOG
            MOV P4,#00011111B
            MOV NUMMENH,#1FH
            MOV RENGLON,#80H
            MOV NUMMEN,#90H
            LCALL LEEROM
            MOV RENGLON,#0C0H
            MOV NUMMEN,#0A0H
            LCALL LEEROM
            MOV NUMMENH,#1EH
              MOV R1,#32H
            MOV DPTR,#90H
ACTRTR1     MOV P4,#00011111B
            MOVX A,@DPTR
            CJNE A,#24H,NACTRTR1
            AJMP CONVBBCD
NACTRTR1    MOV @R1,A
            INC R1
            INC DPTR
            AJMP ACTRTR1
CONVBBCD    MOV R6,#08
            MOV R1,#32H
SIGUEH      MOV A,@R1
            MOV AUXI,A
            LCALL BBCD
            MOV @R1,AUXI
            INC R1
            DJNZ R6,SIGUEH
            LCALL RETRA25
              LCALL INRTR
```

```
                MOV RTRANS,#00
                LCALL WATCHDOG
                LJMP NFIN1

;*****************************************************
;*******   CONVIERTE UN NUMERO DE BINARIO A
BCD

BBCD            LCALL WATCHDOG
                MOV R7,AUXI
                CJNE R7,#9,CHANGE
                AJMP SALBBCD
CHANGE          JC SALBBCD
                MOV A,#00
                MOV CONT3,#00
                MOV CONT2,#00
                MOV CONT1,#00
FALTA           INC CONT1
                INC CONT2
                MOV A,CONT2
                CJNE A,#10,NOPASA
                MOV CONT2,#00
                INC CONT3
NOPASA          MOV A,CONT1
                CJNE A,AUXI,FALTA
                CLR C
                MOV A,CONT3
                RL A
                RL A
                   RL A
                ORL A,CONT2
SALBBCD         RET

;*****************************************************
;*********************   TIEMPO

TIEMPO          LCALL WATCHDOG
                MOV IEN0,#00
                CLR P1.2
                CLR ET0
                MOV TH0,#00
                MOV TL0,#00
                MOV TH1,#0E8H
                MOV P4,#00011111B
                LCALL RETRASO2
                SETB P1.2
                MOV IEN0,#10010101B
                LJMP SALIDAB

                END;
                END
```

```
#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <ctype.h>
#include <process.h>

/* ********* DEFINICION DE TECLAS USADAS ************* */

#define BKS  0x08
#define CR   0x0D
#define ESC  0X1B
#define IMPRIME 0  /*IMPRIME CARACTER*/
#define INICIALIZA 1            /* INICIALIZA LA IMPRESORA*/
#define STATUS  2               /* printer status command */
#define PORTNUM 0               /* port number for LPT1 */


/******* VARIABLES TRASNMISION ******************/

const com1=0;
const LCR = 0x3FB;              /* Fija bits de stop/paro/paridad/longitud*/
const DLAL = 0x3F8;          /* Fija velocidad de trans */
const DLAH = 0x3F9;          /*        "          */
const LSR = 0x3FD;           /* Indica que el dato fue recibido o tranmitido */
const THR = 0x3F8;           /* registro que almacena el dato */

/* ********* DEFINICION DE FUNCIONES ************** */

void main (void);
int get_key();
void mem_arch_ic(void);
void leer_ic(void);
void mostrar_archivo(void);
void imprimir(void);
void formato(void);
void act_hora(void);
void vel_trans(void);
void salida(void);
void linea_com(void);
void ini_pto_ser(void);
void lee_datos_sis(void);
void limpia_palabra(void);
void limpia_archivos(void);
void imprime_archivo(void);
void checa_hora(void);
void checa_dia(void);
void checa_fecha(void);
void rtr(void);
void checa_sistema(void);
void manda_cambio_vel(void);
void fin_cambio_vel(void);
void ayuda(void);
void submenu_mostrar(void);
void periodo(void);
void dias_festivos(void);
void genera_archivos(void);
void dispositivo(void);
void lee_tcram();
```

```
void limpiate_ps(void);
void ordena_archivo_alfabetico(void);
void rapida_cadena();
void or_cadena();

void main()
{
/*  usado para leer caracteres especiales */
union scan
        {
        int c;
        char ch[2];
        }sc;
clrscr();

/* ************ inicializa el modo grafico ***************** */

registerbgidriver(EGAVGA_driver);
registerbgifont(sansserif_font);
initgraph(&gdriver,&gmode,"d:\\turboc\\BGI");

/***************** Selecciona colores de fondo y de dibujo ********/

setbkcolor(BLUE);
clearviewport();
rectangle(0,20,639,479);
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
settextjustify(CENTER_TEXT,CENTER_TEXT);
ayuda();

/***************** Pone la linea superior de opciones *************/

linea_com();
setfillstyle(SOLID_FILL,RED);
setcolor(RED);

bar(25,0,140,27);
setcolor(BLACK);
outtextxy(82,10,"ARCHIVOS");
outtextxy(82,20,"DE IC-RAM");
setcolor(WHITE);
checa_sistema();
limpia_palabra();
setcolor(WHITE);
outtextxy(320,455," Adquiere y/o Procesa la Informacion de ");
outtextxy(320,470," Asistencias, Faltas, Tiempos Extra ... ");
setcolor(BLACK);

/******************** lee el teclado ************************/

fflush(stdin);
do {
  sc.c=get_key();
  if(sc.ch[0]==0)                       /* es tecla especial */
    {
        if(sc.ch[1]==77)
                pos_barra=pos_barra+1;
        if(sc.ch[1]==75)
                pos_barra=pos_barra-1;
    }
  else                                  /* tecla normal */
```

```
        if (sc.ch[0]==CR)
                opcion_barra='s';

if (pos_barra==5)
        pos_barra=1;
if (pos_barra==0)
        pos_barra=4;
switch (pos_barra)
        {
        case 1 :
        if (opcion_barra=='n')
        {
          linea_com();
          setfillstyle(SOLID_FILL,RED);
          setcolor(RED);
          bar(25,0,140,27);
          setcolor(BLACK);
          outtextxy(82,10,"ARCHIVOS");
          outtextxy(82,20,"DE IC-RAM");
          ayuda();
          setcolor(WHITE);
          outtextxy(320,455," Adquiere, Procesa y/o Despliega la Informacion ");

          outtextxy(320,470,"    de Asistencias, Faltas, Tiempos Extra ... ");
          setcolor(BLACK);
        }
        if (opcion_barra=='s')
        {
        if (imagen_archivos_ic!=NULL)
        {
        getimage(10,30,160,130,imagen_archivos_ic);
        menu_arch_ic();
        setfillstyle(SOLID_FILL,RED);
        setcolor(RED);
        bar(10,30,160,55);
        setcolor(BLACK);
        outtextxy(80,44,"ADQUIRIR INF.");
        /****** Primer sub menu *****/
                do
                 {
                 if (pos_bar_y==4)
                   {
                   ayuda();
                   setcolor(WHITE);
                   outtextxy(320,455," Adquiere la Informacign de Chacado del ");
                   outtextxy(320,470,"    Reloj Checador o del Drive en PC    ");
                   setcolor(BLACK);
                   }
                 if (pos_bar_y==3)
                   {
                   ayuda();
                   setcolor(WHITE);
                   outtextxy(320,455,"    Procesa la Informacign de Chacado para     ");
                   outtextxy(320,470," Generar las Faltas, Retardos Tiempos Extra ... ");
                   setcolor(BLACK);
                   }
                 if (pos_bar_y==2)
                   {
                   ayuda();
                   setcolor(WHITE);
                   outtextxy(320,455,"    Manda a Impresion via Puerto Serie    ");
```

```
    outtextxy(320,470,"          un Aun Archivo                ");
    setcolor(BLACK);
    }
  if (pos_bar_y==1)
    {
    ayuda();
    setcolor(WHITE);
    outtextxy(320,455,"   Despliega en Pantalla la Informacion   ");
    outtextxy(320,470," de Faltas, Retardos, Tiempos Extras ... ");
    setcolor(BLACK);
    }
  sc c=get_key();
  if(sc ch[0]==0)                              /* es tecla especial */
{
if(sc ch[1]==72)
pos_bar_y=pos_bar_y++;
if(sc.ch[1]==80)
pos_bar_y=pos_bar_y--,
}
else
{                                              /* tecla normal */
if (sc ch[0]==CR)
opcion_barra_y='s';
f (sc.ch[0]==ESC)
{
salir_y='s';
pos_bar_y=6;
}
}
  if (pos_bar_y==5)
            pos_bar_y=1;
            if (pos_bar_y==0)
            pos_bar_y=4;
            switch (pos_bar_y)
    {
    case 4 :
  if (opcion_barra_y=='n')
    {
    menu_arch_ic();
    setfillstyle(SOLID_FILL,RED);
    bar(10,30,160,55),
    setcolor(BLACK);
    outtextxy(80,44,"ADQUIRIR INF.");
    }
  if (opcion_barra_y=='s')
  {
    if (imagen_leer_ic!=NULL)
    {
    getimage(140,100,460,280,imagen_leer_ic);
    leer_ic();
    ayuda();
    setcolor(WHITE),
    uuttextxy(320,455,"   Teclee el Nombre del Archivo donde se   ");
    outtextxy(320,470,"         Guardara la Informacion         ");
    setcolor(BLACK);
    moveto(180,205);
    i=0,
    settextjustify(LEFT_TEXT,CENTER_TEXT),
    fflush(stdin);
    do
    {
```

```
setcolor(BLACK);
outtext("_");
setcolor(LIGHTGRAY);
moveto(getx()-8,gety());
sc.c=get_key();
outtext("_");
moveto(getx()-8,gety());
setcolor(WHITE);
if(sc.ch[0]==0)                    /* es tecla especial */
  {  }
else
  {
  if ( sc.ch[0] == CR && i>0 )
    {
    sin_ext='s';
    moveto(180,190);
    strcpy(archivoDAT,palabra);
    for (i=0;palabra[i]!='\0';i++)
    {
    if (palabra[i]=='.')
            {
            sin_ext='n';
            break;
            }
    }
  if (sc.ch[0] == ESC)
    {
    salir_y='n';
    sali_nom='s';
    salir_rtr_h='s';
    limpia_palabra();
    }
  if (sc.ch[0] != ESC && sc.ch[0] != CR && i<29)
    {
    if (sc.ch[0] == ' ');
    if (sc.ch[0] == BKS)
            if (i > 0)
               {
               setcolor(LIGHTGRAY);
               moveto(getx()-8,gety());
               sprintf(buffer,"%c",palabra[i-1]);
               outtext(buffer);
               setcolor(WHITE);
               moveto(getx()-8,gety());
    i--;
    }
    if (sc.ch[0]!=' ' && sc.ch[0]!=BKS)
{
if (sc.ch[0] == '\')
  {
  palabra[i]=sc.ch[0];
  i++;
  }
palabra[i]=sc.ch[0];
sprintf(buffer,"%c",palabra[i]);
setcolor(WHITE);
outtext(buffer);
i++;
}
    }
  }
```

```
        }while(sali_nom=='n');
          if (salir_rtr_h=='n')
        {
        dispositivo();
        fflush(stdin);
        sali_nom='n';
        i=1;
        outtextxy(427,170,"*");
        settextjustify(CENTER_TEXT,CENTER_TEXT);
        ayuda();
        outtextxy(320,455,"     Seleccione el Medio por el cual se    ");
        outtextxy(320,470," Adquirira la Informacign Utilizando las Flechas ");
        do
        {
           sc.c=get_key();
           if(sc.ch[0]==0)                  /* es tecla especial */
                   {
                   if(sc.ch[1]==72)
                     {
                     i=1;
                     outtextxy(427,170,"*");
                     bar(423,202,433,209);
                     }
                   if(sc.ch[1]==80)
                     {
                     i=2;
                     outtextxy(427,207,"*"),
                     bar(423,164,433,174);
                     }
                   }
           else
                   {
                   if ( sc.ch[0] == CR)
                     {
                     if (i==1)                          /*si lalectura es por PS*/
                   {
                   lee_datos_sis();
                   setcolor(WHITE);
                   delay(2500);
                   if(a!=8)
                     {
                     do
                      {
                       ayuda();
                       outtextxy(180,460,"Quieres Borrar la FCRAM Si(s) No(n)?"),
                       salir_rtr_f=getch(),
                      }while(salir_rtr_f!='S' && salir_rtr_f!='s' && salir_rtr_f!='N' && salir_rtr_f!='n');
                     setcolor(BLACK);
                     if (salir_rtr_f=='s' || salir_rtr_f=='S')
                       {
                   limpiate_ps();
                   salir_rtr_f='n';
           }
        }
        limpia_palabra();
      }
  a=0;
  if (i==2)                         /* si lalectura es en el drive*/
{
outtext("teram");
setcolor(WHITE);
```

```
            lee_ierant();
            setbkcolor(BLUE);
           clearviewport();
            rectangle(0,20,639,479);
            settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
            settextjustify(CENTER_TEXT,CENTER_TEXT);
            ayuda();
            linea_com();
            setfillstyle(SOLID_FILL,RED);
            setcolor(RED);
            bar(25,0,140,27);
            setcolor(BLACK);
            outtextxy(82,10,"ARCHIVOS");
            outtextxy(82,20,"DE IC-RAM");
            menu_arch_ic();
            setfillstyle(SOLID_FILL,RED);
            setcolor(RED);
            bar(10,30,160,55);
                        setcolor(BLACK);
                        outtextxy(80,44,"ADQUIRIR INF.");
                        }
                         ordena_archivo_alfabetico();
                         salir_y='n';
                         sali_nom='s';
                         }
                        if (sc.ch[0] == ESC)
                         {
                         salir_y='n';
                         sali_nom='s';
                         limpia_palabra();
                         strcpy(archivoORD,"");
                         }
                        }
            }while(sali_nom=='n');
           }
              sali_nom='n';
              salir_rtr_h='n';
 }
 else
 {
 ayuda();
setcolor(WHITE);
 outtextxy(320,455,"  No Existe Suficiente Memoria Disponible ");
 outtextxy(320,470," Trate de Liberar Memoria Antes de Continuar");
sound(900);
delay(300);
 nosound();
 delay(1700);
 setcolor(BLACK);
 }
}
 opcion_barra_y='n';
 break;
 case 3 :
           if (opcion_barra_y=='n')
             {
             menu_arch_ic();
             setfillstyle(SOLID_FILL,RED);
             bar(10,55,160,80);
             setcolor(BLACK);
             outtextxy(80,69,"CONTROL ARCHIVOS");
```

```
    }
 if(opcion_barra_y=='s')
    {
     if(imagen_leer_ic!=NULL)
    {
  formato();
  ayuda();
   salir_rtr='n';
  settextjustify(CENTER_TEXT,CENTER_TEXT);
   ayuda();
   setcolor(WHITE);
   outtextxy(320,450," Teclee el nombre del archivo donde se encuentra  ");
   outtextxy(320,465," la informacign, si esta deacuerdo con el actual ");
   outtextxy(320,480."         de un enter sino teclee el nuevo       ");
   settextjustify(LEFT_TEXT,CENTER_TEXT);
   moveto(180,205);
   i=0;
   do
   {
setcolor(BLACK);
outtext("_");
setcolor(LIGHTGRAY);
moveto(getx()-8,gety());
sc.c=get_key();
outtext("_");
moveto(getx()-8,gety());
setcolor(WHITE);
 if(sc.ch[0]==0)                    /* es tecla especial */
   {  }
 else
 {
  if ( sc.ch[0] == CR )
   {
   if (i==0)
strcpy(palabra,archivoORD);
   limpia_archivos();
 genera_archivos();
   salir_rtr_h='n';
   limpia_palabra();
   salir_y='n';
   sali_nom='s';
   }
  if (sc.ch[0] == ESC)
   {
   salir_y='n';
   sali_nom='s';
   limpia_palabra();
   salir_rtr_h='s';
   salir_rtr_f='s';
   }
 if (sc.ch[0] != ESC && sc.ch[0] != CR && i<29)
   {
   if (i==0)
            {
            limpia_palabra();
            limpia_archivos();
            bar(175,195,350,210),
            moveto(180,205),
            }
   if (sc.ch[0] == ' '),
   if (sc.ch[0] == BKS)
```

```
                if (i > 0)
                  {
                  setcolor(LIGHTGRAY);
                  moveto(getx()-8,gety());
                  sprintf(buffer,"%c",palabra[(i-1)]);
                  outtext(buffer);
                  setcolor(WHITE);
                  moveto(getx()-8,gety());
                  i--;
                  }
                  if (sc ch[0]!=' ' && sc ch[0]!=BKS)
                          {
                          if (salir_rtr=='n')
                            {
                            settextjustify(CENTER_TEXT,CENTER_TEXT);
                            setcolor(BLACK);
                            formato();
                            moveto(180,205);
                            salir_rtr='s';
                            settextjustify(LEFT_TEXT,CENTER_TEXT);
                            setcolor(WHITE);
                            }
                if (sc.ch[0] == '\r')
                          {
                          palabra[i]=sc.ch[0];
                          i++;
                          }
                           }
                 }
                  }while(sali_nom=='n');
                  ayuda();
                  sali_nom='n';
                  }
                  else
                  {
                  ayuda();
                  outtextxy(320,470," Trate de Liberar Memoria Antes de Continuar");
                  nosound();
                  delay(1700);
                  setcolor(BLACK);
   salir_rtr_h='y';
   }
   if (salir_rtr_h=='n')
            {
            settextjustify(CENTER_TEXT,CENTER_TEXT);
            setcolor(BLACK);
            periodo();
            ayuda();
            setcolor(WHITE);
outtextxy(320,455," Teclee el Periodo que Quiere Verificar Ejemplo");
outtextxy(320,470,"    mes1 05  dia1 09   al mes2 17  dia2 25    ");
settextjustify(LEFT_TEXT,CENTER_TEXT);
settextjustify(LEFT_TEXT,CENTER_TEXT);
fflush(stdin);
do
{
          if(sc ch[0]==0)                   /* es tecla especial */
            {  }
          else
 {
 if ( sc ch[0] == CR && i==8)
```

```
    {
    fflush(stdin);
    setfillstyle(SOLID_FILL,GREEN);
    bar(100,439,539,479);          /* BARRA DE LOS DIAS */
    setfillstyle(SOLID_FILL,LIGHTGRAY);
    fflush(stdin);
    }
  if (sc.ch[0] == ESC)
    {
    salir_y='n';
    sali_nom='s';
    limpia_palabra();
    salir_rtr_f='y';
    }
     sprintf(buffer,"%c%c",palabra[0],palabra[1]);
     mes1=atoi(buffer);
     if ( mes1 != 0 && mes1 <= 12 && isdigit(palabra[0]) && isdigit(palabra[1]))
 {
 moveto(360,165);
 }
    else
 {
 i=0;
 bar(246,156,270,168);
 moveto(250,165);
 }
   }
            if (i==4)
    {
    sprintf(buffer,"%c%c",palabra[2],palabra[3]);
    dia1=atoi(buffer);
    if ( dia1 != 0 && dia1 <= 31 && isdigit(palabra[2]) && isdigit(palabra[3]))
           {
           moveto(250,220);
           }
             else
           {
             mes2=atoi(buffer);
             if ( mes2 != 0 && mes2 <= 12 && isdigit(palabra[4]) && isdigit(palabra[5]))
                    {
                    moveto(360,220);
                    }
           }
  if (i==8)
    {
    sprintf(buffer,"%c%c",palabra[6],palabra[7]);
    dia2=atoi(buffer);
    }
}
 }while(sali_nom=='n');
 sali_nom='n';
putimage(170,100,imagen_leer_ie,COPY_PUT);
 free (imagen_leer_ie);
 }
 else
 {
 ayuda();
 setcolor(WHITE);
 outtextxy(320,455," No Existe Suficiente Memoria Disponible ");
 outtextxy(320,470," Trate de Liberar Memoria Antes de Continuar");
 setcolor(BLACK);
```

```
 }
    if (salir_rtr_f=='n')
 {
 if (imagen_leer_ic!=NULL)
 {
 getimage(170,100,430,330,imagen_leer_ic);
 settextjustify(CENTER_TEXT,CENTER_TEXT);
 dias_festivos();
 ayuda();
 setcolor(WHITE);
 outtextxy(320,455,"   Teclee las Fechas de los Dias Festivos    ");
 outtextxy(320,470,"    Enter si no existen Dais Festivos       ");
 settextjustify(LEFT_TEXT,CENTER_TEXT);
 moveto(250,165);
 i=0;
 settextjustify(LEFT_TEXT,CENTER_TEXT);
 fflush(stdin);
 do
 {
    setcolor(WHITE);
 if(sc.ch[0]==0)                    /* es tecla especial */
  {  }
else
  {
  if ( sc.ch[0] == CR && i==12)
    {
    setfillstyle(SOLID_FILL,LIGHTGRAY);
    fflush(stdin);
    setcolor(BLACK);
sali_nom='s';
limpia_palabra();
salir_y='n';
    }
  if (sc.ch[0] != ESC && sc.ch[0] != CR)
              {
            if (i > 0)
              {
               if (i==2)
               moveto(266,165);
               if (i==4)
               moveto(376,165);
              setcolor(LIGHTGRAY);
              moveto(getx()-8,gety());
              sprintf(buffer,"%c",palabra[(i-1)]);
              outtext(buffer);
              setcolor(WHITE);
              moveto(getx()-8,gety());
              i--;
              }
            if (sc.ch[0]!=' ' && sc.ch[0]!=BKS &&i<=11)
              {
              if (sc.ch[0] == '\')
                {
                palabra[i]=sc.ch[0];
                i++;
                }
              palabra[i]=sc.ch[0];
              sprintf(buffer,"%c",palabra[i]);
              setcolor(WHITE);
              outtext(buffer);
              i++;
```

```
                if (i==2)
            {
            sprintf(buffer,"%c%c",palabra[0],palabra[1]);
            mfest[0]=atoi(buffer);
            if ( mfest[0] != 0 && mfest[0] <= 12 && isdigit(palabra[0]) && isdigit(palabra[1]))
                    {
                    moveto(360,165);
                    }
                  else
                    {
                    i=0;
                    bar(246,156,270,168);
                    moveto(250,165);
                    }
                      }
          if (i==6)
            {
            sprintf(buffer,"%c%c",palabra[4],palabra[5]);
            mfest[1]=atoi(buffer);
            if ( mfest[1] != 0 && mfest[1] <= 12 && isdigit(palabra[4]) && isdigit(palabra[5]))
                    {
                    moveto(360,220);
                    }
            else
                    {
                    i=4;
                    bar(246,210,270,222);
                    moveto(250,220);
                    }
            }
          if (i==8)
            {
            sprintf(buffer,"%c%c",palabra[6],palabra[7]);
            dfest[1]=atoi(buffer);
            if ( dfest[1] != 0 && dfest[1] <= 31 && isdigit(palabra[6]) && isdigit(palabra[7]))
          {
          moveto(250,275);
          }
            else
          {
          moveto(360,220);
          }
            }
          if (i==10)
  {
  sprintf(buffer,"%c%c",palabra[8],palabra[9]);
  mfest[2]=atoi(buffer);
  if ( mfest[2] != 0 && mfest[2] <= 12 && isdigit(palabra[8]) && isdigit(palabra[9]))
{
          moveto(360,275);
          }
          if (i==12)
  {
  sprintf(buffer,"%c%c",palabra[10],palabra[11]);
  dfest[2]=atoi(buffer);
  if ( dfest[2] != 0 && dfest[2] <= 31 && isdigit(palabra[10]) && isdigit(palabra[11]))
                    {
                    i=10;
                    bar(355,265,380,282);
                    moveto(360,275);
                    }
```

```
            }
            }while(sali_nom=='n');
             free (imagen_leer_ic);
            }
             else
             {
             ayuda();
             setcolor(WHITE);
             outtextxy(320,455,"  No Existe Suficiente Memoria Disponible ");
             outtextxy(320,470," Trate de Liberar Memoria Antes de Continuar");
             }
            }
  settextjustify(CENTER_TEXT,CENTER_TEXT);
  setcolor(BLACK);
  }
 opcion_barra_y='n';
  break;

  case 2 :
 if (opcion_barra_y=='n')
  {
  menu_arch_ic();
  outtextxy(82,94,"IMPRIMIR");
  }
 if (opcion_barra_y=='s')
  {
  imsize=imagesize(140,100,460,280);
  imagen_leer_ic=malloc(imsize);
  if (imagen_leer_ic!=NULL)
  {
  getimage(140,100,460,280,imagen_leer_ic);
  imprimir();
  ayuda();
  setcolor(WHITE);
  outtextxy(320,455,"    Teclee el nombre del archivo que     ");
  outtextxy(320,470,"       desea mandar a impresion      ");
              fflush(stdin);
              do
              {
            moveto(getx()-8,gety());
            setcolor(WHITE);
             if(sc.ch[0]==0)                    /* es tecla especial */
              {  }
             else
             {
              if ( sc.ch[0] == CR && i>0)
               {
               fflush(stdin);
               setfillstyle(SOLID_FILL,GREEN);
               bar(100,439,539,479);            /*  BARRA DE LOS DIAS */
               setfillstyle(SOLID_FILL,LIGHTGRAY);
               fflush(stdin);
               setcolor(BLACK);
               imprime_archivo();
               limpia_palabra();
               salir_y='n';
               sali_nom='s';
               }
              if (sc.ch[0] == ESC)
                           {
                           salir_y='n';
```

```
                    sali_nom='s';
                    limpia_palabra();
                    }
          if (sc.ch[0]!=' ' && sc.ch[0]!=BKS )
                    {
                    if (sc.ch[0] == '\')
                      {
                      palabra[i]=sc.ch[0];
                      i++;
                      }
                    palabra[i]=sc.ch[0];
                    sprintf(buffer,"%c",palabra[i]);
                    setcolor(WHITE);
                    outtext(buffer);
                    i++;
                    )
          }
}while(sali_nom=='n');
sali_nom='n';
settextjustify(CENTER_TEXT,CENTER_TEXT);
setcolor(BLACK);
}
else
          {
          ayuda();
          outtextxy(320,455,"  No Existe Suficiente Memoria Disponible ");
          outtextxy(320,470," Trate de Liberar Memoria Antes de Continuar");
          setcolor(BLACK);
          }
          }
          opcion_barra_y='n';
          break;

          case 1 :
        if (opcion_barra_y=='n')
          {
          menu_arch_ic();
          outtextxy(82,120,"MOSTRAR ARCHIVO");
          }
        if (opcion_barra_y=='s')
        {
                if (imagen_menu_mos!=NULL)
                {
                getimage(50,105,200,230,imagen_menu_mos);
                submenu_mostrar();
                setcolor(BLACK);
                outtextxy(120,119,"FALTAS");
                fflush(stdin);
                do
                {
                 sc.c=get_key();
                 if(sc.ch[0]==0)                        /* es tecla especial */
                          {
                          if (sc.ch[0]==ESC)
                          {
                          salir_mos='s';
                          pos_bar_mos=7;
                          limpia_palabra();
                          salir_y='n';
                          sali_nom='s';
                          }
```

```
                }
          switch (pos_bar_mos)
                {
                case 5
                if (opcion_barra_mos=='n')
                  {
                    submenu_mostrar();
                    setfillstyle(SOLID_FILL,RED);
                    setcolor(RED);
                    bar(50,105,200,130);
           setcolor(BLACK);
           outtextxy(120,119,"FALTAS");
           }
        break;

    case 4 :
        if (opcion_barra_mos=='n')
          {
          submenu_mostrar();
          setfillstyle(SOLID_FILL,RED);
          setcolor(RED);
          bar(50,130,200,155);
          setcolor(BLACK);
          outtextxy(120,144,"LISTA COMPLETA");
          }
        break;

case 3 :
        if (opcion_barra_mos=='n')
          {
          submenu_mostrar();
          setfillstyle(SOLID_FILL,RED);
          setcolor(RED);
          bar(50,155,200,180);
          setcolor(BLACK);
          outtextxy(120,169,"PUNTUALIDA");
          }
        break;

case 2 :
        if (opcion_barra_mos=='n')
          {
          submenu_mostrar();
          setfillstyle(SOLID_FILL,RED);
          setcolor(RED);
          bar(50,180,200,205);
          setcolor(BLACK);
          outtextxy(120,194,"RETARDOS");
          }
        if (opcion_barra_mos=='s')
          {
          }
        break;

case 1 :
        if (opcion_barra_mos=='n')
          {
          submenu_mostrar();
          setfillstyle(SOLID_FILL,RED);
          setcolor(RED);
          bar(50,205,200,230);
```

```
                outtextxy(120,219,"TIEMPO EXTRA");
                }
              break;
                }
     opcion_barra_mos='n';
}while(salir_mos=='n');
salir_mos='n';
pos_bar_mos=1;
}
else
{
ayuda();
setcolor(WHITE);
outtextxy(320,455,"  No Existe Suficiente Memoria Disponible ");
outtextxy(320,470," Trate de Liberar Memoria Antes de Continuar");
}
  }
 opcion_barra_y='n';
  break;
    }
  }while(salir_y == 'n');
  pos_bar_y=4;
  opcion_barra_y='n';
  putimage(10,30,imagen_archivos_ic,COPY_PUT);
  free(imagen_archivos_ic);
  salir_y='n';
  }
  else
  {
  ayuda();
  }
}
opcion_barra='n';
salir_prog='n';
ayuda();
 setcolor(WHITE);
outtextxy(320,455," Adquiere y/o Procesa la Informacion de ");
outtextxy(320,470," Asistencias, Faltas, Tiempos Extra ... ");
setcolor(BLACK);
break;

case 2 :
  If ( opcion_barra == 'n')
{
  linea_com();
  setfillstyle(SOLID_FILL,RED);
  setcolor(RED);
  bar(180,0,305,27);
  setcolor(BLACK);
  outtextxy(242,10,"ACTUALIZAR");
  outtextxy(242,20,"HORA Y FECHA");
  opcion_barra='n';
  ayuda();
            }
            if ( opcion_barra == 's' && sin_posibilidad=='y')
              {
            ayuda();
            setcolor(WHITE);
            outtextxy(320,455,"  No Existe Comunicación con el Reloj  ");
            outtextxy(320,470,"     Cheque las Conexiones del Reloj     ");
            outtextxy(320,455,"        Para Hacer Uso de esta Opcion     ");
```

```
outtextxy(420,4[illegible]  Salga del Programa e Intente Nuevamente ");
                                        delay(2000)
                                        setcolor(BLACK)

if ( opcion  barra[illegible]  && sin  posibilidad=='n'

imsize=imagesize(120,100,490,330)
imagen_act_hora=malloc(imsize);
if (imagen_act_hora!=NULL)
{
getimage(120,100,490,330,imagen_act_hora);
/*putimage(120,100,imagen_act_hora,COPY_PUT);*/
act_hora();
settextjustify(LEFT_TEXT,CENTER_TEXT);
i=0;
salir_rtr='n';
setcolor(BLACK);
outtextxy(216,195,"/  /");
outtextxy(216,265,"    ");
moveto(198,195);
flush(stdin);
do
 {
if (i>=0 && i<6)
   {
   ayuda();
   setcolor(WHITE);
   outtextxy(140,450,"Ejemplo     9 de noviembre de 1994       ");
   outtextxy(140,470,"           09 / 11 / 94                  ");
   setcolor(BLACK);
   }
if (i>=6 && i<=11)
    {
          ayuda();
          setcolor(WHITE);
          outtextxy(110,450,"Ejemplos: 5 hrs 35 min 10 seg AM        PM     ");
          outtextxy(110,470,"           05   35   10        17   35   10");
          setcolor(BLACK);
   }
  if (i>11)
  {
          ayuda();
          setcolor(WHITE);
          outtextxy(140,450,"DIAS:     LUNES=LU MARTES=MA MIERCOLES=MI  ");
          outtextxy(110,470,"    JUEVES=JU VIERNES=VI SABADO=SA DOMINGO=DO");
          setcolor(BLACK);
  }
          setcolor(BLACK);
outtext(" ");
setcolor(LIGHTGRAY);
moveto(getx()-8,gety());
sc.c=get_key();
outtext(" ");
moveto(getx()-8,gety());
setcolor(WHITE);
if(sc.ch[0]==0)                 /* es tecla especial */
  {
else
  {
  if ( sc.ch[0] == CR && i==14)
```

```
            fflush(stdin);
            setfillstyle(SOLID_FILL,GREEN);
            bar(100,439,539,479);          /* BARRA DE LOS DIAS */
            setfillstyle(SOLID_FILL,LIGHTGRAY);
            rtr();
            fflush(stdin);
            setcolor(BLACK);
            sali_nom='s';
            limpia_palabra();
            salir_y='n';
            }
 if (sc.ch[0] == ESC)
            {
             salir_y='n';
             sali_nom='s';
             limpia_palabra();
            }
 if (sc.ch[0] != ESC && sc.ch[0] != CR)
            {
            if (sc.ch[0] == ' ');
            if (sc.ch[0] == BKS)
 if (i > 0)
  {
  setcolor(LIGHTGRAY);
  moveto(getx()-8,gety());
  sprintf(buffer,"%c",palabra[(i-1)]),
  outtext(buffer);
  setcolor(WHITE);
  moveto(getx()-8,gety());
  i--;
  }
if (sc.ch[0]!=' ' && sc.ch[0]!=BKS &&i<=13)
  {
        if (sc.ch[0] == '\')
          {
          palabra[i]=sc.ch[0];
          i++;
          }
        palabra[i]=sc.ch[0];
        sprintf(buffer,"%c",palabra[i]);
        setcolor(WHITE);
        outtext(buffer);
        i++;
        if (i==2)
          {
sprintf(buffer,"%c%c",palabra[0],palabra[1]);
          DD=atoi(buffer);
          if ( DD != 0 && DD <= 31)
            {
        moveto(222,195);
            }
          else
            {
            i=0;
            bar(196,187,214,198);
            moveto(198,195);
            }
          }
        if (i==4)
          {
          sprintf(buffer,"%c%c",palabra[2],palabra[3]);
```

```
                    Mm=atoi(buffer);
                    if ( Mm != 0 && Mm <  12)
                      ;
                      moveto(246,195);
                      }
                    else
            {
            i=2;
            bar(220,187,238,198);
            moveto(222,195);
            }
          }
        if (i==6)
          {
          sprintf(buffer,"%c%c",palabra[4],palabra[5]);
          AA=atoi(buffer);
          if ( AA <= 99 && isdigit(palabra[4]) && isdigit(palabra[5]) )
            {
                  moveto(198,265);
            }
          else
            {
            i=4;
            bar(244,187,262,198);
            moveto(246,195);
            }
          }
        if (i==8)
          {
sprintf(buffer,"%c%c",palabra[6],palabra[7]);
            HH=atoi(buffer);
            if ( HH <= 23 && isdigit(palabra[6]) && isdigit(palabra[7]))
        {
        moveto(222,265);
        }
          else
        {
        i=6;
        bar(195,258,216,268);
        moveto(198,265);
        }
          }
           if (i==10)
        {
        sprintf(buffer,"%c%c",palabra[8],palabra[9]);
        MM=atoi(buffer);
        if ( MM <= 59 && isdigit(palabra[8]) && isdigit(palabra[9]))
          {
          moveto(246,265);
          }
         else
          {
          i=8;
          bar(220,258,236,268);
                    moveto(222,265);
                    }
                  }
                }
          }
         }while(sali_nom=='n');
        sali_nom='n';
```

```
            putimage(120,100,imagen_act_hora,COPY_PUT);
            free(imagen_act_hora);
            settextjustify(CENTER_TEXT,CENTER_TEXT);
            setcolor(BLACK);
            }
            else
            {
            ayuda();
            setcolor(WHITE);
            outtextxy(320,455,"  No Existe Suficiente Memoria Disponible ");
            outtextxy(320,470," Trate de Liberar Memoria Antes de Continuar");
            }
ayuda();
setcolor(WHITE);
outtextxy(320,455," Permite ajustar la Hora, Fecha del Reloj ");
outtextxy(320,470,"          Checador desde la PC          ");
setcolor(BLACK);
             }
           salir_rtr_d='n';
           salir_rtr_f='n';
           salir_rtr_h='n';
           opcion_barra='n';
           salir_prog='n';
           break;

           case 3 :
           if ( opcion_barra == 'n')
           {
             linea_com();
             setfillstyle(SOLID_FILL,RED);
             setcolor(RED);
             bar(340,0,465,27);
             setcolor(BLACK);
             outtextxy(402,10,"VELOCIDAD DE");
             outtextxy(402,20,"TRANSMISION");
             salir_prog='n';
             opcion_barra ='n';
             ayuda();
             setcolor(WHITE);
             outtextxy(320,455," Permite cambiar la Velocidad de Comunicacion ");
             outtextxy(320,470,"   Entre la PC y el reloj Checador     ");
             setcolor(BLACK);
           }
           if( opcion_barra == 's' && sin_posibilidad=='y')
             {
           ayuda();
           setcolor(WHITE);
           outtextxy(320,455," No Existe Comunicacion con el Reloj  ");
           outtextxy(320,470,"     Cheque las Conexiones del Reloj     ");
           delay(2000);
           setcolor(BLACK);
             }
           if( opcion_barra == 's'&& sin_posibilidad=='n')
           {
           imsize=imagesize(200,100,400,330);
           imagen_vel_trans=malloc(imsize);
           if (imagen_vel_trans!=NULL)
           {
           vel_transf();
           ayuda();
           setcolor(WHITE);
```

```
outtextxy(320,462," Teclee la Velocidad de Comunicaci¢n Deseada ");
setcolor(BLACK);
settextjustify(LEFT_TEXT,CENTER_TEXT);
switch(velocidad_puerto)
  {
  case 12:
 outtextxy(265,170,"*");
 break;
 case 24:
 outtextxy(265,200,"*");
 break;
  case 48:
 outtextxy(265,230,"*");
 break;
  case 96:
 outtextxy(265,260,"*");
 break;
  }
moveto(320,227);
i=0;
salir_rtr='n';
while(salir_rtr=='n')
    {
 setcolor(BLACK);
 outtext("_");
 setcolor(LIGHTGRAY);
 moveto(getx()-8,gety());
 sc.c=get_key();
 outtext("_");
 moveto(getx()-8,gety());
 setcolor(WHITE);
 if(sc.ch[0]==0)                    /* es tecla especial */
              {  }
               else
                 {
              if(sc.ch[0[==ESC)
                {
                salir_rtr='y';
                limpia_palabra();
                }
              if(sc.ch[0]==CR)
                {
                 velocidad_puerto=atoi(palabra);
                 switch(velocidad_puerto)
                   {
                    case 9600:
                selvel=1;
                manda_cambio_vel();
                velocidad_puerto=12;
                ini_pto_ser();
                fin_cambio_vel();
                salir_rtr='y';
                break;
                   case 4800:
                selvel=2;
                velocidad_puerto=24;
                manda_cambio_vel();
                ini_pto_ser();
                fin_cambio_vel();
                salir_rtr='y';
              break;
```

```
                    case 2400:
                  selvel=3;
                  manda_cambio_vel();
                  velocidad_puerto=48;
                  ini_pto_ser();
                  fin_cambio_vel();
                  salir_rtr='y';
                 break;
                    case 1200:
                  selvel=4;
                  manda_cambio_vel();
                  velocidad_puerto=96;
                  ini_pto_ser();
                  fin_cambio_vel();
                  salir_rtr='y';
                 break;
                   default:
                 setfillstyle(SOLID_FILL,LIGHTGRAY);
                 bar(310,220,360,230);
                 moveto(320,227);
                 i=0;
                 salir_rtr='n';
                 break;
                   }
                    limpia_palabra();
                   }
                 if (sc.ch[0] != ESC && sc.ch[0] != CR)
                   {
                   if (sc.ch[0] == ' ');
                   if (sc.ch[0] == BKS)
                     if (i > 0)
                 {
                  setcolor(LIGHTGRAY);
                  i--;
                  }
                   if (sc.ch[0]!=' ' && sc.ch[0]!=BKS && i<=3)
                     {
                     palabra[i]=sc.ch[0];
                     sprintf(buffer,"%c",palabra[i]);
                     setcolor(WHITE);
                     outtext(buffer);
                     i++;
                     }
                   }
          }
}
      putimage(200,100,imagen_vel_trans,COPY_PUT);
      free (imagen_vel_trans);
      setcolor(BLACK);
      salir_rtr='n';
      settextjustify(CENTER_TEXT,CENTER_TEXT);
      }
      else
      {
      ayuda();
      setcolor(WHITE);
      outtextxy(320,455," No Existe Suficiente Memoria Disponible ");
      outtextxy(320,470," Trate de Liberar Memoria Antes de Continuar");
      }
      ayuda();
      setcolor(WHITE);
```

```
        outtextxy(320,455," Permite cambiar la Velocidad de Comunicaci¢n ");
        outtextxy(320,470,"   Entre la PC y el reloj Checador    ");
                setcolor(BLACK);
              }
        opcion_barra='n';
        salir_prog='n';
        brenk;
        case 4 :
        if ( opcion_barra == 'n')
        {
          linea_com();
          setfillstyle(SOLID_FILL,RED);
          setcolor(RED);
          bar(505,0,596,27);
          setcolor(BLACK);
          outtextxy(548,15,"SALIR");
          ayuda();
          setcolor(WHITE);
          outtextxy(320,462," Termina la Secci¢n y Sale al Sistema Operativo ");
          setcolor(BLACK);
        }
        if ( opcion_barra == 's')
        {
                imsize=imagesize(190,100,430,280);
                imagen_salida=malloc(imsize);
                getimage(190,100,430,280,imagen_salida);
                /*putimage(190,100,imagen_salida,COPY_PUT);*/
                salida();
                sc.c=get_key();
                if(sc.ch[0]==0)                 /* es tecla especial */
                  {  }
                else
                  {
                if( sc.ch[0] == CR)
                   {
                   putimage(190,100,imagen_salida,COPY_PUT);
                   free (imagen_salida);
                   salir_prog='s';
                   }
                if (sc.ch[0] == ESC)
                   {
                   salir_prog='n';
                   putimage(190,100,imagen_salida,COPY_PUT);
                   free (imagen_salida);
                   }
          }
        }
        opcion_barra='n';
        break;
}
  }
 while(salir_prog =='n');
closegraph();
printf("%s",palabra);
}
```