ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO INTELIGENTE DE MONITOREO PARA UN PARQUEADERO, EN BASE A MICROCONTROLADORES (PICs)

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

JUAN LUIS HEREDIA VELASTEGUI

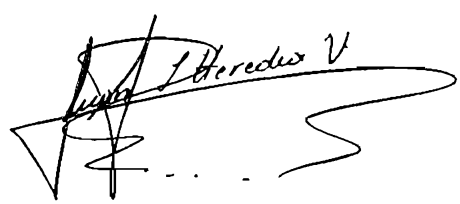
DIRECTOR: ING. FERNANDO FLORES

Quito, Diciembre 2005

DECLARACIÓN

Yo, Juan Luis Heredia Velasteguí, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.



Juan Luis Heredia Velasteguí

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Juan Luis Heredia Velasteguí, bajo mi supervisión.

Ing. Fernando Flores
DIRECTOR DE PROYECO

AGRADECIMIENTO

Agradezco a Dios por darme la oportunidad de alcanzar uno de mis sueños, y poder entregarlo a las personas que amo.

A mis padres y mi familia por haberme dado la oportunidad de adquirir el conocimiento para cada día ser mejor ser humano y sobre todo por ser las personas que me ayudaron a iniciar este pequeño sueño.

A la persona que da luz a mi vida, mi Negrita, por ser la luz que me ha ayudado a no desfallecer y poder terminar uno de mis proyectos.

A mis jefes, por darme las facilidades para cumplir este proyecto; y a mis compañeros y amigos de trabajo por apoyarme en cada paso que di, y sobre todo por estar pendientes del avance.

A mis amigos del CETTIETRI, que siempre me han ayudado. A los profesores que contribuyeron con su conocimiento, y que ayudaron para culminar mi carrera.

Mi inmensa gratitud a todas las personas que colaboraron para que este trabajo se cumpla, gracias por siempre.

"Los buenos deseos no bastan para realizar un mundo mejor, es necesario actuar "

Juan Luis Heredia

DEDICATORIA

A la mujer que inspira mi vida, mi mejor Amiga Marcela Basantes, quien me ha apoyado durante todo este tiempo, sobre todo en los momentos difíciles, y se convirtió en la voz que no me dejó desfallecer y a quien puedo decir cumplí. A las personas que me dieron la oportunidad de nacer, mi Madre Zoila Velasteguí y mi Padre Héctor Heredia, por enseñarme a ser perseverante. Y a mi familia porque gracias a su apoyo este y otros sueños se han hecho realidad.

A todas las personas que tienen sueños de un mejor país, de un mejor planeta y porque no, de un mejor Universo, para que siempre pongamos en hechos lo que podemos crear en nuestra mente.

"Nosotros tenemos que ser el cambio que queremos ver en el mundo"

CONTENIDO

RESUMEN Y PRESENTACION.

El prototipo es un dispositivo que busca facilitar la administración de lugares cerrados o abiertos como son los parqueaderos, para de esta forma optimizar tiempo, recursos y dinero; todo esto se logra teniendo un control de las plazas que se dispone, y mediante sensores como los loops magnéticos poder detectar cuales se ocupan.

El sistema busca crear modularidad entre cada una de sus partes, para proporcionar diferentes medios de control comos son señales luminosas y sonoras, en cada uno de los módulos.

El Prinmop constituye el punto departida para proyectos de mayor alcance, que involucren diferentes tipos de sensores, acoplándose a la realidad de cada área a ser monitoreada, un número mayor de dispositivos que ayuden a que el sistema sea manejado de una forma más automática.

La utilización de herramientas como el compilador Pic Basic Pro, permiten que los proyectos sean mas fáciles de entender y analizar, para de esta manera los programadores puedan ir implementando mejoras en el sistema y lograr cada vez una mayor integración de otro tipo de elementos.

La utilización de los loops magnéticos para realizar el sensado de los vehículos, le proporciona al prototipo la característica de poder ser instalado en cualquier medio, sea este un espacio abierto o cerrado; y la característica de ser instalados bajo el piso proporciona al sistema discreción, sobre todo si estos dispositivos son utilizados como medios de seguridad.

Los Microcontroladores PIC han revolucionado el diseño electrónico, no solo por su manejo de periféricos con una gran facilidad, sino también por su reducido tamaño y su gran versatilidad para la programación, es así que se han creado herramientas que facilitan aun más el manejo de estos dispositivos.

La unión del hardware y el software permiten crear herramientas que nos ayudan a manejar determinados sistemas de una forma mas adecuada, es así que los datos proporcionados por los MODEMAG viajan hasta el MINT – PC y este los entrega aun computador para que sean procesados mediante software y le proporcionen al administrador del sistema una forma mas clara del estado del plazas.

El módulo de detección, toma los datos de los sensores magnéticos colocados bajo cada una de las plazas y los envía hacia el módulo de interfaz, para que este encargue de codificarlo según sea una plaza libre u ocupada y envía la información hacia la computadora y mediante el software se muestre los resultados al usuario administrador. Además posee una combinación de switch y compuertas lógicas para poder realizar reservas manuales de las plazas y también sensar las plazas ocupadas de forma automática mediante los loops magnéticos o de forma manual.

El módulo de control consta de un teclado y una pantalla de cristal liquido, para poder enviar mensajes de alarma al sistema, y también poder recibir los mensajes provenientes de los diferentes módulos. Para darle seguridad al sistema, el módulo consta de una clave, que puede ser configurada por el usuario, además desde este módulo se puede enviar mensajes pregrabados hacia el panel mímico, y mediante pulsadores se pude activar las alarmas.

El módulo que controla el panel informativo, maneja una pantalla de cristal líquido, matrices de leds, leds, dispositivos I2C, y la comunicación con el módulo de interfaz. Este módulo es el encargado de mostrar diferentes tipos de información al usuario

del parqueadero, como por ejemplo mensajes pregrabados, número de plazas libres y el estado de las plazas mediante leds.

El módulo de interfaz con la computadora se encarga de recibir los datos de los diferentes módulos y enviar la información que los otros módulos requieran

En la computadora se carga un software diseñado en lengua Visual Basic que toma los datos del hardware y los muestra al usuario en la pantalla de visualización de plazas, también permite obtener estadísticas del sistema de cada una de las plazas, fecha y hora de ocupación y desocupación, y el monto cobrado por el servicio; además posee una pantalla donde se ingresa cada una de las tarjetas proporcionadas a los usuarios,

CAPITULO 1

1. CONCEPTOS GENERALES

1.1 MICROCONTROLADORES PIC

1.1.1 ¿QUÉ ES UN MICROCONTROLADOR?

Un microcontrolador es un circuito integrado que posee todos los componentes de un computador como son: CPU, memoria RAM, EEPROM, y circuitos de entrada y salida.

Al microcontrolador se lo debe programar para que realice desde actividades muy sencillas como el parpadeo de un led, hasta actividades más complejas como la automatización de una fábrica. Un microcontrolador puede remplazar a varios *circuitos* lógicos, es decir puede realizar las actividades de compuertas AND, OR, NOT, NAND, conversores A/D, D/A, temporizadores, decodificadores, contadores, etc., lo que permite realizar diseños con placas de reducido tamaño y pocos elementos.

Luego de ser programado, en su memoria reside solo el programa que controla una aplicación determinada; sus líneas de entrada y salida soportan directamente varios actuadores y sensores del dispositivo a controlar.

Los microcontroladores PIC (Peripherical Interface Controller), en su mayoría son fabricados por MICROCHIP Technology INC, la cual mantiene el liderazgo en su fabricación frente a sus demás competidores debido a la gran variedad, gran velocidad, bajo consumo de potencia, bajos costos, y la gran disponibilidad de herramientas para su programación.

Los microcontroladores en la actualidad son muy usados en varias áreas, como informática (periféricos del computador), en electrodomésticos de la línea blanca (lavadoras, hornos, lavavajillas, etc.), así también en la línea marrón (televisores, videos, aparatos musicales, etc.). Así también se usan en sistemas de vigilancia, supervisión y alarmas, en sistemas de comunicación, modernos teléfonos, así también son usados en la electromedicina e instrumentación, todo esto lleva a pensar la gran área de cobertura que están alcanzando los microcontroladores.

Los PIC son una familia de microcontroladores, cuya arquitectura, capacidades, juego de instrucciones y especialmente su bajo costo lo hacen muy útil en pequeñas aplicaciones, así como parte de otras aplicaciones de mayor envergadura, sustituyendo a una gran cantidad de circuitos lógicos convencionales.

1.1.2. ¿PORQUE TRABAJAR CON PICs?

En los últimos años, se han convertido en los preferidos de los diseñadores, esto es por el precio, la velocidad, la información, la facilidad de uso, las herramientas de apoyo, no existe una razón específica de su popularidad, lo que si se sabe es que tiene una imagen bien ganada de sencillez y utilidad.

Los PIC se han popularizado entre los profesionales, debido a las siguientes características:

Tienen un juego de instrucciones reducido, 35 en la gama media, lo que los vuelve de fácil manejo. Existe una gran cantidad de información, es fácil de obtener y .económica. Frente a sus competidores posee un precio comparativamente inferior.
Sus parámetros de funcionamiento son buenos respecto a velocidad, consumo, tamaño, alimentación, código compacto, etc.

Poseen herramientas de desarrollo fáciles y baratas. Existe una gran variedad de herramientas de hardware, que permiten grabar, depurar, borrar y comprobar el comportamiento de los PIC. La gran variedad de modelos de PIC, permite elegir el que mejor responde a los requerimientos de la aplicación.
Una de las principales razones del éxito de los PIC, se basa en su utilización. Cuando se aprende a manejar uno de ellos, conociendo su arquitectura y su juego de instrucciones, es muy fácil pasar a cualquier otro componente.

1.1.3 CARACTERISTICAS PRINCIPALES

1.1.3.1 Arquitectura Interna

El procesador al usar Arquitectura Harvard, permite obtener elevados rendimientos en el procesamiento de las instrucciones, debido a que la memoria de datos y la memoria de instrucciones son independientes y cada una dispone de su propio sistema de buses de acceso, lo que permite acceder a ambas simultáneamente.

El microcontrolador posee todos los elementos de un computador, pero con características fijas que no se pueden alterar.

Las partes principales de un microcontrolador son:

- Procesador
- Memoria no volátil para contener el programa
- Memoria de lectura y escritura para guardar los datos
- Líneas e E/S para los controladores de periféricos
- Comunicación paralelo
- Comunicación serial
- Diversas puertas de comunicación (bus I2C, USB, etc.)
- Recurso auxiliares:

- Circuito de reloj
- Temporizadores
- Perro Guardián (watchdog)
- Conversor AD y DA
- Comparadores analógicos
- Protección ante fallos de la alimentación
- Estado de reposo o de bajo consumo

1.1.3.2 Segmentación

Usa segmentación (pipe – line) que permite la ejecución de una instrucción mientras se busca la siguiente (salvo en los de familias altas). La ejecución de cada instrucción tarda 8 periodos de reloj, formando 2 ciclos (búsqueda y ejecución) de 4 periodos cada uno, ejecutándose paralelamente al usar el primer ciclo la memoria de programa y el segundo la de datos.

1.1.3.3 Formato de las Instrucciones

El formato de todas las instrucciones es de la misma longitud. En la gama baja todas las instrucciones tienen una longitud de 12 bits. En la gama media tienen una longitud de 14 bits y 16 bits los de la gama alta. Esta característica es muy ventajosa en la optimización de la memoria de instrucciones y facilita enormemente la construcción de ensambladores y compiladores.

Al usar instrucciones de igual longitud, se permite optimizar el uso de la memoria de programa (todas las instrucciones ocupan 12, 14 ó 16 bits según el microcontrolador, lo que permite incluir en una sola palabra el código de operación y operándoos; además este tamaño es la anchura del bus de datos de la memoria de programa.

1.1.3.4 Juego de Instrucciones.

El procesador responde a un juego de instrucciones reducido, denominada arquitectura RISC (Computadores de Juego de Instrucciones Reducido), cuya característica principal es tener un juego de instrucciones de máquina, pequeñas y simples, de forma que la mayoría de instrucciones se ejecutan en un ciclo de reloj.

La arquitectura Harvard, arquitectura RISC y la segmentación, permiten al PIC tener un alto rendimiento y elevada velocidad. Los modelos de la gama baja disponen de un repertorio de 33 instrucciones, 35 los de la gama media y casi 60 los de la alta.

1.1.3.5 Ortogonalidad de las Instrucciones.

Cualquier instrucción puede manejar cualquier elemento de la arquitectura como fuente o como destino. Instrucciones ortogonales, es decir, todas pueden operar con todos los operandos.

1.1.3.6 Arquitectura basada en un Banco de Registros.

Arquitectura basada en bancos de registros, es decir, todos los elementos del microcontrolador (memoria de datos, puertos de E/S, temporizadores, etc.) físicamente se encuentran implementados como registros

1.1.3.7 Herramientas de soporte potentes y económicas.

Se encuentran a disposición de los usuarios numerosas herramientas, para el desarrollo de hardware y software, una de las empresas más grandes en este campo es Microchip. Existe una gran cantidad de programadores, simuladores de software, emuladores en tiempo real, Ensambladores, Compiladores C, Intérpretes y Compiladores BASIC, etc.

La arquitectura Harvard y la técnica de segmentación son los principales recursos en los que se apoya el elevado rendimiento que caracteriza estos dispositivos programables, mejorando dos características esenciales:

- ✓ Velocidad de ejecución.
- ✓ Eficiencia en la compactación del código.

1.1.4 FAMILIA DE LOS PICs

Una de las labores más importantes del ingeniero de diseño es la elección del microcontrolador que mejor satisfaga las necesidades del proyecto con el mínimo presupuesto.

Para aplicaciones de baja envergadura se utilizan pocos recursos, en cambio las aplicaciones complejas requieren numerosos y potentes recursos. Basados en esta filosofía, Microchip ha diseñado diversos modelos de microcontroladores orientados, a cubrir de forma optima las necesidades de cada proyecto. Así, hay disponibles microcontroladores sencillos y baratos para atender las aplicaciones simples y otros complejos y más costosos para las de mayor envergadura.

En la mayor parte de bibliografía tan solo se consideran tres familias de microcontroladores, despreciando a la denominada gama baja, puesto que se consideran una subfamilia formada por componentes de las otras gamas. En este documento se prefiere mencionarlos debido a que los PIC de la gama baja son muy apreciados en las aplicaciones de control de personal, en sistemas de seguridad y en dispositivos de bajo consumo que gestionan receptores y transmisores de señal, además su pequeño tamaño los hace ideales en muchos proyectos donde esta cualidad es fundamental.

1.1.4.1 Gama Baja

Tienen una memoria de programa (ROM o EPROM) de 12 bits y de 512, 1024 ó 2048 palabras y una memoria de datos de 25, 72 ó 73 bytes. Trabajan hasta 20 MHz y disponen de 12 ó 20 líneas de E/S de alta corriente, un temporizador y de 33 instrucciones. Además, como el resto de los Pic están provistos de perro guardián, auto inicialización (POR o Power on Reset), modo de bajo consumo (Sep), reloj interno mediante cristal o red RC y protección contra lectura de código.

En la siguiente figura se muestra el diagrama de conexiones de esta gama.

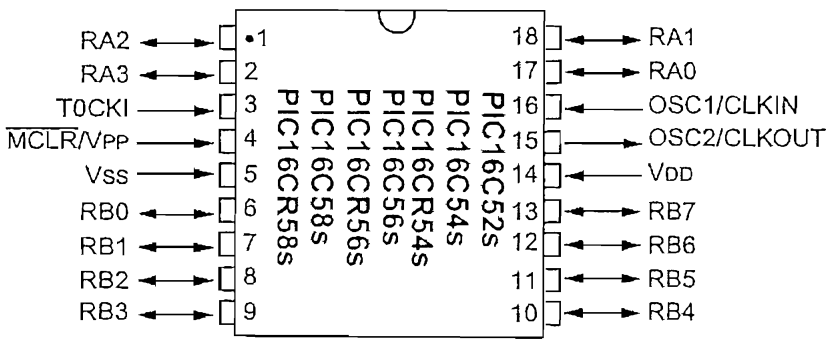


Figura 1.1

Diagrama de pines de los PIC de la gama baja que responden a la nomenclatura PIC16C54/56.

Tanto los Pic de la gama baja como los de la familia PIC 16/17, presentan los siguientes recursos:

1. Sistema POR ("Power on Reset").

Todos los PIC tienen la facultad de generar una autoreinicialización o auto reset al conectarles la alimentación.

2. Perro Guardián ("Watchdog o WDT ").

Para evitar que el sistema se quede colgado, el sistema posee un temporizador que produce un reseteo automático, si no se recarga luego de que pase un tiempo prefijado; esto evita que cuando el programa ingrese a alguna parte y no pueda salir se quede indefinidamente en ese ciclo.

3. Código de Protección.

Esta característica permite proteger el programa cuando se realiza la grabación, y así evitar que sea leído. También disponen los PIC de posiciones reservadas para registrar números de serie, códigos de identificación, prueba, etc.

4. Líneas de E/S de alta corriente.

Las líneas de E/S de los PIC pueden proporcionar o absorber una corriente de salida comprendida entre 20 y 25 mA, capaz de excitar directamente ciertos periféricos.

5. Modo de reposo (Bajo Consumo o "Sleep").

Ejecutando esta instrucción (Sleep), tanto el CPU como el oscilador principal se detienen, y así se reduce notablemente el consumo.

Así también esta familia presenta las siguientes restricciones:

La pila o "stack" sólo dispone de dos niveles lo que supone no poder encadenar más de dos subrutinas. Los microcontroladores de la gama baja no admiten interrupciones.

La siguiente tabla muestra algunas características de esta gama:

MODELO	MEMORIA PROGRAMA (X12 BITS) EPROM ROM	MEMORIA DATOS (bytes)	FREC. MAXIMA	LINEAS E/S	TEMPORIZADORES	PINES
PIC16C52	384	25	4 MHz	4	TMR0 + WDT	18
PIC16C54	512	25	20 MHz	12	TMR0 + WDT	18
PIC16C54A	512	25	20 MHz	12	TMR0 + WDT	18
PIC16CR54A	512	25	20 MHz	12	TMR0 + WDT	18
PIC16C55	512	24	20 MHz	20	TMR0 + WDT	28
PIC16C56	1K	25	20 MHz	12	TMR0 + WDT	18
PIC16C57	2K	72	20 MHz	20	TMR0 + WDT	28
PIC16CR57B	2K	72	20 MHz	20	TMR0 + WDT	28
PIC16C58A	2K	73	20 MHz	12	TMR0 + WDT	18
PIC16CR58A	2K	73	20 MHz	12	TMR0 + WDT	18

Tabla 1.1

1.1.4.2 Gama Media

En esta gama existen encapsulados desde 18 pines hasta 68 pines y es la gama mas variada y completa de los PICs. En esta familia se encuentra el rendidor PIC 16x84 y sus variantes.

Esta gama tiene nuevas prestaciones a las que poseían los de la gama baja, lo que los convierte en elementos más adecuados para sistemas más complejos. Admiten interrupciones, poseen comparadores de magnitudes analógicas, convertidores A/D, puertos serie y varios temporizadores, y otras características según el modelo.

En la siguiente figura se muestra el diagrama de conexionado de uno de estos PIC.

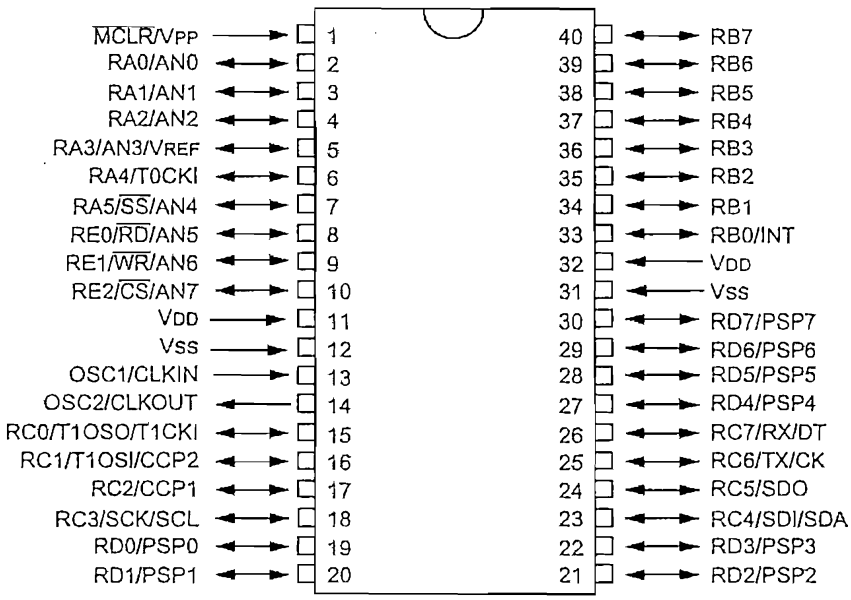


Figura 1.2

Diagrama de pines del PIC16C74, uno de los modelos más representativos de la gama media

El ancho de la memoria de programa es de 14 bits, pudiendo ser ROM, EPROM o EEPROM. El repertorio de instrucciones es de 35, de 14 bits cada una y compatibles con la gama baja. Sus distintos modelos contienen todos los recursos que se precisan en las aplicaciones de los microcontroladores de 8 bits. También disponen de interrupciones y una pila de 8 niveles que permite el anidamiento de subrutinas.

En la siguiente tabla se muestra las principales características de algunos modelos de esta familia.

MODELO	MEM. PROG	MEMORIA DATOS		REG. ESPEC	TEMPOR	INTER	E/S	RANGO VOLT.	PINES
		RAM	EEPROM						
PIC16C84	1Kx14 EEPROM	36	64	11	TMR0+ WDT	4	13	2-6	18
PIC16F84	1Kx14 FLASH	68	64	11	TMR0+ WDT	4	13	2-6	18
PIC16F83	512x14 FLASH	36	64	11	TMR0+ WDT	4	13	2-6	18
PIC16CR84	1Kx14 ROM	68	64	11	TMR0+ WDT	4	13	2-6	18
PIC16CR83	512x14 ROM	36	64	11	TMR0+ WDT	4	13	2-6	18

Tabal 1.2

Características relevantes de los modelos PIC 16x8x de la gama media

Encuadrado en la gama media también se halla la versión PIC14C000, que soporta el diseño de controladores inteligentes para cargadores de baterías, pilas pequeñas, fuentes de alimentación ininterrumpibles y cualquier sistema de adquisición y procesamiento de señales que requiera gestión de la energía de alimentación. Los PIC 14C000 admiten cualquier tecnología de las baterías como Li-Ion, NiMH, NiCd, Ph y Zinc [1].

El temporizador TMR1 posee un circuito oscilador que puede trabajar sincrónicamente, y que puede incrementarse así el microcontrolador entre en el

modo de reposo ("sep"), lo que posibilita la implementación de un reloj en tiempo real.

Las líneas de E/S presentan una carga "pull-up" activada por software.

1.1.4.3. Gama Alta

El repertorio de instrucciones aumenta a 55 ó 58, según el modelo, las instrucciones son de 16 bits, y disponen de un sistema de gestión de interrupciones sectorizadas y muy potente.

También poseen varios controladores de periféricos, puertas de comunicación serie y paralelo con elementos externos, un multiplicador hardware de gran velocidad y mayores capacidades de memoria, que alcanza las 8K palabras en la memoria de instrucciones y 454 bytes en la memoria de datos.

Una de las características mas destacadas de esta gama es la arquitectura abierta que posee, esto quiere decir la posibilidad que tienen para trabajar con elementos externos. Para este fin, los pines sacan al exterior las líneas de los buses de datos, direcciones y control, a las que se conectan memorias o controladores de periféricos.

Esta característica obliga a esta gama tener una gran cantidad de pines comprendido entre 40 y 44. Esta familia se usa para aplicaciones muy especiales con grandes requerimientos.

PINES	41/44	41/44	41/44	64/65	64/65
RE					

1.1.4.4 Gama Pequeña.

Con solo 8 pines, disponen de 6 líneas de E/S y una frecuencia máxima de 4 MHz. Disponen de una red RC interna para el oscilador aprovechando al máximo los pines disponibles. Algunos modelos incluyen conversores A/D y una memoria EEPROM de datos. Esta gama es posterior a las anteriores para cubrir unas necesidades muy concretas, como es sustituir a un gran volumen de lógica cableada o implementar por hardware partes de sistemas microprocesadores mas complejos.
Su alimentación es de corriente continua y esta entre 2.5 V y 5.5 V, y su consumo es de menos de 2 mA, cuando trabajan a 5 V y 4 MHz. El formato de sus instrucciones puede ser de 12 o de 14 bits, y el paquete de instrucciones puede ser de 33 ó 35 instrucciones respectivamente.

En la siguiente figura se muestra el diagrama de conexiones de esta gama.

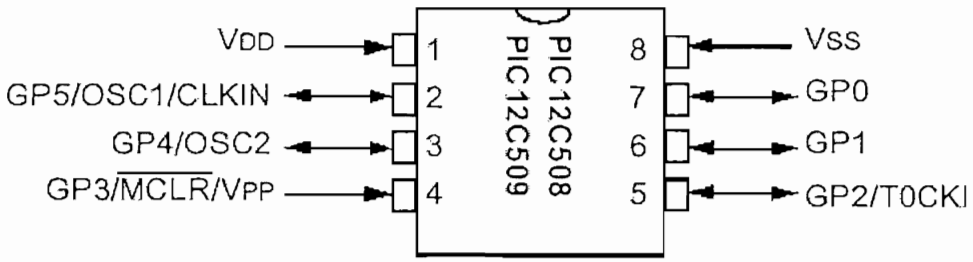


Figura 1.3

Diagrama de conexiones de los PIC12Cxx de la gama baja.

En la siguiente tabla se muestra las principales características de los PICs de esta subfamilia.

MODELO	MEMORIA PROGR	MEMORIA DATOS	FREC. MAXIMA	LINEAS E/S	ADC 8 BITS	TEMPORIZ	PINES
PIC12F508	512x12	25x8	4 MHz	6		TMR0+WDT	8
PIC12F509	1024x12	41x8	4 MHz	6		TMR0+WDT	8
PIC12F670	512x14	80x8	4 MHz	6		TMR0+WDT	8
PIC12F671	1024x14	128x8	4 MHz	6	2	TMR0+WDT	8
PIC12F672	2048x14	128x8	4 MHz	6	4	TMR0+WDT	8
PIC12F680	512X12 FLASH	80x8 16x8 EEPROM	4 MHz	6	4	TMR0+WDT	8
PIC12F681	1024x14 FLASH	80x8 16x8 EEPROM	4 MHz	6		TMR0+WDT	8

Tabla 1.4

Características de los modelos PIC12C (F) x de la gama baja.

1.1.5 VENTAJAS.

Las instrucciones de 12, 14 ó 16 bits de ancho son compatibles con los dispositivos siguientes posteriores, para maximizar la eficiencia en el procesamiento y el funcionamiento. Las instrucciones y los datos son transferidos en buses separados permitiendo el procesamiento simultáneo.

Conductos de dos estados que permiten que una instrucción se está ejecutando mientras la siguiente se esta cargando. Instrucciones de palabras de un ancho sencillo, incrementando la eficiencia del software y reduciendo los requerimientos de memoria para el programa. Instrucciones sencillas, rápidas y fáciles de aprender.

Compatibilidad entre los dispositivos, lo que permite realizar el código para un dispositivo y poder utilizarlo para otro mejorado (superior). Para resolver aplicaciones sencillas se precisan pocos recursos; en cambio, las aplicaciones grandes requieren numerosos y potentes. Siguiendo esta filosofía, Microchip construye diversos modelos de microcontroladores orientados a cubrir las necesidades de cada proyecto. Así existen disponibles microcontroladores sencillos y baratos para atender las aplicaciones simples y otra complejas y más costosos para las de mucha envergadura.

1.2. COMUNICACIÓN SERIAL SINCRONICA I2C

1.2.1. INTRODUCCION.

I2C es un bus de dos alambres, fue inventado por Philips Semiconductores, para la comunicación entre ICs en 1980, se ha convertido en el bus serial Standard, que se ha implementado en un gran número de ICs, y con licencias otorgadas a más de 50 compañías con un total de 1000 dispositivos compatibles I2C.

En sus inicios fue especificado para 100 Kbps, se ha extendido para soportar velocidades de hasta 3.4 Mbps, y en modo **HIGH** – Speed (Hs-mode) ofrece la función. de desplazamiento de nivel de voltaje, que es una solución ideal para los sistemas de tecnología mezclada, donde las altas velocidades y la variedad de voltajes (5 V, 3 V o menor) son comúnmente usados.

El modo HS es compatible con todos los sistemas existentes del bus I2C, incluyendo el estándar original (S-mode) y el modo Fast (F-mode). Diferentes sistemas de velocidad pueden ser mezclados fácilmente, con un dispositivo maestro en modo Hs especialmente desarrollado., la conexión en paralelo es usada para conectar las partes mas lentas del sistema.

El bus Inter.-IC o I2C, se ha establecido como la solución mundial para aplicaciones integradas. Es usado en una gran cantidad de microcontroladores y aplicaciones de telecomunicaciones como en control, diagnostico y administración de potencia. Su simplicidad se hay mantenido incluso con las mejoras a la especificación original.

1.2.2 ESPECIFICACIONES.

Está orientado a las aplicaciones de 8-bits controladas por un microprocesador y estos son básicamente los criterios que se deben establecer:

Un sistema consiste en al menos un microcontrolador y varios sistemas periféricos como memorias o circuitos diversos.

El costo de conexión entre los varios dispositivos dentro del sistema debe ser el mínimo.

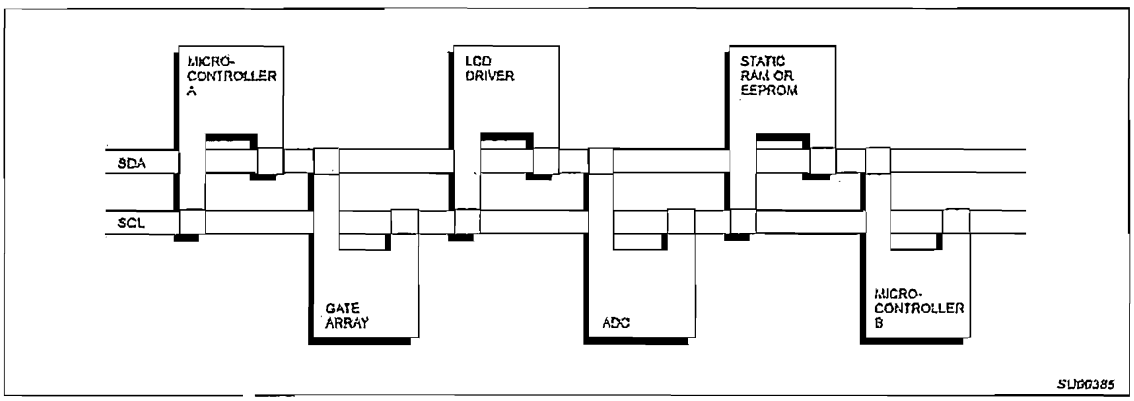
El sistema que usa este Bus no requiere una alta tasa de transferencia de datos.
La total eficiencia del sistema depende de la correcta selección de la naturaleza de los dispositivos y de la interconexión de la estructura del bus.

1.2.3 EL CONCEPTO DEL BUS I2C.

Está compuesto por dos hilos físicos, uno de datos llamado SDA y otro de reloj SCL, los cuales transportan la información entre los diversos dispositivos conectados al bus.

Cada dispositivo (microcontrolador, LCD, memoria o teclado) es reconocido por una única dirección, y cualquiera de estos dispositivos puede actuar como emisor o receptor de datos, dependiendo de la función del dispositivo. Un display es solo un receptor de datos, mientras que una memoria recibe y transmite datos.

Es así que en función de que envíe o reciba datos, se debe considerar los dispositivos como Maestros (master) o esclavos (Slave). Ver la siguiente figura.



Ejemplo del bus I2C usando dos microcontroladores.

Figura 1.4

1.2.4 TERMINOLOGIA.

Las definiciones o términos utilizados en relación a las funciones del bus I2C, son las siguientes:

Maestro (Master): dispositivo que inicia una transferencia, determina la temporización y la dirección del tráfico de datos en el bus y termina un envío de datos. Cuando se conectan varios dispositivos maestros a un mismo bus la configuración obtenida se denomina "multi-maestro".

Esclavo (Slave): Son los dispositivos que reciben señales de comando y de reloj provenientes del dispositivo maestro. Cualquier dispositivo conectado al bus de datos incapaz de generar pulsos de reloj.

Bus Desocupado (Bus Free): Cuando ambas líneas (SDA y SCL) están inactivas., y se obtiene un estado lógico alto. Únicamente en este momento es cuando un dispositivo maestro puede comenzar a hacer uso del bus.

Comienzo (Start): se genera esta condición, cuando un dispositivo maestro hace ocupación del bus. La línea de datos (SDA) toma un estado bajo mientras la línea de reloj (SCL) permanece alta.

Parada (Stop): Cuando un dispositivo maestro deja libre el bus, genera esta condición. La línea de datos tomo un estado lógico alto mientras que el reloj permanece también en ese estado.

Dato Válido (Data Valid): Sucede cuando un dato presente en la línea SDA es estable mientras la línea SCL esta en nivel lógico alto.

Formato de Datos (Data Format): La transmisión se realiza a través de 8 bits o 1 byte. A cada byte le sigue un noveno pulso de reloj durante el cual el dispositivo receptor del byte debe generar un pulso de reconocimiento, conocido como ACK (Acknowledge). Esto se logra situando la línea de datos a un nivel lógico bajo mientras transcurre el noveno pulso de reloj.

Dirección (Address): Los dispositivos que poseen este bus, tienen su propia y única dirección de acceso, la cual viene pre establecida por el fabricante, también existen dispositivos en los cuales se puede establecer externamente una parte de la dirección de acceso. Lo que permite que varios dispositivos del mismo tipo se puedan conectar al mismo bus sin problemas de identificación. La dirección 00 es la denominada "de acceso general", por la cual responden todos los dispositivos conectados al bus.

Lectura/Escritura (Bit R/W): Cada dispositivo dispone de una dirección de 7 bits. El octavo bit (el menos significativo o LSB) enviado durante la operación de direccionamiento corresponde al bit que indica el tipo de operación a realizar. Si este bit es alto el dispositivo maestro lee información proveniente de un dispositivo esclavo. En cambio, si este bit es bajo el dispositivo maestro escribe información en un dispositivo esclavo.

Multi-Master: más de un master puede controlar el bus al mismo tiempo sin corrupción de los mensajes.

Arbitraje: Procedimiento que asegura que si uno o más master simultáneamente deciden controlar el Bus, solo uno es permitido a controlarlo y el mensaje saliente no es deteriorado.

Sincronización: Procedimiento para sincronizar las señales del reloj y de dos o mas dispositivos.

1.2.5 COMPONENTES DEL BUS I2C.

Este bus se basa en tres señales:

SDA (System Data): por la cual viajan los datos entre los dispositivos.
SCL (System Clock): por esta línea circulan los pulsos de reloj que sincronizan el sistema.
GND (Masa): Interconectada entre todos los dispositivos "enganchados" al bus.

Tanto SDA como SCL son del tipo drenador abierto, similar a las de colector abierto, pero estas están asociadas a un transistor de efecto de campo (FET). Se deben colocar en estado alto (conectar a VCC por medio de resistencias de Pull-Up) para lograr que el bus tenga una estructura que permita colocar en paralelo múltiples entradas y salidas.

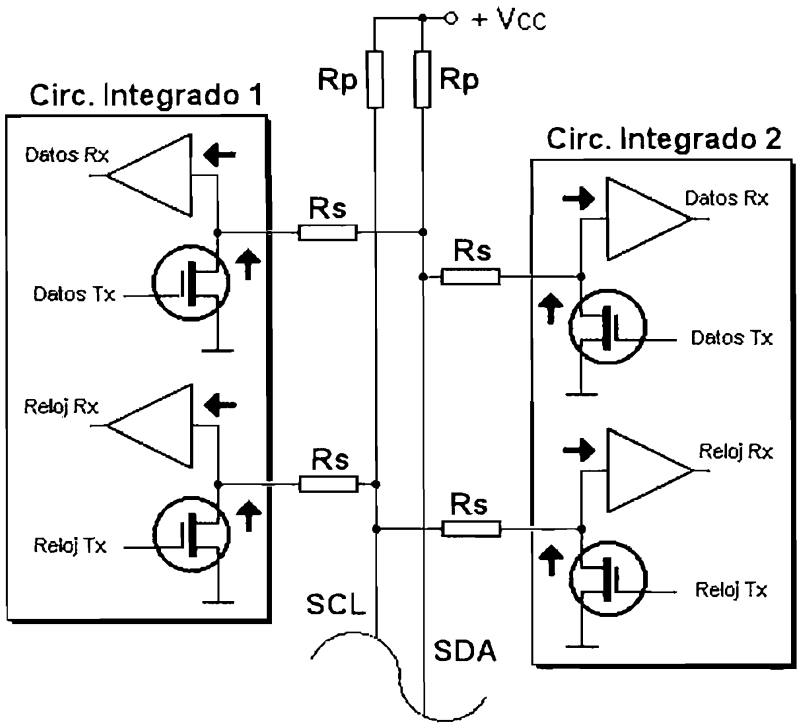


Figura 1.5

En la figura se muestra la configuración eléctrica básica del bus. Al estar inactivas las dos líneas presentan niveles altos. Al presentar las líneas una impedancia máxima de 400 pF, limita el número de dispositivos que se pueden conectar.

Las velocidades de transferencia en el bus son:

- ✓ Modo estándar aproximadamente a 100 Kbps
- ✓ Modo rápido aproximadamente a 400 Kbps.
- ✓ Modo alta Velocidad mas de 3.4 Mbps.

1.2.6 PROTOCOLO DEL BUS.

1.2.6.1 Características Generales

Para realizar una transmisión se debe respetar un protocolo, es así que para el bus I2C se tienen algunas normas. Tan pronto como el bus este libre, un dispositivo maestro puede ocuparlo, generando una condición de inicio. El primer byte transmitido luego de la condición de inicio, contiene los siete bits que componen la dirección del dispositivo destino seleccionado, y el octavo bit indica la operación deseada (lectura o escritura). Si el dispositivo al que va direccionada la información se encuentra en el bus, este responde enviando un pulso de reconocimiento o ACK. Luego de esto puede iniciarse la transferencia de información entre dispositivos.

Debido a la variedad de diferentes tecnologías usadas en los dispositivos conectados al Bus I2c los niveles lógicos de "0" (Bajo) y "1" (Alto) no esta fijados y dependen de la tensión de alimentación del circuito. Un pulso de reloj se genera por cada bit de datos transferidos.

Es decir para operar un esclavo en el bus I2C son necesarios seis simples códigos, suficientes para enviar o recibir información.

Un bit de inicio.
7 bits ó 10 bits de direccionamiento.
Un R/W bit que define si el esclavo es trasmisor o receptor.
Un bit de reconocimiento.
Mensaje dividido en Bytes
Un bit de stop.

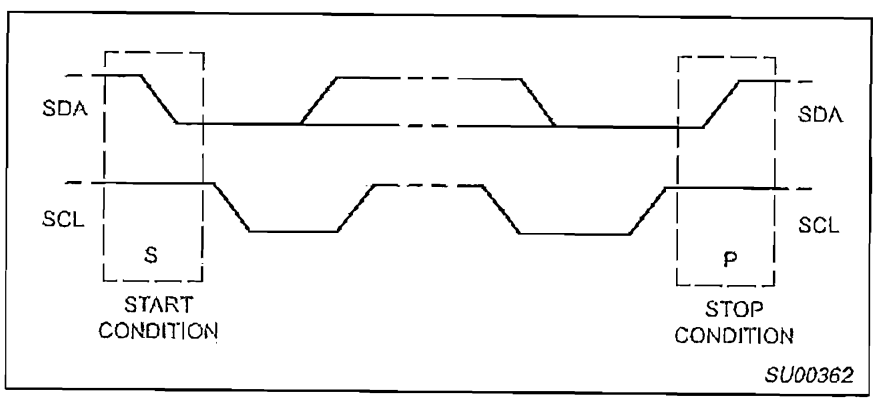
Cuando la señal R/W obtiene un nivel lógico bajo, el dispositivo maestro envía datos al dispositivo esclavo hasta que deja de recibir los pulsos de reconocimiento, o hasta que se hayan transmitido todos los datos.

En el caso contrario, es decir cuando la señal R/W tiene un nivel lógico alto, el dispositivo maestro genera pulsos de reloj durante los cuales el dispositivo esclavo puede enviar datos. Luego de cada byte recibido el dispositivo maestro (que en este momento esta recibiendo datos) genera un pulso de reconocimiento. El dispositivo maestro puede dejar libre el bus generando una condición de parada (Stop). Si se desea seguir transmitiendo, el dispositivo maestro puede generar otra condición de inicio en lugar de una condición de parada. Esta nueva condición de inicio se denomina "inicio repetitivo" y se puede emplear para direccionar un dispositivo esclavo diferente o para alterar el estado del bit lectura /escritura (R/W).

1.2.6.2 Condiciones de Inicio (Start) y Parada (Stop).

Para que la transferencia de datos en el Bus I2C se realice, se necesitan de dos condiciones básicas que son el Inicio y Stop. Estas son:

INICIO (START): Una transición de "1" a "0" (caída) en la línea de datos (SDA) mientras la línea de reloj (SCL) esta a "1".
PARADA (STOP): Una transición de "0" a "1" (ascenso) en la línea de datos (SDA) mientras la línea de reloj (SCL) esta a "1". Como se muestra en la siguiente figura.



Condiciones de Start y Stop

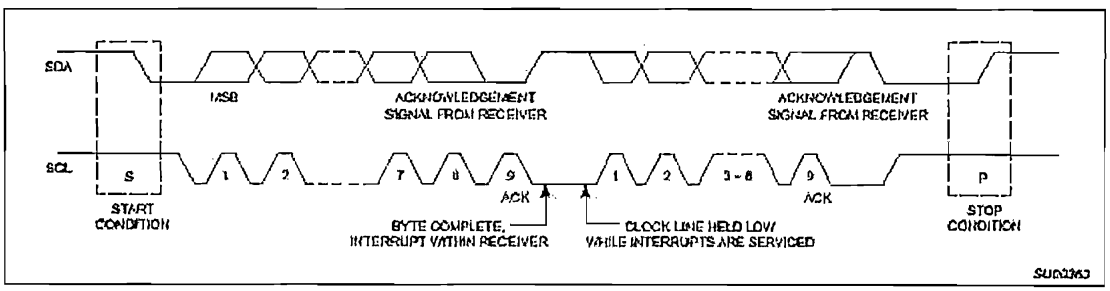
Figura 1.6

Tanto la condición de Inicio como la de Stop, siempre son generadas por el Master. El Bus I2C se considera ocupado después de la condición de inicio. El Bus se considera libre de nuevo luego de un cierto tiempo tras la condición de Stop.

Es decir al pulso "1" de la línea SCL le puede corresponder un pulso "0" o "1" de la línea SDA en función de la información del byte que se envíe, recordemos que cada bit de SDA le corresponde un bit SCL, pero nunca salvo en la condición de inicio a un bit de SCL le corresponde una situación de "1" a "0" o sea pasa por dos estados la línea SDA, al contrario ocurre en la condición de Stop que el master envía un bit a la línea SCL mientras cambia en la SDA de "0" a "1" durante el tiempo que esta enviando la señal de "1" a SCL.

1.2.6.3 Transferencia De Datos.

No existe restricción para el número de bytes que se pueden enviar en la línea SDA. Cada byte debe ir seguido por un bit de reconocimiento, el byte de datos se transfiere empezando por el bit de menos peso (7) precedido por el bit de reconocimiento (ACK). Como se muestra en la siguiente figura.



Transferencia de datos en el Bus I2C

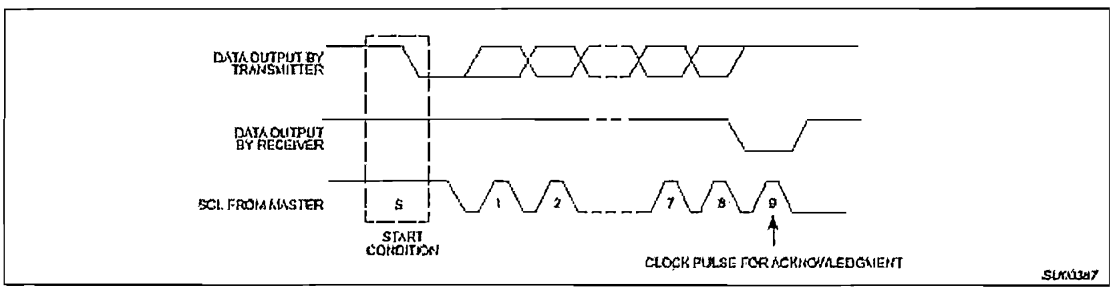
Figura 1.7

Si un dispositivo esclavo no puede recibir o transmitir un byte de datos completo hasta que haya acabado alguno de los trabajos que realiza, puede mantener la línea SCL a "0" lo que fuerza al Master a permanecer en un estado de espera. Los datos

continúan transfiriéndose cuando el dispositivo esclavo esta listo para otro byte de datos y desbloquea la línea de reloj (SCL).

1.2.6.4 Bit De Reconocimiento (Ack).

Para la transferencia de datos este bit es obligatorio. El pulso de reloj correspondiente al bit de reconocimiento (ACK) es generado por el Master. El trasmisor desbloquea la línea SDA ("1") durante el pulso de reconocimiento. El receptor debe poner a "0" la línea SDA durante el pulso ACK de modo que siga siendo "0" durante el tiempo que el master genera el pulso "1" de ACK. Esto se muestra en la siguiente figura.



Reconocimiento en el Bus I2C

Figura 1.8

Un receptor generalmente esta obligado a generar un ACK cuando ha sido direccionado, después de que cada byte ha sido recibido.

Cuando un dispositivo esclavo no genera el bit ACK (porque esta haciendo otra cosa y no puede atender el Bus) debe mantener la línea SDA a nivel "1" durante el bit ACK. El Master entonces puede generar una condición de STOP abortando la transferencia de datos o repetir la condición de Inicio enviando una nueva transferencia de datos.

El Master se encarga de detectar, cuando un esclavo – receptor no desea recibir mas bytes, y ya no envía información. Esta condición se reconoce, porque el esclavo no genera el bit ACK en el primer byte que sigue. El esclavo pone la línea SDA a "1" lo que es detectado por el Master el cual genera la condición de Stop o repite la condición de inicio.

Si un Master – receptor esta recibiendo datos de un Esclavo – transmisor debe generar un bit ACK tras cada byte recibido del transmisor, para finalizar la transferencia de datos no debe generar el ACK tras el último byte enviado por el esclavo. El esclavo – transmisor debe permitir desbloquear la línea SDA generando el Master la condición de stop o de Inicio.

1.2.6.5 Arbitraje y Generación de Señales de Reloj.

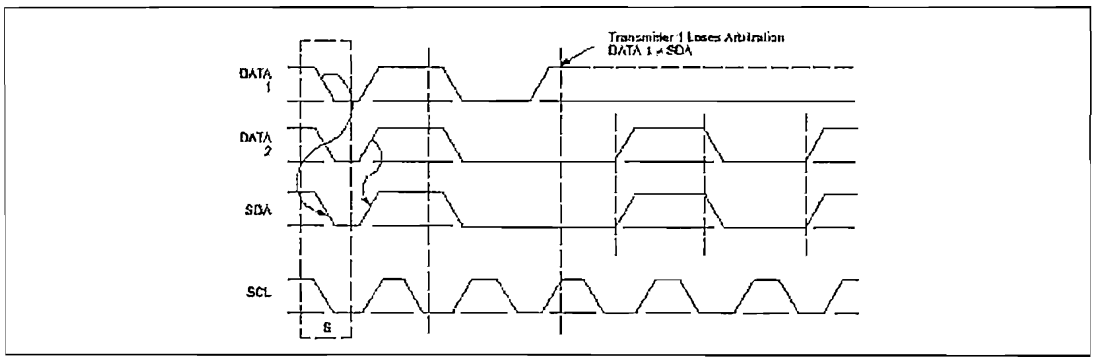
Para la sincronización los dispositivos que actúan como Master generan su propia señal de reloj sobre la línea SCL al producirse la transferencia de datos en el bus I2C. Cuando el reloj se encuentra en estado "1" los bits de datos son válidos. Un control es necesario para mantener un orden en los diversos bits que se generan.

Usando una conexión AND entre todos los dispositivos del bus a la línea SCL, se puede producir la sincronización del reloj. Es decir cuando el Master tiene una transición de "1" a "0" en la línea SCL, logra que esta línea se mantenga en este estado. Sin embargo la transición de "0" a "1", no produce un cambio en la línea SCL, si otro reloj se encuentra en un periodo de "0". En consecuencia permanecerá en "0" la línea SCL, tanto como el periodo mas largo de cualquier dispositivo cuyo nivel sea "0". Los dispositivos que tienen un periodo más corto de reloj "0" entran en un periodo de espera.

La línea de reloj se desbloquea, cuando todos los dispositivos conectados al bus terminan con su periodo "0", es decir pasa a nivel "1". Por lo que hay que diferenciar entre los estados de reloj de los dispositivos y los estados de la línea SCL, y todos

los dispositivos empiezan a nivel "1". El primer dispositivo que completa su nivel "1" pone nuevamente la línea SCL a "0".

"Un master puede iniciar una transmisión solo si el bus esta libre. Dos o más Master pueden generar una condición de inicio en el bus, lo que da como resultado una condición de inicio general. Cada master debe comprobar si el bit de datos que transmite junto a su pulso de reloj, coincide con el nivel lógico en la línea de datos SDA. El arbitraje se realiza sobre la línea SDA, mientras la línea SCL esta a nivel "1", de una manera tal que el master que transmite un nivel "1", pierda el arbitraje sobre otro master que envía un nivel "0" a la línea de datos SDA. Esta situación continua hasta que se detecte la condición de Stop generada por el master que se hizo cargo del bus." [1] Como se muestra en la siguiente figura.



Proceso de arbitraje entre dos master

Figura 1.9

El arbitraje se produce entre dos o mas master involucrados, dependiendo del número de microcontroladores que se encuentren conectados al bus, este proceso no afecta los datos transferidos inicialmente por el master ganador..

"El arbitraje puede continuar varios bits hasta que se de la circunstancia de control de Bus por uno de los Master "[1]

Los master perdedores luego del arbitraje se deben poner en modo master – receptor y esclavo, en vista de que los datos que envía el master dominante pueden ser para uno de ellos. El master perdedor puede generar pulsos de reloj hasta el fin del byte en el cual se determina que pierde el arbitraje.

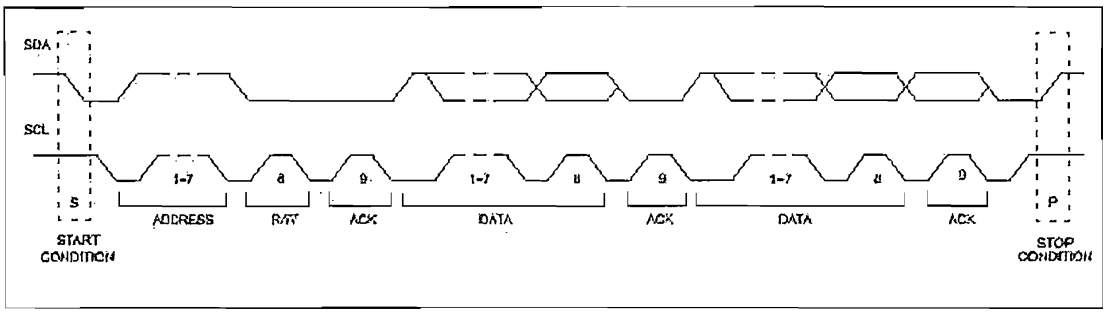
"En el momento que un Master toma el control solo este master toma las decisiones y genera los códigos de dirección, no existen master centrales, ni existen ordenes prioritarias en el Bus.

Especial atención debe ponerse si durante una transferencia de datos el procedimiento de arbitraje esta todavía en proceso justo en el momento en que se envía al bus una condición de stop. Es posible que esta situación pueda ocurrir, en este caso el master afectado debe mandar códigos de Inicio o stop"[1]

1.2.6.6 Formato de los Datos.

"Después de la condición de Start un código de dirección de un esclavo es enviada esta dirección tiene 7 bits seguidos por un octavo código que corresponde a una dirección R/W (0 – indica transmisión/ 1 – indica solicitud de datos). Una transferencia de datos siempre acaba con una condición de stop generada por el master, sin embargo si un master todavía desea comunicarse con el bus puede generar repetidamente condiciones de Start y direccionar a otro esclavo sin generar primero la condición de stop "[1]

El formato de los datos transferidos, se muestra en la siguiente figura.



Transferencia completa de datos

Figura 1.10

Varias combinaciones de lectura y escritura son posibles dentro de una misma transferencia de datos.

Los posibles formatos de transferencia son:

1.- Master transmite al esclavo – receptor.

Secuencia de Transmisión del Master

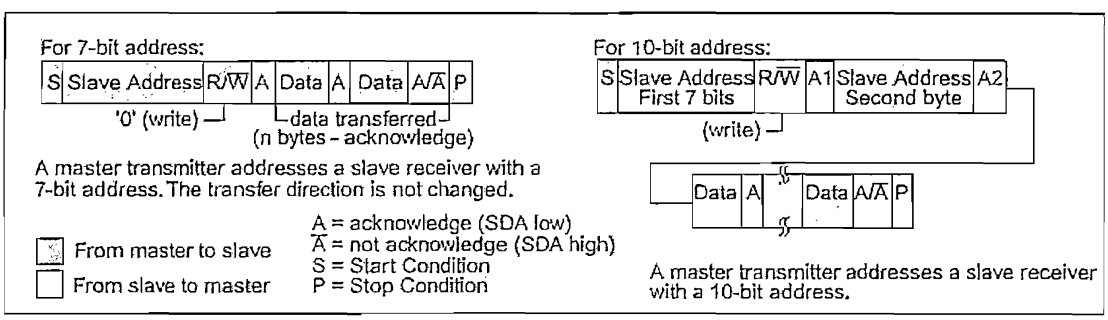


Figura 1.11

2.- Master lee a un esclavo inmediatamente después del primer byte.

En el primer reconocimiento el master – transmisor se convierte en un master – receptor y el esclavo - receptor es un esclavo – transmisor. El primer reconocimiento es aun generado por el esclavo. La condición de stop es generada por el Master, el

cual ha enviado previamente un reconocimiento. Esto se muestra en al siguiente figura.

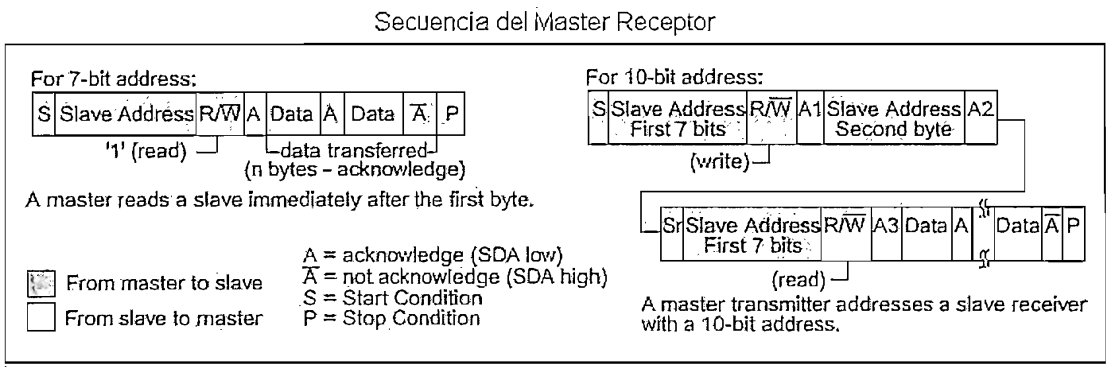


Figura 1.12

3.- Formato Combinado.

"Durante un cambio de dirección dentro de una transferencia, la condición de Start y la dirección del esclavo son ambos repetidos, pero con el bit R/W invertido. Si un Master – receptor envía una condición repetida de Star, el esclavo previamente ha enviado un no-reconocimiento." [1] Tal como se muestra en al siguiente figura.

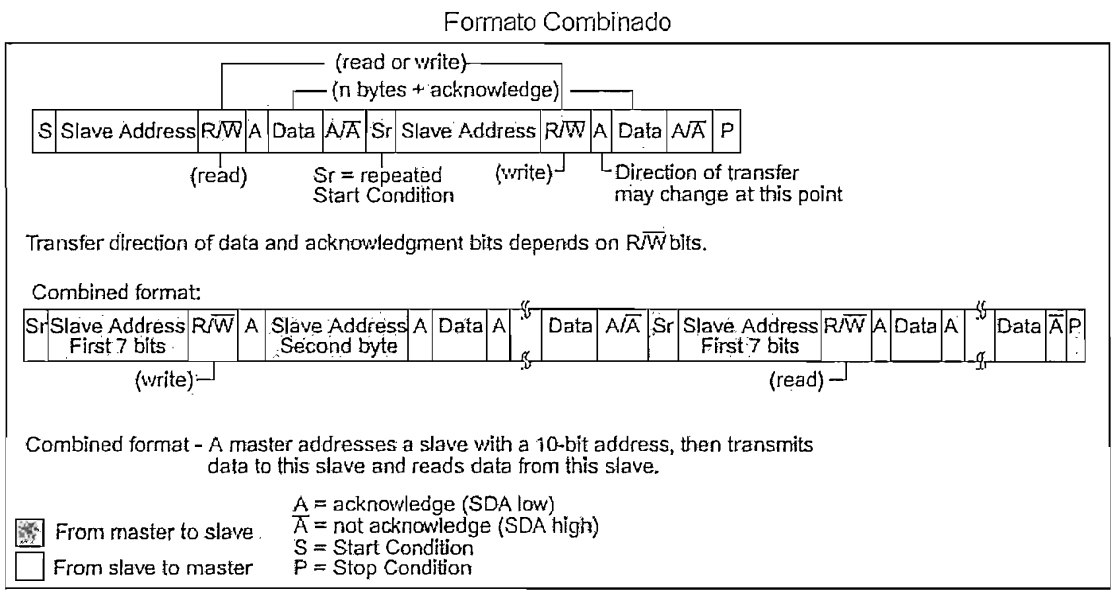


Figura 1.13

Notas:

"Se pueden combinar diversos formatos de direccionamiento."
"Las direcciones para el mayor o menor acceso a las posiciones de la memorias debe ser tomada por el diseñador del dispositivo."
"Durante le primer byte de datos la posición de la memoria interna debe ser escrita."
"Después de la condición Star. La dirección del esclavo es repetida, los datos pueden ser transferidos."
"Cada byte es seguido por un bit de reconocimiento como indican los bloques en la secuencia."
"Una condición de Start inmediatamente seguida por una condición de Stop es un formato ilegal."
"Los dispositivos compatibles con el bus I2C deben poder reajustar su bus lógico a la recepción de una o más condiciones de Start."

1.2.6.7 Direccionamiento.

Para realizar el direccionamiento en el bus I2C, el byte que va después de la condición de Start, determina el dispositivo esclavo seleccionado.

Para direccionar a todos los dispositivos se usa la "llamada general" (byte 00000000), al usar esta dirección, en teoría todos los dispositivos deben responder con un reconocimiento (A), sin embargo pueden haber dispositivos condicionados a ignorar esta dirección. El segundo byte de la "llamada general" **DEFINE** entonces la acción a tomar.

"Hay dos formatos de dirección. El más simple es el formato de 7-bit con un bit R/W que permite direccionar hasta 128 dispositivos, que en la práctica se reduce a 112 debido a que las restantes direcciones son de uso reservado.

El más complejo es el de 10 bits con un bit R/W. Para el formato de 10-bit, dos bytes deben ser transmitidos con los primeros cinco bits que especifiquen una dirección de 10-bit." [1]. Todo esto se muestra en las siguientes figuras.

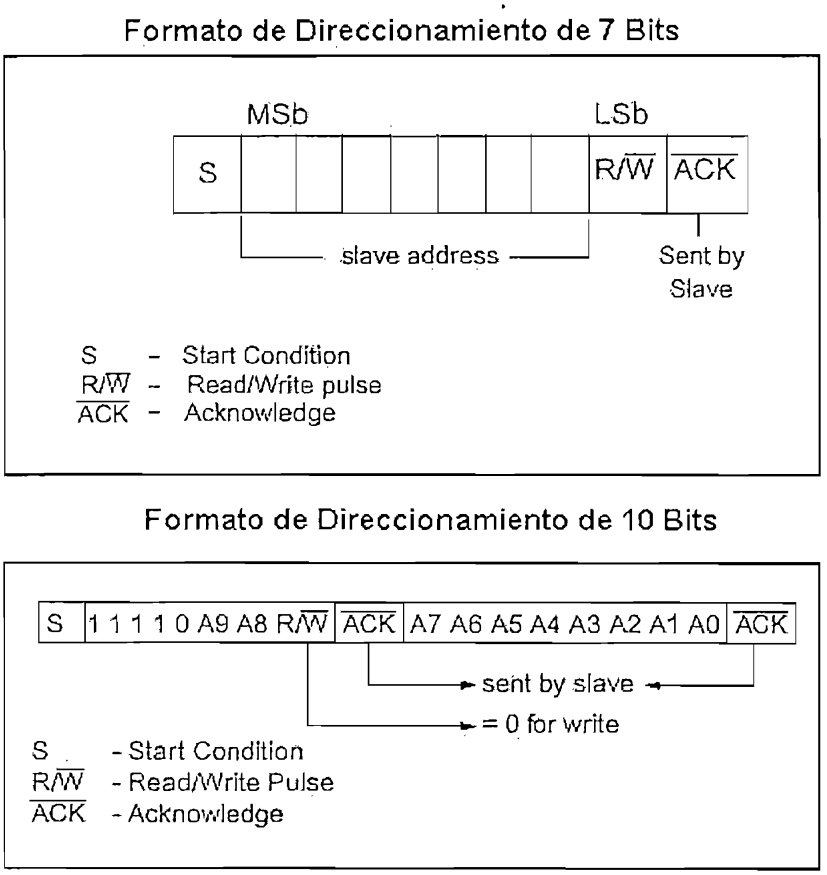
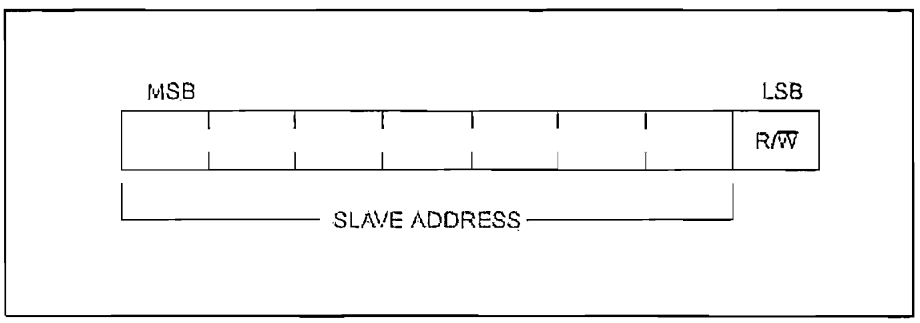


Figura 1.14

Al usar un direccionamiento de 10-bits, se puede usar hasta 1024 direcciones adicionales, para no tener problemas en la localización de direcciones esclavas, cuando el número de dispositivos I2C ha crecido rápidamente.

El direccionamiento de 10-bits no afecta a los dispositivos que tienen un direccionamiento de 7-bits, lo que permite conectarlos al mismo bus I2C, y ambos tipos de dispositivos pueden ser usados en sistemas con modos Standard, Fast o High – Speed.

En los primeros 7 bits del primer byte, se define la dirección del esclavo, como se muestra en la siguiente figura.



Formato del Primer Byte

Figura 1.15

La acción a tomar se determina en el octavo bit, con un "0" en esta posición, significa que el Master escribirá información en el esclavo seleccionado, y con un "1" significa que el Master leerá información del esclavo.

Si luego de la condición de inicio el Master envía una dirección, cada dispositivo conectado al bus analiza los 7 primeros bits de la dirección con la suya propia, y si esta coincide este es el dispositivo direccionado, y será un esclavo-receptor o esclavo-emisor dependiendo del bit R/W.

Al tener la dirección una parte fija y una programable, se pueden conectar dispositivos idénticos al bus, se los activa mediante la parte fija y se los controla mediante la parte programable.

Philips determinó direcciones reservadas que no se deben usar en el bus I2C. La combinación 11110xx de las direcciones esclavo se reservan para las direcciones de 10 bits.

La siguiente tabla muestra 8 direcciones reservadas.

Dirección esclavo	R/bit	Descripción
0000 000	0	Dirección de llamada general
0000 000	1	Byte de Inicio
0000 001	X	dirección CBUS
0000 010	X	Dirección reservada para bus de diferentes formatos
0000 011	X	Reservadas para futuros propósitos
0000 1XX	X	
1111 1XX	X	
1111 0XX	X	Direccionamiento esclavo de 10 bits

"No se permite reconocer a ningún dispositivo en el byte de Start"
"La dirección CBUS están reservadas para permitir la compatibilidad entre dispositivos I2C y CBUS en el mismo sistema. Los dispositivos compatibles del Bus I2C no están autorizados para responder a esta dirección."

1.2.6.8 Especificaciones Eléctricas y de Tiempos.

En vista de que al Bus I2C se pueden conectar diferentes dispositivos, las tensiones dependen de dos factores, de la tensión necesaria para cada uno de los dispositivos, y de una cierta normativa bastante elástica para las líneas SDA y SCL.

Por lo general la alimentación de las líneas SCL y SDA, debe ser 5 V, manteniendo la siguiente tolerancia:

Máxima tensión permitida a nivel bajo ("0") ----> 1.5 V

Mínima tensión permitida a nivel alto ("1") -----> 3 V

Características de las líneas SDA y SCL para los dispositivos I2C.

PARAMETRO	Símb.	Standard-Mode I2C Bus		Fast – Mode I2C Bus		Unidad
		Min.	Máx.	Min.	Máx	
Frecuencia del reloj SCL	f_{SCL}	0	100	0	400	Khz.
Tiempo libre del bus entre la condición de Stop y Start	t_{BUF}	4.7	-	1.3	-	μs
Tiempo de espera (repetir) para la condición de inicio. Después de este periodo se genera el primer pulso de reloj	$t_{HD;STA}$	4.0	-	0.6	-	μs
Periodo bajo del reloj SCL	t_{LOW}	4.7	-	1.3	-	μs

Periodo alto del reloj SCL	t_{HIGH}	4.0	-	0.6	-	μs
Tiempo de instalación para repetir la condición de inicio.	$t_{SU;STA}$	4.7	-	0.6	-	μs
Tiempo de espera de datos: Para Master compatible con CBUS Para dispositivos I2C	$t_{HD;DAT}$	 5.0 $0^{1)}$	- -	- $0^{1)}$	- $0.9^{2)}$	 μs μs
Tiempo De instalación de datos	$t_{SU;DAT}$	250	-	$100^{3)}$	-	ηs
Incremento de tiempo en las dos señales SDA y SCL	t_r	-	1000	$20+0.1C_b^{4)}$	300	ηs
Decremento de tiempo en las dos señales SDA y SCL	t_f	-	300	$20+0.1C_b^{4)}$	300	ηs
Tiempo de instalación de la condición de inicio	$t_{SU;STO}$	4.0	-	0.6	-	μs
Capacidad de carga para las líneas de bus	C_b	-	400	-	400	pF

CAPITULO 2.

2. ESPECIFICACIONES TÉCNICAS DEL PROTOTIPO INTELIGENTE DE MONITOREO PARA UN PARQUEADERO - PRINMOP.

2.1 CARACTERISTICAS GENERALES DEL SISTEMA – PRINMOP

El sistema en general está constituido por 4 módulos electrónicos y el software que se instala en la PC, donde cada módulo es controlado en base a PICs.

Los módulos construidos son los siguientes:

- Módulo de Detección Magnética – **MODEMAG** (Controlado por el PIC # 1)
- Panel Mímico Informativo – **PMI** (Controlado por el PIC # 2)
- Módulo de Control - **MODCON** (Controlado por el PIC # 3 y 4)
- Módulo de Interfaz con la PC – **MINT-PC** (Controlado por el PIC # 5)
- Software de visualización y reporte – **SVR** (Diseñado en Visual Basic).

El módulo de detección, recoge las señales que entregan los loops magnéticos instalados, y los entrega al módulo de interfaz con la PC, usando una codificación diferente tanto para la señal de plaza ocupada como para la señal de plaza libre, además cada modulo MODEMAG instalado utiliza codificación diferente, respecto a los demás módulos de detección, para facilitar la modularidad y crecimiento.

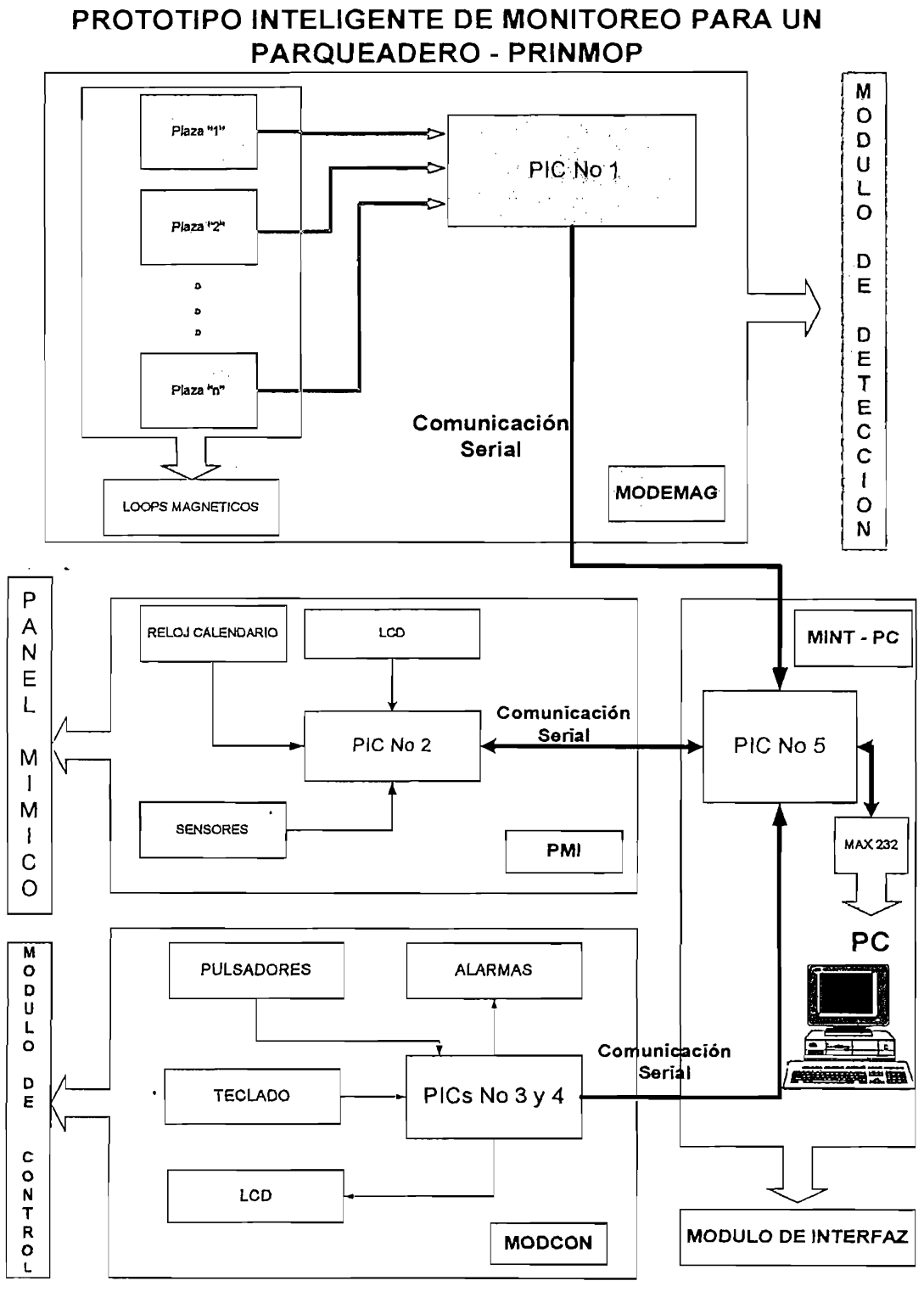
El Panel mímico informativo, mediante comunicación I2C entre algunos de sus componentes, y de esta forma ocupando tan solo dos hilos para su comunicación, lo que implica menor espacio físico en la tarjera electrónica, pero sobre todo la capacidad de colocar una mayor cantidad de dispositivos en el modulo. Entrega al usuario la información que proviene del módulo de interfaz con la PC, que le llega de los diferentes módulos conectados al mismo, mediante comunicación serial. Además también mediante el uso de reloj calendario envía a su panel informativo información de la fecha y hora.

El módulo de control envía información para que se muestre en el panel mímico informativo, y también genera alarmas sonoras y controla el abrir y cerrar del portero y de cualquier otro dispositivo que se necesite manipular desde este módulo y que facilite la labor del operador del sistema, ya sea en situaciones normales para poder guiar de mejor forma a las personas y vehículos que ingresan al parqueadero, como en situaciones de emergencia que se necesite que tanto personas como vehículos despejen el área.

El módulo de interfaz, recoge datos de los otros módulos y los envía hacia la PC, así también envía datos provenientes de la PC a algunos de los otros módulos, este modulo es la parte central del sistema, puesto que a través del mismo pasan las señales hacia los diferentes componentes del sistema.

El software de visualización y reporte, esta diseñado en Visual Basic y Access para observar en su panel las plazas que se encuentran ocupadas y las libres, poder ingresar las tarjetas de los usuarios y así poder facturar, así como también se puede tener un reporte del tiempo de ocupación y una estadística de uso de las plazas.

El siguiente esquema muestra un diagrama general del sistema:



2.2 CARACTERISTICAS DE LOS MODULOS DEL PRINMOP.

2.2.1. MODULO DE DETECCION MAGNETICA - MODEMAG.

2.2.1.1 Características.

Este módulo se encarga de receptar las señales que envía cada loop magnético colocado en las plaza del parqueadero, el sistema del loop actúa como un interruptor, que cada vez que un vehículo se estaciona sobre una plaza, este interruptor se cierra y se envía un señal al modulo de detección, y cada vez que el vehículo se retira de la plaza el interruptor se abre, y se envía una nueva señal al módulo de detección, en este modulo de detección el PIC censa sus puertos configurados como entradas y envía esta señal al modulo de interfaz con la PC.

El PIC del módulo de detección recoge las señales de sus puertos que pueden ser uno lógico, cuando no existe presencia de vehículo en la plaza que se encuentra sensando, puesto que todos los puertos conectados a un loop magnético se encuentran conectados a una resistencia de Pull Up y el loop actuando como interruptor se encuentra abierto, o puede recibir un cero lógico como señal de que un vehículo se encuentra ocupando la plaza, puesto que el loop actuando como interruptor se encuentra cerrado y conectando cada puerto a tierra; este uno lógico o cero lógico se transforma en una señal compuesta por dos caracteres, y estos son el número de la plaza y el identificativo del módulo al que pertenece, esto se debe a que se pueden instalar varios módulos de detección. Si el identificativo del módulo se encuentra antes del número de la plaza quiere decir que la plaza esta siendo ocupada, por ejemplo la señal emitida será "A1" para la plaza número 1 del módulo "A", o caso contrario puede generarse la señal compuesta primero por el número de la plaza, seguida del identificativo del módulo, por ejemplo la señal emitida puede ser "1A", que significa que la plaza 1 del modulo "A", se encuentra libre.

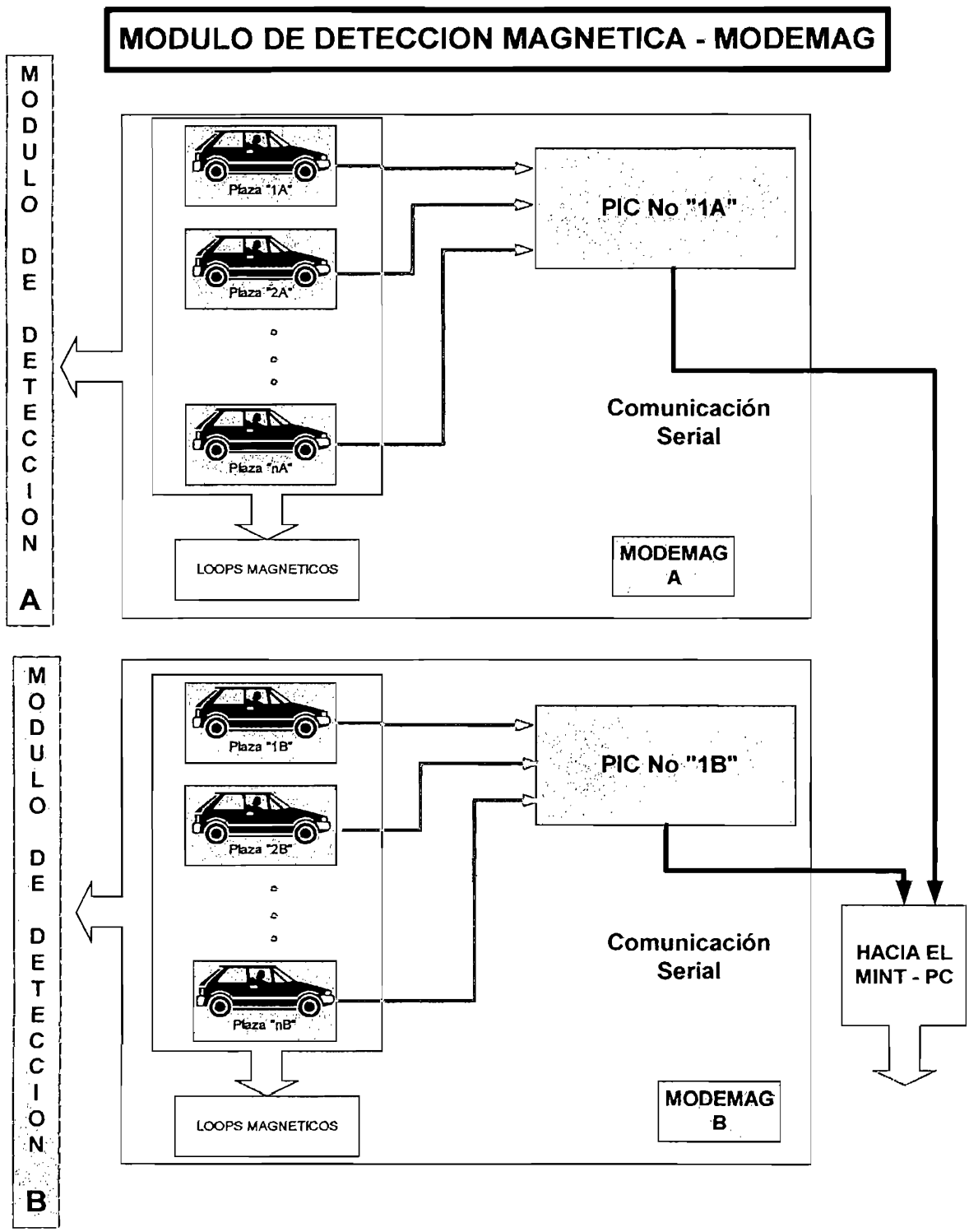
El sistema brinda una característica modular, es decir que si el número de plazas crece, es cuestión de instalar un nuevo modulo de detección, con sus respectivos loops magnéticos, y este nuevo modulo conectarlo serialmente al modulo de interfaz con la PC, esta característica permite crecer fácilmente. Las plazas de cada módulo se encuentran identificadas por un número y una letra, el número identifica la plaza, y la letra identifica a cada módulo.

En el prototipo cada módulo puede controlar 10 plazas, en la siguiente tabla se muestra la notación utilizada para identificar a las plazas de cada modulo.

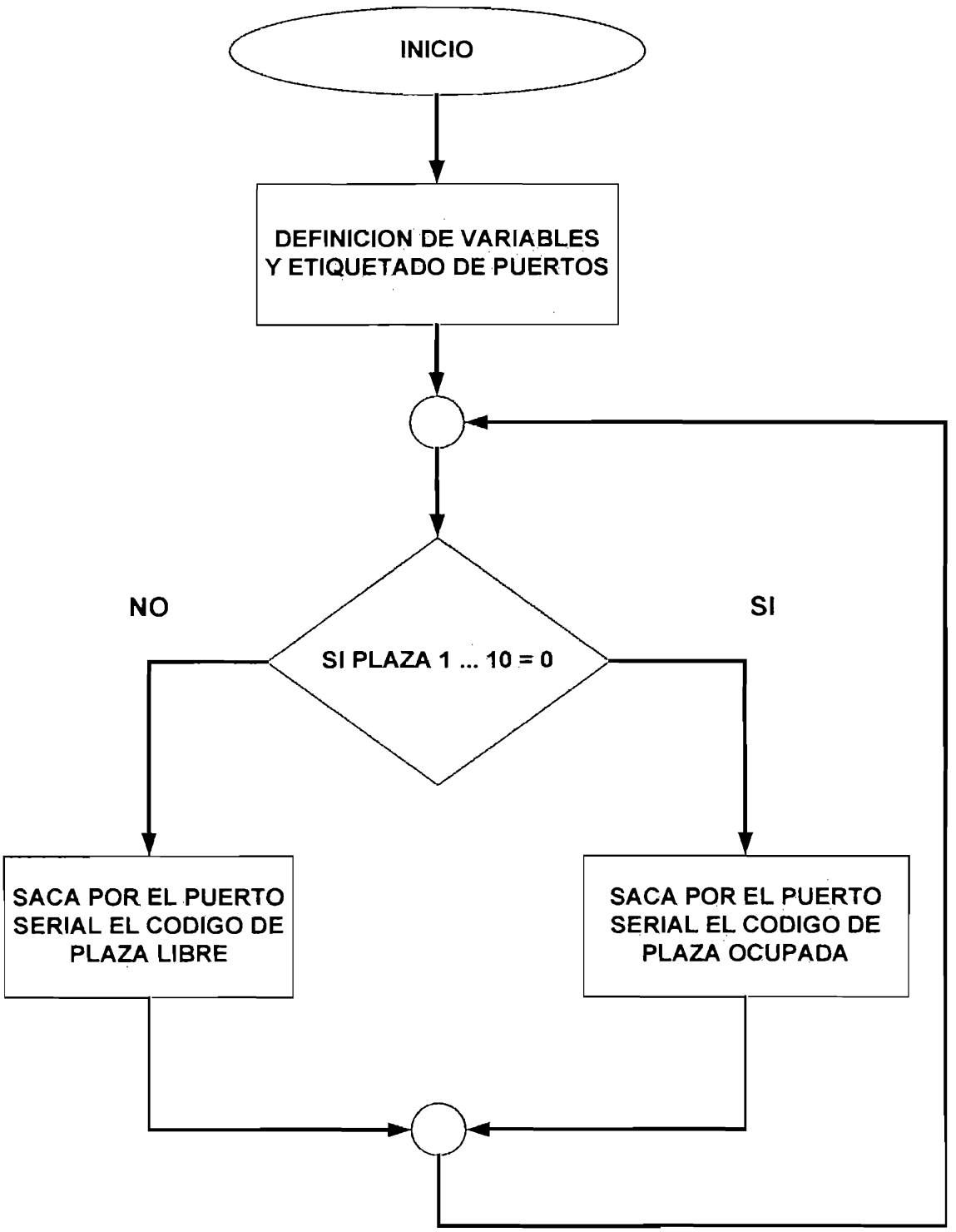
MODEMAG "A"		MODEMAG "B"		MODEMAG "C"	
No. de plaza	Identificativo del módulo	No. de plaza	Identificativo del módulo	No. de plaza	Identificativo del módulo
1	0A	1	0B	1	0C
2	1A	2	1B	2	1C
3	2A	3	2B	3	2C
4	3A	4	3B	4	3C
5	4A	5	4B	5	4C
6	5A	6	5B	6	5C
7	6A	7	6B	7	6C
8	7A	8	7B	8	7C
9	8A	9	8B	9	8C
10	9A	10	9B	10	9C

El identificativo del módulo se ha iniciado desde cero, para ahorrar espacio en el programa que se incluye dentro del PIC y también lograr una mejor rapidez en el barrido de las plazas. Cada módulo de detección, para conectarse al modulo de interfaz con la PC, utiliza un pin del PIC del módulo de interfaz, esto permite crecer fácilmente en el numero de módulos de detección.

2.2.1.2 Diagrama de Bloques.



2.2.1.3 Flujograma del Programa del Pic #1



2.2.2 PANEL MIMICO INFORMATIVO – PMI

2.2.2.1 Características.

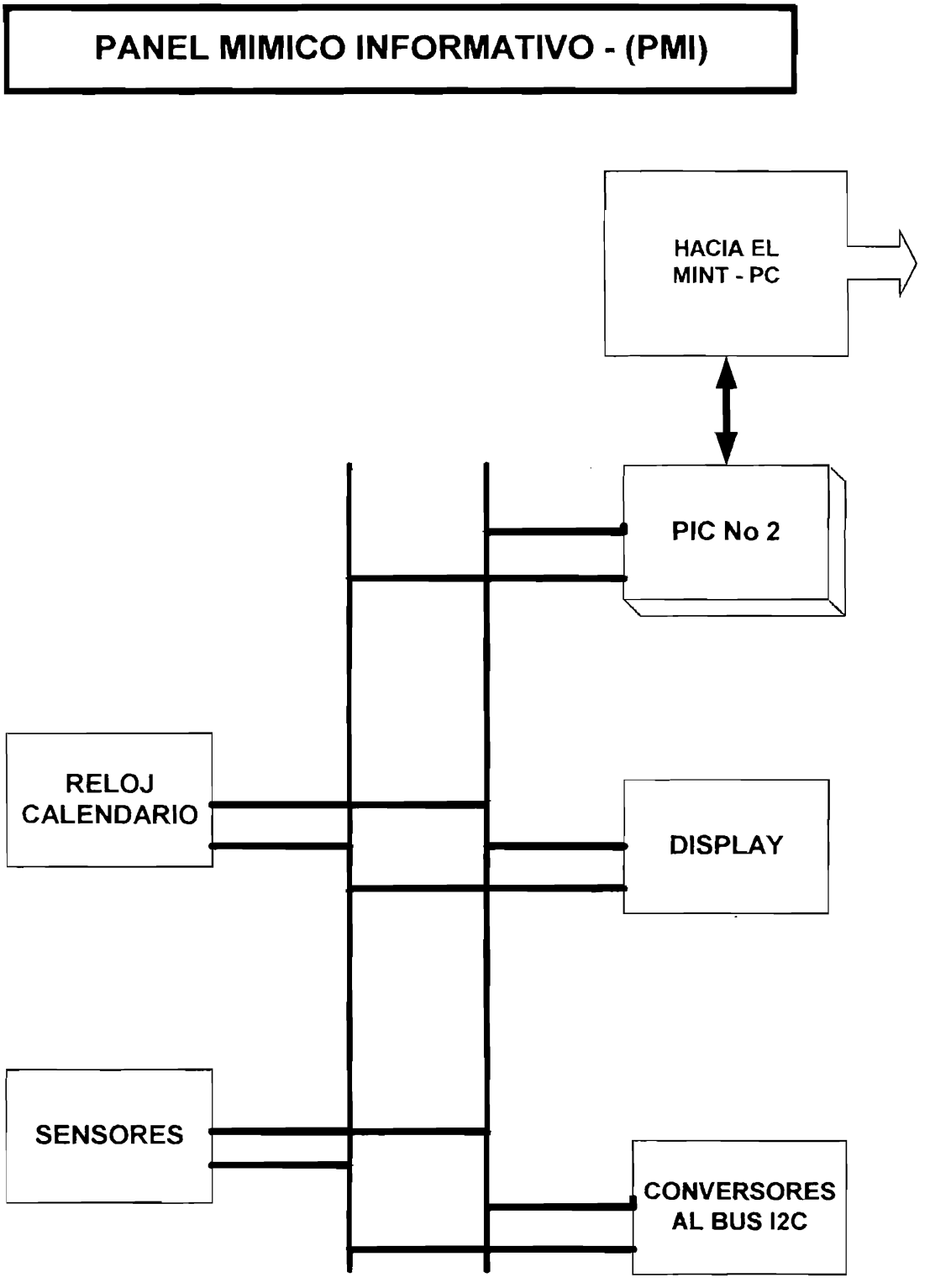
Este módulo se encarga de transmitir información al usuario, que proviene del módulo de interfaz con la PC, de recoger información de los sensores para informar al operador sobre el estado del sistema, y de tomar las señales del MODEMAG y reflejarlas en el panel que se muestra al usuario de las plazas libres y ocupadas.

La comunicación entre algunos dispositivos del panel informativo se realiza mediante el protocolo I2C, para esto se utiliza dispositivos con la característica I2C. Este protocolo permite reducir espacio en la placa electrónica, debido a que utiliza para la comunicación entre los diferentes dispositivos tan solo dos hilos, además el protocolo permite incorporar un número bastante grande de dispositivos, que tan solo se ve limitado por la capacitancia de la línea. Los demás dispositivos se manejan utilizando el gran número de puertos que posee el PIC 16F877.

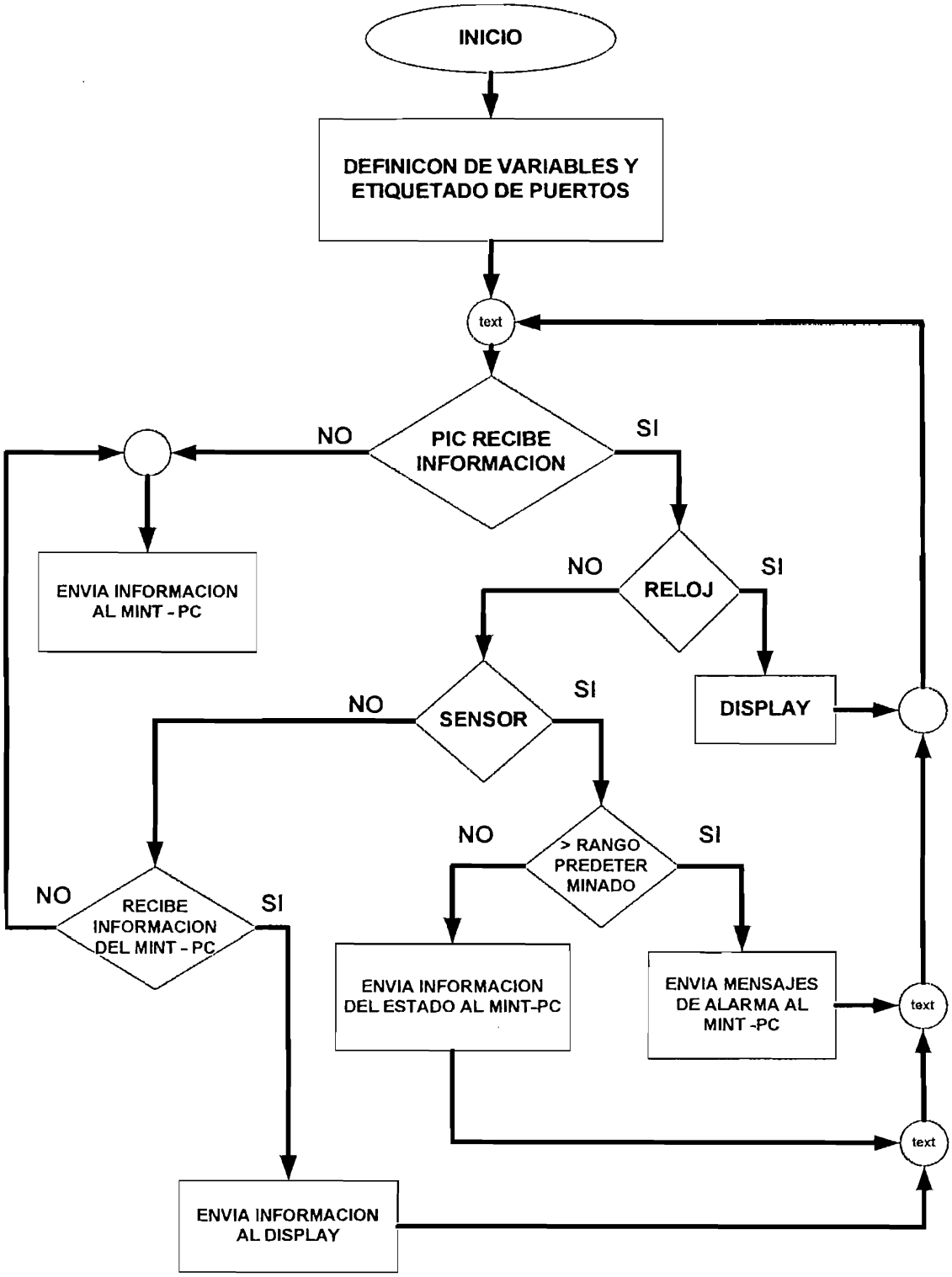
En este módulo se puede transmitir mensajes pregrabados desde el modulo de control, estos por lo general son mensajes que informan de alguna situación especial, así como también refleja información del reloj calendario que tiene el módulo, esta información también se envía al operador del sistema.

El panel mímico se coloca a la entrada del parqueadero, para que pueda transmitir la información al usuario y este pueda ingresar con mayor facilidad, es decir con un mejor conocimiento de la localización de las plazas libres y de la cantidad de las mismas.

2.2.2.2 Diagrama De Bloques.



2.2.2.3 Flujograma Del Programa Del Pic #2



2.2.3 MODULO DE CONTROL

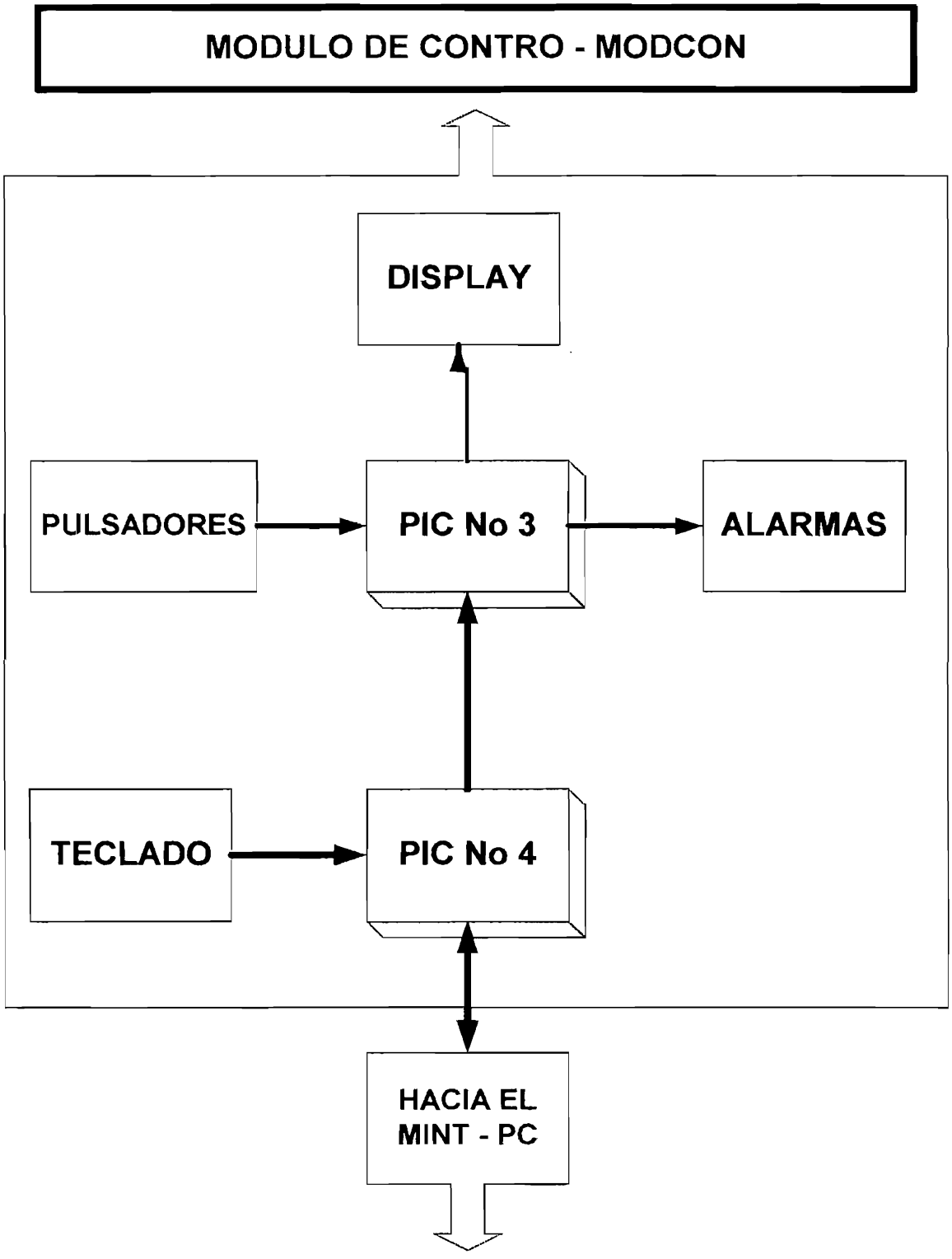
2.2.3.1 Características.

Este módulo se encuentra formado por dos PICs, uno de ellos maneja el LCD, pulsadores y las alarmas, mientras que el otro se encarga de controlar el teclado y la comunicación serial con el modulo de interfaz.

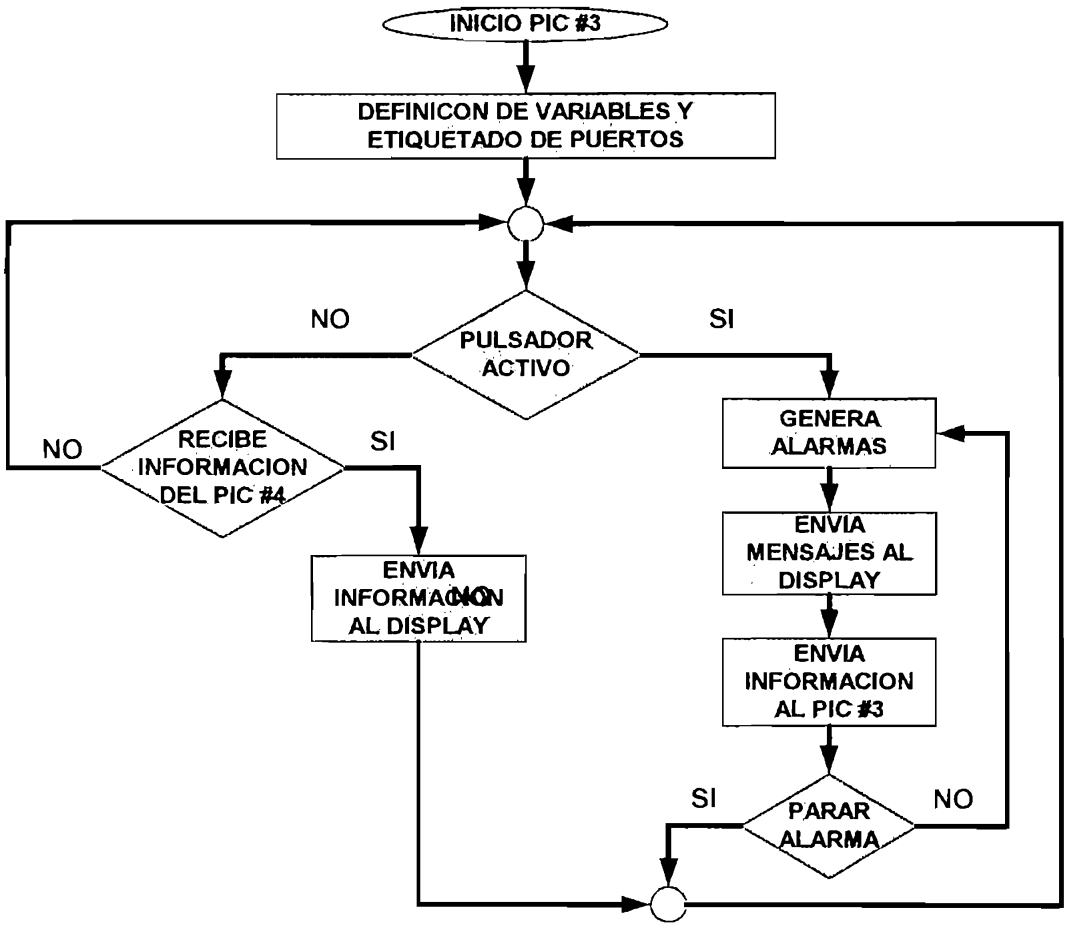
Desde este módulo el operador del sistema puede generar alarmas sonoras en caso de presentarse situaciones especiales como robos, incendios, inundaciones y demás percances que pueden poner en peligro la integridad de las personas y vehículos dentro del parqueadero.

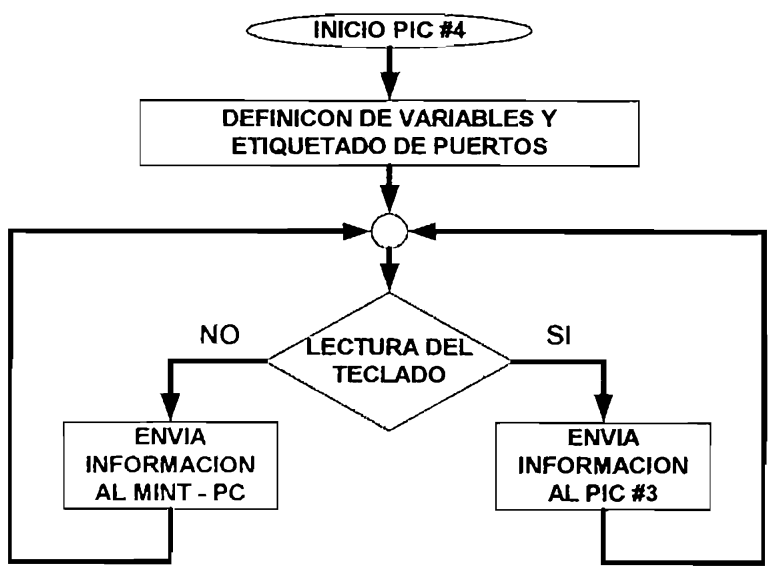
En su display de cristal líquido puede visualizar los mensajes pregrabados que envía al panel mímico, estos mensajes sirven para informar de alguna situación especial a los usuarios que se encuentran en la entrada. Para iniciar el envío de los mensajes pregrabados, es necesario ingresar una clave que permite ingresar al menú de los mensajes y seleccionar el más adecuado para transmitir la información requerida a los usuarios.

2.2.3.2 Diagrama De Bloques



2.2.3.3 Flujograma Del Programa De Los Pics No 3 Y 4





2.2.4 MODULO DE INTERFAZ CON LA PC (MINT – PC)

2.2.4.1 Características

Este módulo se encarga de recibir las señales que envían los MODEMAG instalados, ya sea de una plaza ocupada o una plaza libre, dependiendo del orden del número de la plaza y del identificador del módulo, y esta información la envía a la PC para que el programa lo traduzca y lo muestre al operador del sistema de una forma gráfica si la plaza esta libre u ocupada.

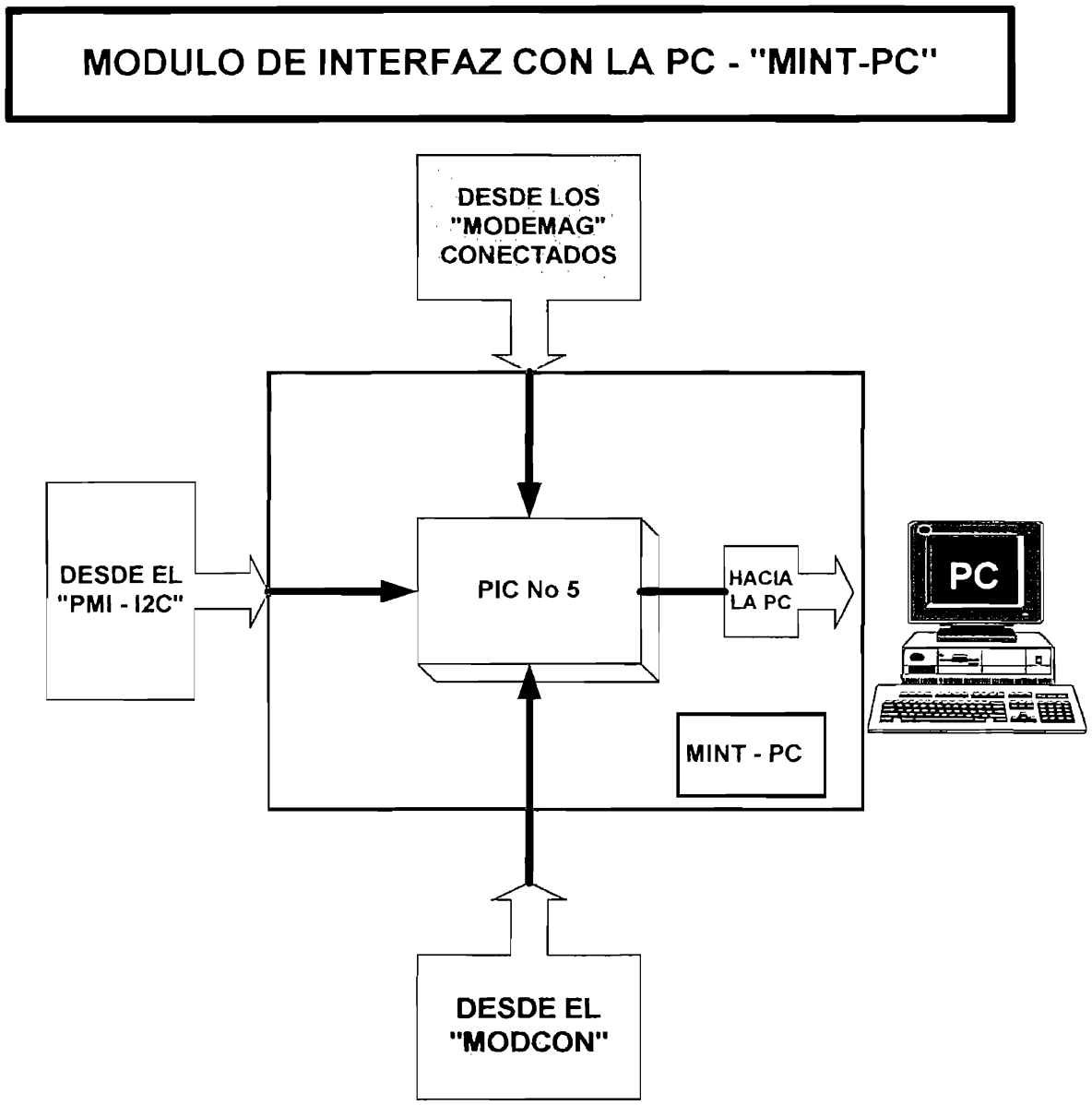
También se encarga de transmitir la información hacia el panel mímico proveniente del MODCON

En este módulo se encuentran los diferentes conectores para la instalación de los MODEMAG de expansión. A medida que se van expandiendo el número de plazas, los módulos de detección también crecen y el MINT – PC debe procesar mayor información.

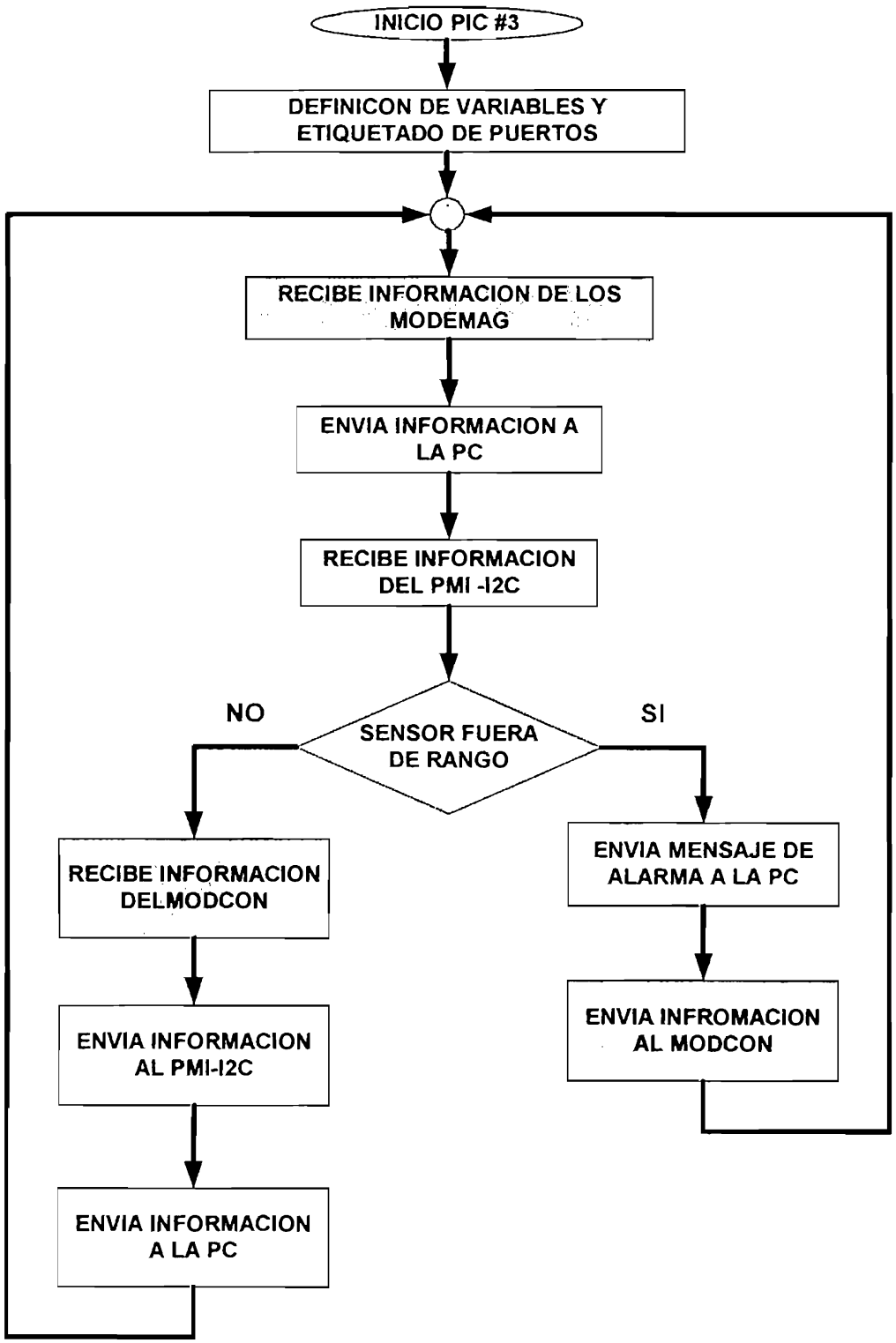
Este módulo puede ser mejorado sustancialmente con un PIC de mejores características, para así proporcionar al sistema un mejor rendimiento, para situaciones en que se tenga que trabajar con un número bastante grande de plazas.

En este módulo se puede dar procesamiento a la información que llega de los otros módulos para poder generar determinados procesos en los demás módulos, en especial en los que muestran información.

2.2.4.2 Diagrama De Bloques



2.2.4.3 Flujograma Del Programa Del Pic No 5.



2.2.5 SOFTWARE DE VISUALIZACION Y REPORTE (SVR)

2.2.5.1 Características

El software se encuentra diseñado en Visual Basic y esta conectado a una base de datos construida en Access, lo que le permite ingresar los datos tanto del identificativo del usuario que ingresa al parqueadero a través de tarjetas con códigos, como la información del tiempo de ocupación de las plazas, y almacenar esta información para luego poder generar un reporte del sistema.

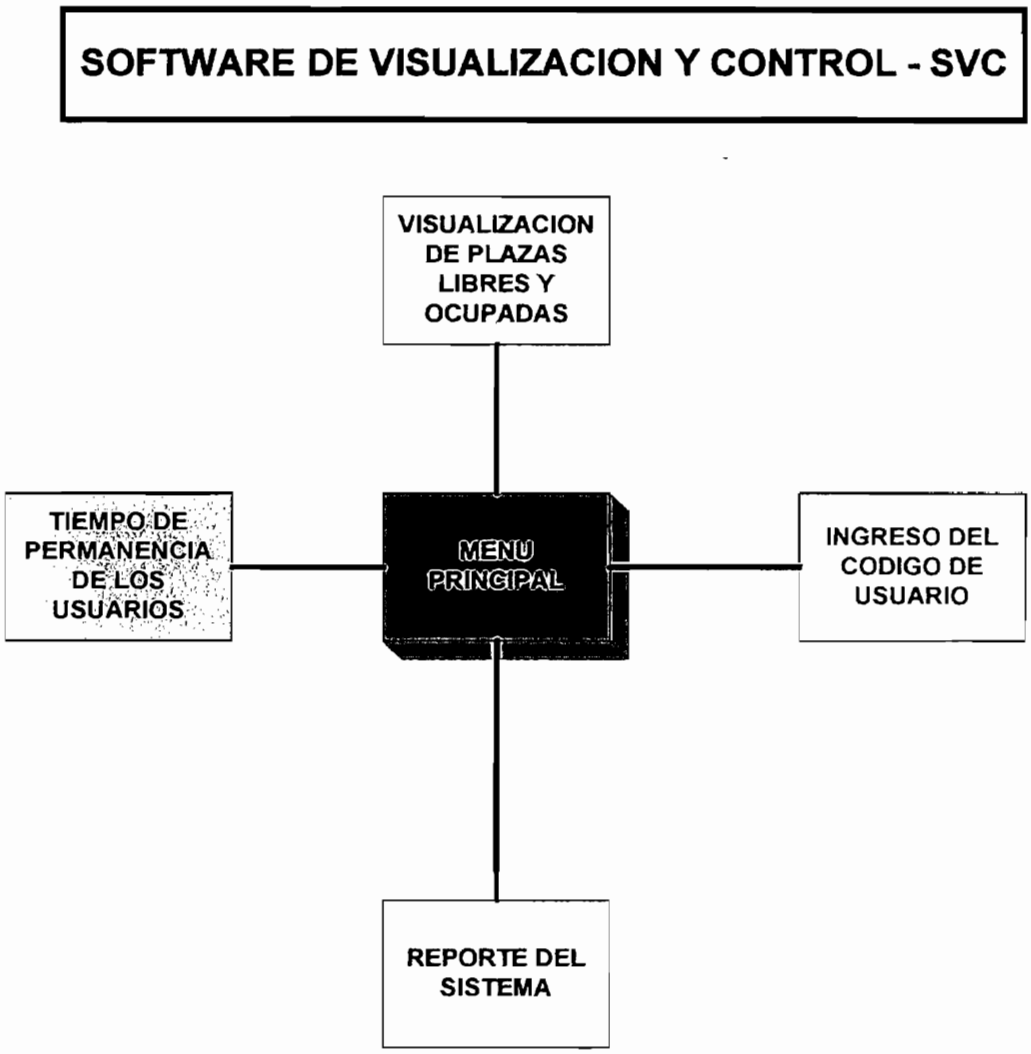
El programa consiste en varias pantallas que muestran la información que el operador del sistema desee visualizar, esta información puede ser un diagrama visual de la posición de las plazas y de su estado, es decir si se encuentra libre u ocupada, también puede requerir mirar los tiempos de permanencia de cada una de las usuarios, y además poder ingresar los datos de la diferentes tarjetas proporcionadas a los usuarios del sistema, esta seria la información habitual que el operador necesita para su trabajo.

La pantalla de visualización de las plazas, consiste en un diagrama esquemático de la posición de cada una de las plazas y además posee una visualización mediante colores del estado de ocupación, si el color es verde la plaza esta libre y si el color es rojo la plaza esta ocupada. Al insertar un nuevo módulo se debe configurar el diagrama esquemático para obtener una imagen de la posición de las nuevas plazas.

Otra de las pantallas del programa permite obtener una estadística que se genera por cada una de las plazas, donde se muestra la fecha y hora de entrada y salida del usuario que utilizó determinada tarjeta, y el costo que generó el uso del parqueadero.

También se puede visualizar la pantalla que sirve para registrar los datos de los usuarios que ingresan al sistema y poder realizar la facturación al momento de salir del parqueadero.

2.2.5.2 Diagrama De Bloques



2.2.6 DETECTOR MAGNETICO.

2.2.6.1 Generalidades.

El Detector se encuentra formado por cuatro partes, las espiras que se entierran en el pavimento o en el suelo que se destina para la detección, el cable de enlace que une la espira con su oscilador, los osciladores y el circuito lógico que está formado por un microcontrolador, que se encarga de determinar en base a la configuración del equipo si debe o no generar la señal de salida.

El detector de vehículos se basa en el sensado por inducción de la espira magnética. Es decir que al detectar un cuerpo metálico sobre la espira entrega una señal diferente a su estado de reposo.

El tamaño de la espira no es fijo, sino que se puede variar según la necesidad, ésta va instalada bajo la calzada, la variación del tamaño, posición y profundidad de la espira dependerá de las condiciones de los vehículos a detectar.

El módulo detector controlado por un microcontrolador, genera la señal de salida, que puede ser a niveles TTL , a Transistor , a Relé, y a circuitos opto aislados, lo que permite conectarse a diversos dispositivos como regulación del tráfico, gestión de semáforos, control de accesos, gestión de parking, zonas limitadas y barreras, y peajes en autopistas.

Algunos de los fabricantes de estos módulos son: Exemys, Capsys, BEA, Sic Transcore, Controles S.A., Forn Valls, Faac, Sorasata, etc.

2.2.6.2 Características Generales.

Entre los diferentes productos en el mercado, se ha seleccionado el modulo fabricado por Faac, por tener características muy adecuados para los fines del proyecto. Las características del producto se detallan a continuación:

- ✓ Microcontrolado
- ✓ Fuente switching de alimentación.
- ✓ 1 ó 2 canales.
- ✓ 4 frecuencias seleccionables.
- ✓ 4 niveles de sensibilidad seleccionable.
- ✓ Salidas opto aisladas, a relé, a transistor o salida digitales (TTL).
- ✓ Salida con seguridad ante falla.
- ✓ Diagnostico de fallas avanzado.
- ✓ Multiplexado de canales.
- ✓ Sintonía automática.
- ✓ Salida por presencia o por pulso.
- ✓ Seguimiento de fluctuaciones ambientales.

2.2.6.3 Principio de Funcionamiento.

Su principio de funcionamiento se basa en la medición de la variación de la inductancia de la espira magnética, que se produce cuando una masa metálica pasa por sobre la espira de detección.

La espira que se coloca como detector es parte de un circuito oscilador, que produce un campo magnético. Cuando una masa metálica, como puede ser un vehículo pasa sobre la espira se produce un cambio en la frecuencia del oscilador. Dicha variación es detectado por el circuito microcontrolado, que dependiendo de la configuración que se haya colocado en el equipo, determina si debe o no generar la señal de

salida, y esta activa los demás circuitos que se coloquen detrás de los módulos de detección, para el proyecto envía una señal TTL al MODEMAG, la cual es captada por el PIC # 1, y dependiendo del puerto que cambie de estado se determina que plaza se encuentra libre u ocupada.

2.2.6.4 Reset

El sistema posee un botón de reset, que se encuentra en la parte frontal, el cual permite reinicializar el dispositivo y poder configurar el equipo con los parámetros que mas se acomoden a nuestras necesidades, en base a las llaves de configuración (DIP SWITCH).
Cada vez que se varíe la configuración del sistema, mediante el cambio de posición de los DIP SWITCH, se debe presionar el botón de RESET, para que la nueva configuración del equipo entre a funcionar.

2.2.6.5 Auto sintonía.

El proceso de sintonía automática se genera cuando se pone en marcha el dispositivo, esto permite verificar el nivel base de referencia de disparo. El indicativo que el dispositivo se encuentra en el proceso de auto sintonía inicial, es que todos los leds de indicación permanecen encendidos.

Este proceso permite compensar el corrimiento de la frecuencia, que se produce debido a fluctuaciones ambientales.

2.2.6.6 Indicación De Estados.

En la parte frontal se encuentran tres leds indicadores de estado. Los leds de ambos canales se encienden cuando se detecta la presencia de un vehículo, y permanecen en este estado hasta que el vehículo se encuentra fuera del alcance de la espira detectora.

A más de indicar la presencia de un vehículo sobre la espira detectora, también proporcionan información del estado de cada uno de los canales en caso de falla.

LED CH1 Y CH2	ESTADO
1 Flash	Espira en cortocircuito
2 Flashes	Espira desconectada o frecuencia muy baja
3 Flashes	Frecuencia muy alta
Led encendido	Detección de vehículo

Tabla 2.1Indicación de estado de los leds

2.2.6.7 Sensibilidad.

Se puede seleccionar hasta 4 niveles de sensibilidad de detección por cada canal. Este factor se sensibilidad se especifica como $\delta L/L[\%]$, es decir, el mínimo cambio que se debe producir en la inductancia de la espira para que se active la salida dividido esa misma inductancia pero a lazo abierto.

Normalmente un vehículo produce un $\delta L/L = 3\%$ aproximadamente.

SENSIBILIDAD	$\delta L/L$
Baja	0.50
Media	0.10
Alta	0.05
Muy Alta	0.02

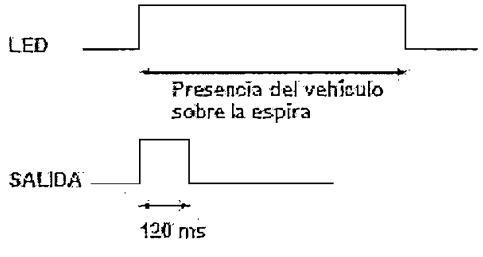
Tabla 2.2

2.2.6.8 Señal De Salida.

2.2.6.8.1 Por Pulso.

Esta característica de la señal de salida es cuando el vehículo ingresa al área de la espira, y la señal que genera el modulo principal es un pulso que dura aproximadamente 120 ms.

En la siguiente figura se muestra esta característica.



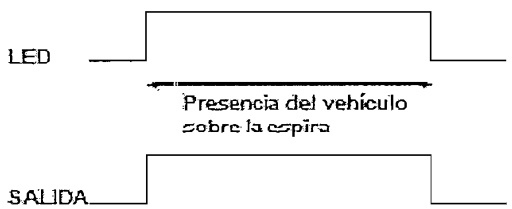
Señal de salida por pulso

Figura 2.1

2.2.6.8.2 Permanente Durante Presencia.

Mientras el vehículo se encuentra sobre la espira la señal de salida es permanente.

Esta característica se muestra en la siguiente figura.



Señal de salida permanente

Figura 2.2

2.2.6.9 Tiempo de Presencia (Pres).

Tanto para la señal de salida por pulso como para la señal de salida permanente, si el vehículo permanece sobre la espira más de un cierto tiempo, se produce una resintonización del equipo debido a un reset automático. Este tiempo, denominado tiempo de presencia, dependerá del tamaño de la masa metálica detectada. Para un automóvil, este reset se producirá luego de aproximadamente 1 hora de presencia sobre la espira.

2.2.6.10 Frecuencia (Frec).

Para cada canal se puede seleccionar una frecuencia de trabajo distinta, esto evitara interferencias con otros equipos detectores instalados en la cercanía. Entre cada frecuencia existe un cambio en la magnitud de aproximadamente un 5 %.

La geometría de la espira determina el valor de la frecuencia. El rango de frecuencias de trabajo esta entre 25 Khz. y 120 Khz. Por debajo o sobre este rango el detector generara una señal de falla. En los módulos de dos canales, se realiza la detección

de un canal a la vez, para evitar interferencia entre los mismos, es así que no existe posibilidad de interferencia mutua, puesto que, mientras el un canal detecta el otro permanece apagado. Esta conmutación no afecta al funcionamiento normal del equipo debido a la rapidez con la que actúa.

2.2.6.11 Construcción e Instalación de la Espira y Cable de Enlace.

El sistema se constituye por dos partes, la espira o bucle que es el censor y el cable de enlace que es el segmento que une la espira con el circuito microcontrolado, estos dos elementos son parte de un oscilador.

Se recomienda utilizar un bucle de 8 m de perímetro con tres vueltas, conectado mediante un cable de enlace de 50m, puesto que estas medidas cumplen un valor de inductancia de L = 120mH (bucle mas cable de enlace), para realizar las pruebas de los detectores. Para lograr un mejor rendimiento, es recomendable que el cable de enlace se a lo mas corto posible.

Es recomendable utilizar un cable de sección $1.5mm^2$ o superior, para lograr que la resistencia óhmica del bucle más cable de enlace sea inferior a los 10Ω, y así cumplir con las especificaciones de algunos fabricantes.

El punto de unión entre el bucle y el cable de enlace, es un punto sensible, y se debe tener cuidado en la instalación, para evitar futuros problemas, y esto se logra con soldadura de estaño o mediante terminales adecuados o regletas, también se debe tener cuidado con el aislamiento de este punto.

2.2.6.11.1 Cable De Enlace.

Este cable debe estar bien sujeto, para evitar cualquier vibración que pueda generar cambios en la frecuencia, y esto traducirse como una presencia de vehículo. Es

recomendable la utilización de cable blindado o apantallado, por la existencia de atenuación de posibles parásitos industriales y por darle protección mecánica ante posibles roedores.

La sección del cable deberá ser de $1.5mm^2$, y los dos conductores deberán estar trenzados a razón de unos 20 cruces por metro de longitud. Es necesario evitar la proximidad de otros cables eléctricos que alimente motores, contactores, etc. Su longitud debe ser la mínima posible y no sobrepasar los 100 m, en caso de que se tenga que superar esta distancia, se debe cuidar que:

La sensibilidad sea aceptable (la inductancia del cable disminuye la sensibilidad).
La atenuación excesiva puede bloquear el oscilador, para evitar esto debe aumentar la sección del cable, disminuyendo así las perdidas debido ala resistencia serie, que debe ser inferior a los 10Ω .

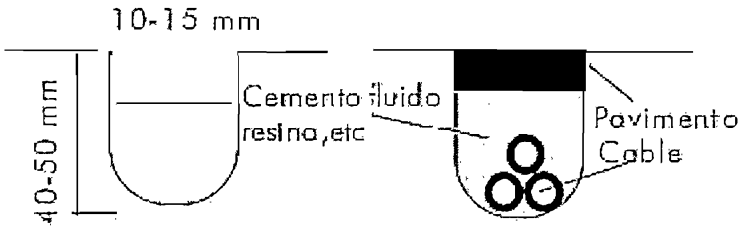
Si existen varios cables de enlace, se los debe separar entre 10 y 20 cm., excepto cuando existen dos bucles para un solo detector, en esta caso las señales van multiplexadas. Así también se debe aumentar el número de vueltas del bucle y usar cable mallado, el cual de debe conectar a tierra del lado del detector.

2.2.6.11.2 Espira O Bucle.

La canaleta para instalar el bucle debe tener las siguientes medidas:

Profundidad: 4 – 5 cm.
Anchura: 10- 1.5 cm.

Como se muestra en la siguiente figura.



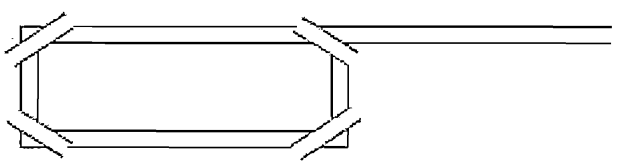
Ancho y profundidad de la canaleta

Figura 2.3

Una vez realizada la canaleta se debe limpiar con un chorro de agua (aire comprimido).

Si es necesario detectar vehículos más pequeños como bicicletas, se debe reducir la profundidad de la canaleta (3 cm.). Los lados mas cortos deben seguir la dirección del transito, los lados mas largos deberán estar a una distancia mayor a 1 m. La sección del cable debe ser $1.5mm^2$ o mayor, y es recomendable utilizar cable multifilar.

En la canaleta se debe evitar los ángulos, cortando o redondeando los vértices, como se muestra en la siguiente figura.



Redondeo de los vertices

Figura 2.4

Depositar unos 2 cm. de arena en el fondo de la canaleta, sobre esto colocar los cables del bucle y cubrir adecuadamente con cemento fluido, el cual, una vez fraguado se recubrirá con el material de superficie adecuado.

En lo posible se debe evitar los suelos metálicos y los enrejados, ya que si bien esta masa metálica será integrada por el detector, la sensibilidad se reducirá de forma proporcional a la densidad y proximidad de la parte metálica.

Si se toman las recomendaciones descritas anteriormente, el tiempo de vida o de duración del bucle, pasa a depender del aislante utilizado y de las condiciones climáticas del entorno. Con un aislante de buena calidad como el teflón se logra un tiempo de vida de bastantes años.

Las dimensiones del bucle dependen de la aplicación y del entorno. El bucle debe cubrir la mayor parte del objeto a detectar, para vehículos en movimiento la parte menos larga del bucle debe ser paralela al sentido del movimiento.

El número de vueltas dependerá de la sensibilidad que se desee lograr para cada uno de los bucles. El perímetro del bucle se distribuirá según el número de vueltas y las características del lugar de instalación.

La siguiente figura permite destacar las características de construcción del bucle y especialmente el número de vueltas, según la tabla.

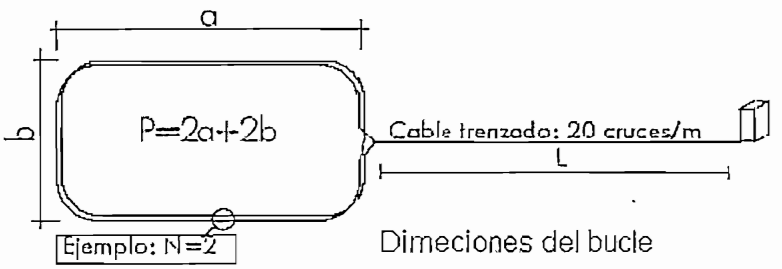


Figura 2.5

PERIMETRO BUCLE P	NUMERO DE VUELTAS N
1- 4 m	4 a 8
4 – 8 m	3 a 4
8 – 16 m	2 a 3
+ de 16 m	1 a 2

Tabla 2.3

CAPITULO 3.

3. DISEÑO DE HARDWARE Y SOFTWARE.

3.1. DISEÑO DE HARDWARE.

3.1.1 DISEÑO DEL MODULO DE DETECCION MAGNETICA – MODEMAG.

3.1.1.1 Microprocesador utilizado.

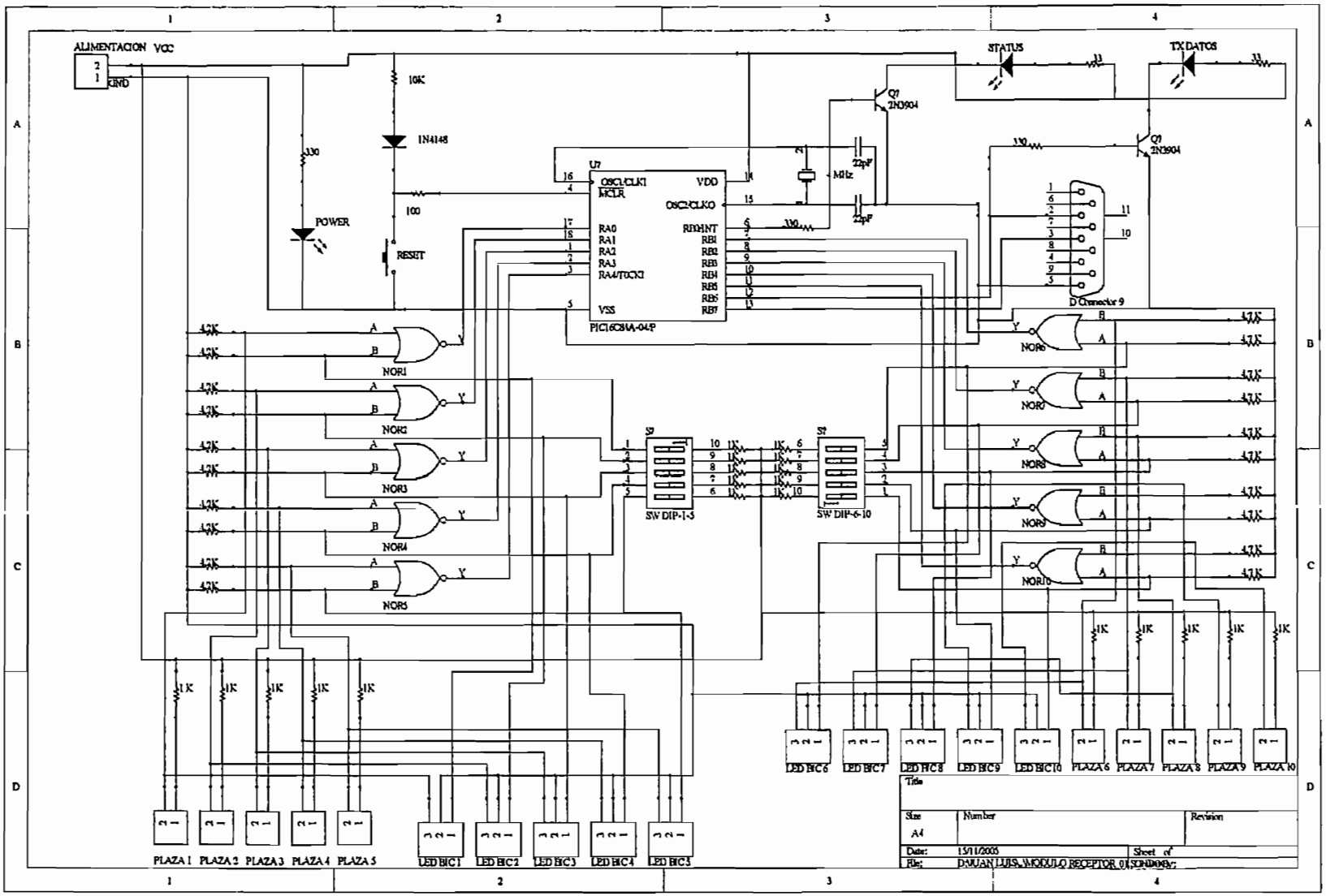
Para el diseño del MODEMAG se ha seleccionado el PIC 16F84A, por ser pequeño pero rendidor como se menciona en alguna literatura. Su tamaño pequeño de tan solo 18 pines, permite construir aplicaciones con un buen alcance y sin ocupar un gran espacio físico.

El 16F84A, permite utilizar 13 pines de sus 18 totales, para configurarlos como entradas o salidas y poder controlar aplicaciones en el medio externo. Al utilizar dos de sus pines para conectar un oscilador externo, proporciona estabilidad al sistema.

Este modelo posee algunas características importantes puesto que pertenece a la gama media, tales como temporizadores, interrupciones, memoria de programa tipo Flash, 13 E/S, etc, para mayor detalle de sus características ver Anexo.

Otra de las razones para utilizar el PIC 16F84A se debe a su compatibilidad con el Pic Basic Pro, que es un compilador en basic, el cual facilita la programación en alto nivel.

3.1.1.2 Esquemático.



3.1.1.3 Justificación del diseño.

Los puertos configurados como entradas, se encuentran unidos a una compuerta NOR de dos entradas, para darle al módulo la posibilidad de realizar una selección manual de las plazas, o permitir que se realice automáticamente mediante el Loop magnético. Las entradas de la compuerta se encuentran en 0 lógico, para obtener a la salida un 1 lógico, y así indicar que la plaza se encuentra libre, sea de forma manual o automática, estos cambios son interpretados por el programa del PIC y envía la información al MINT – PC, sobre plazas libres y ocupadas.

Tanto a la entrada A como a la entrada B de la compuerta se conectan en paralelo, el conector del loop magnético y el dip switch de la conexión manual respectivamente.

La compuerta NOR maneja la siguiente lógica:

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Tabla3.1

Esta lógica aplicada al diseño genera la siguiente tabla:

ENTRADA A (LOOP)	ENTRADA B(SWITCH)	SALIDA
Abierto(0)	Abierto(0)	1
Abierto(0)	Cerrado(1)	0
Cerrado(1)	Abierto(0)	0
Cerrado(1)	Cerrado(1)	0

Tabla 3.2

Esto indica que si tanto la parte manual que corresponde al dip switch, como la parte automática que corresponde al loop, pueden estar abiertas que el resultado será un 1 lógico, que indica que la plaza esta libre, así también si se activa solo la parte manual o solo la parte automática, el resultado es un cero lógico que indica que la plaza esta ocupada, además existe la posibilidad de que se active tanto la parte manual como la parte automática, cuyo resultado es un cero lógico que indica que la plaza esta ocupada.

El dip switch puede ser utilizado para realizar test en el prototipo y poder probar tanto los MODEMAG como el funcionamiento del programa cargado en la PC.

Otra funcionalidad de los dip swicth es poder reservar plazas, ya sea para clientes especiales, o para realizar algún mantenimiento, y así ya no permitir el ingreso de más vehículos.

Al activarse de forma accidental la parte manual de alguna plaza, generaría un numero incorrecto de plazas ocupadas, para solventar esos inconvenientes, se ha conectado un led bicolor de cátodo común, y los ánodos se encuentran conectados a cada una de las entradas de las compuertas NOR, lo que permite visualizar si se ha activado la parte manual con el color rojo, o la parte automática con el color verde, y si accidentalmente se activaron las dos a la vez obtenemos una combinación de verde y rojo, es decir que si no esta reservado por alguna situación especial, el led debería estar totalmente apagado si la aplaza esta libre.

La siguiente tabla muestra una descripción de de los diferentes estados que puede tomar el led bicolor.

LED BICOLOR		DESCRIPCION
Rojo	Verde	
apagado	apagado	Plaza libre, no se encuentra reservada manualmente
apagado	encendido	Plaza ocupada, sensado mediante el loop
encendido	apagado	.Plaza ocupada o reservada de forma manual.
encendido	encendido	Plaza ocupada, sensado mediante el loop y de forma automática (Existe la posibilidad de que se active la forma manual accidentalmente)

Tabla 3.3

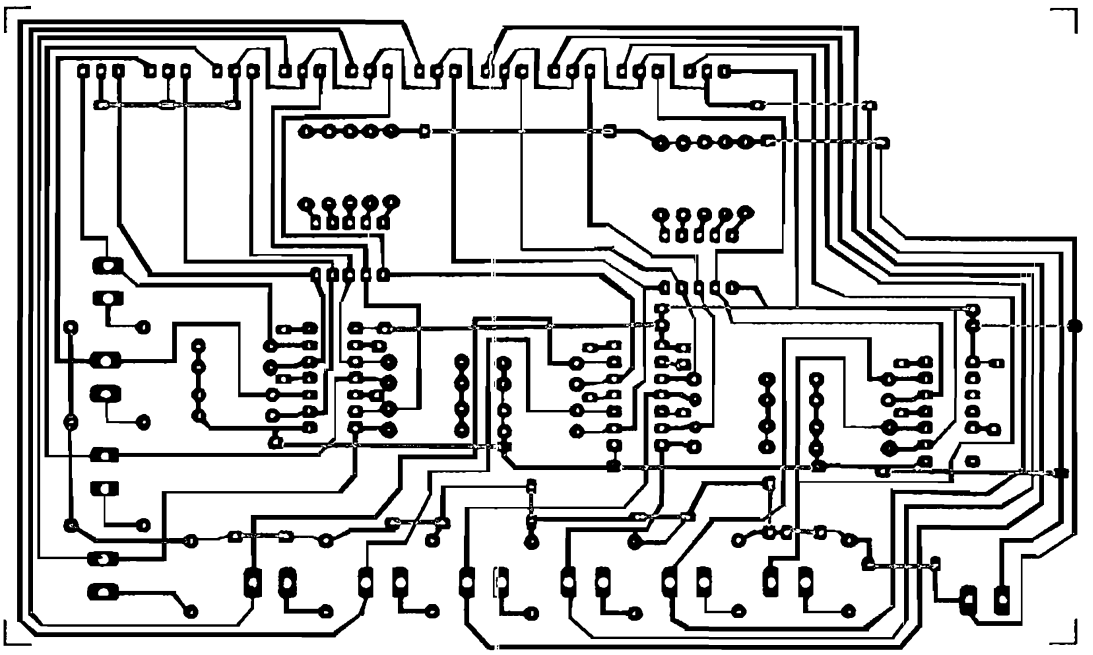
Las resistencias de 4.7 K conectadas a cada una de las entradas de las compuertas NOR, cumplen la función de realizar un divisor de voltaje y entregar 4 V al led bicolor y así poder encenderlo.

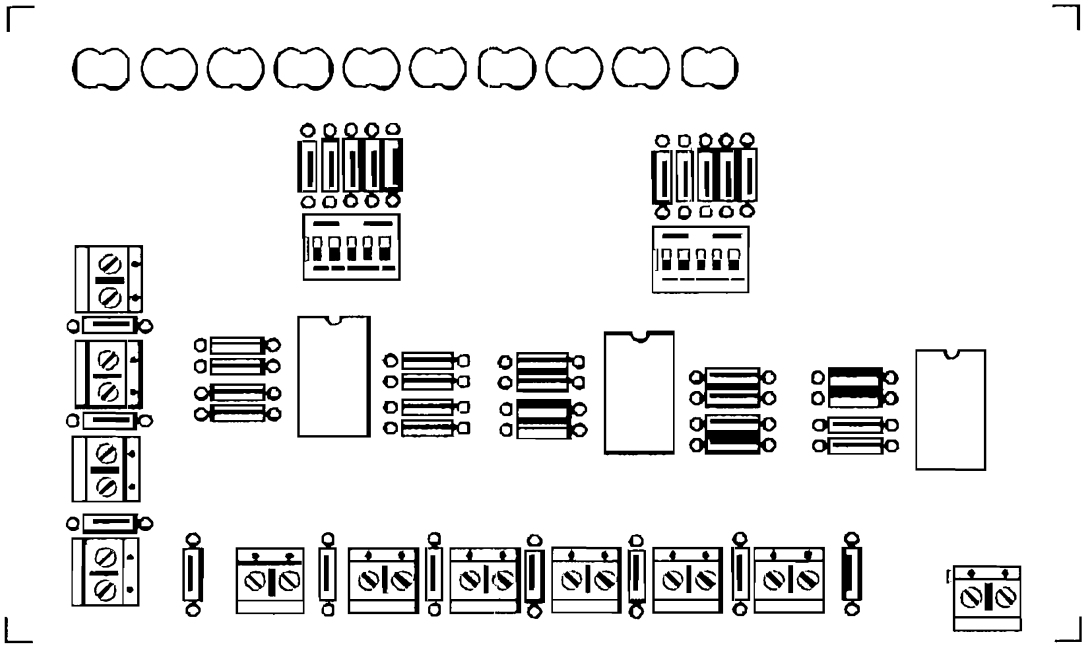
El módulo consta de tres leds indicadores, como se muestra en la siguiente tabla:

LED	ESTADO	DESCRIPCION
Power	Encendido	Módulo alimentado
	Apagado	Módulo sin energía
Tx Datos	Destellando	Módulo transmitiendo datos
	Apagado	Módulo no transmite datos
Status	Destella 5 veces	Modulo de inicia luego de un reset o corte de energía
	Destella 2 veces	Módulo en funcionamiento normal.

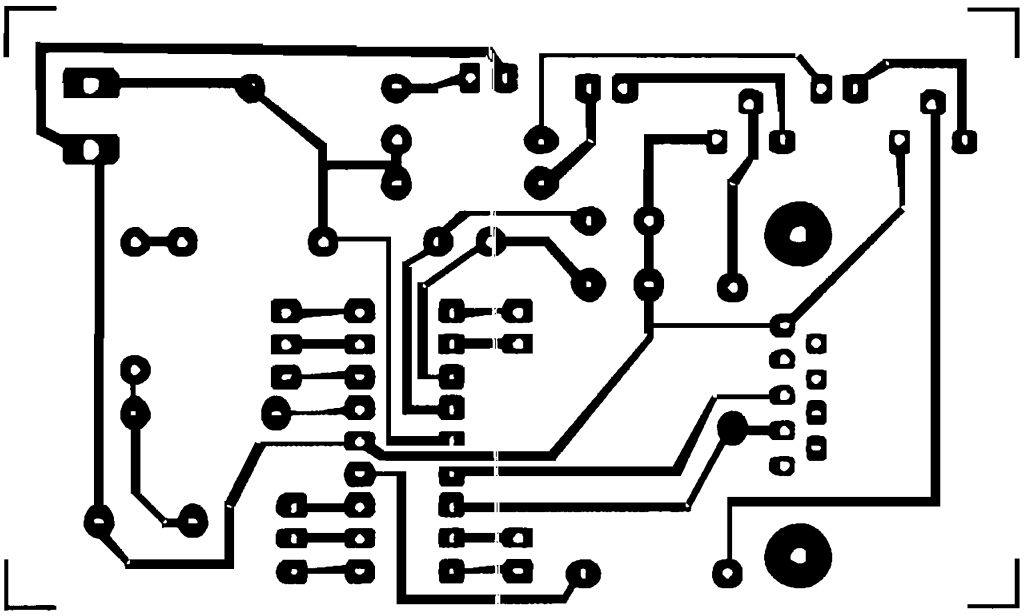
Tabla 3.4.- Leds indicadores del MODEMAG

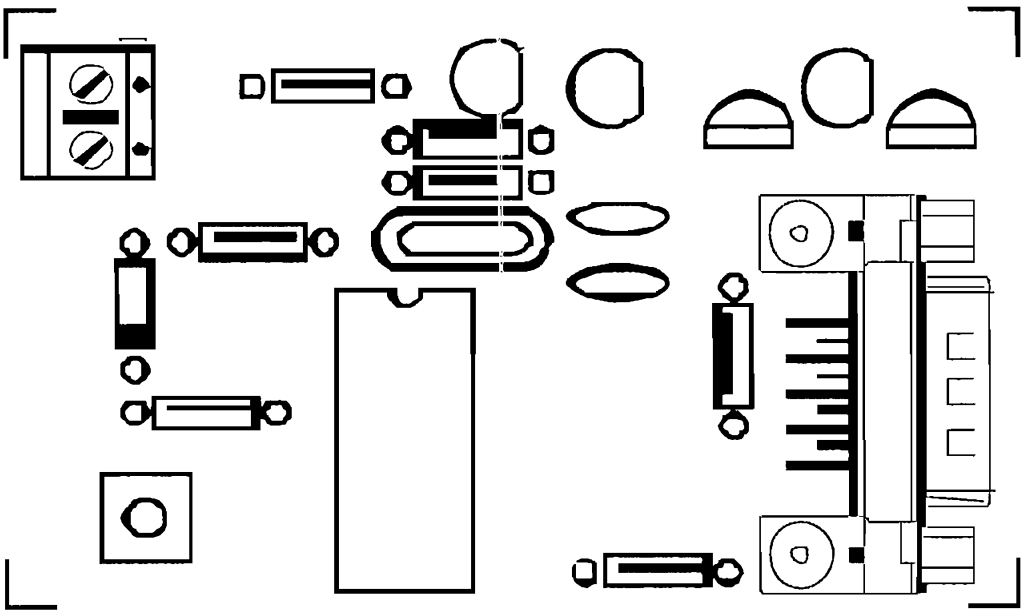
3.1.1.4 Layout (Ruteado).





Se muestra el ruteado de la placa y la distribución de los elementos de la primera placa del MODEMAG. Las líneas de color rojo y azul indican los puentes colocados en la placa





Se muestra el ruteado de la segunda placa del MODEMAG y la distribución de los elementos. Esta placa contiene el PIC, el conector hacia el MINT – PC, y los leds indicadores.

3.1.2 DISEÑO DEL PANEL MIMICO INFORMATIVO - PMI.

3.1.2.1 Microprocesador utilizado.

Se utiliza el PIC 16F877A, por su tamaño y el número de entradas/salidas disponibles, lo que permite manejar las matrices de leds, el RTC, el LCD y los leds indicadores de las plazas y dispositivos I2C como el reloj calendario.

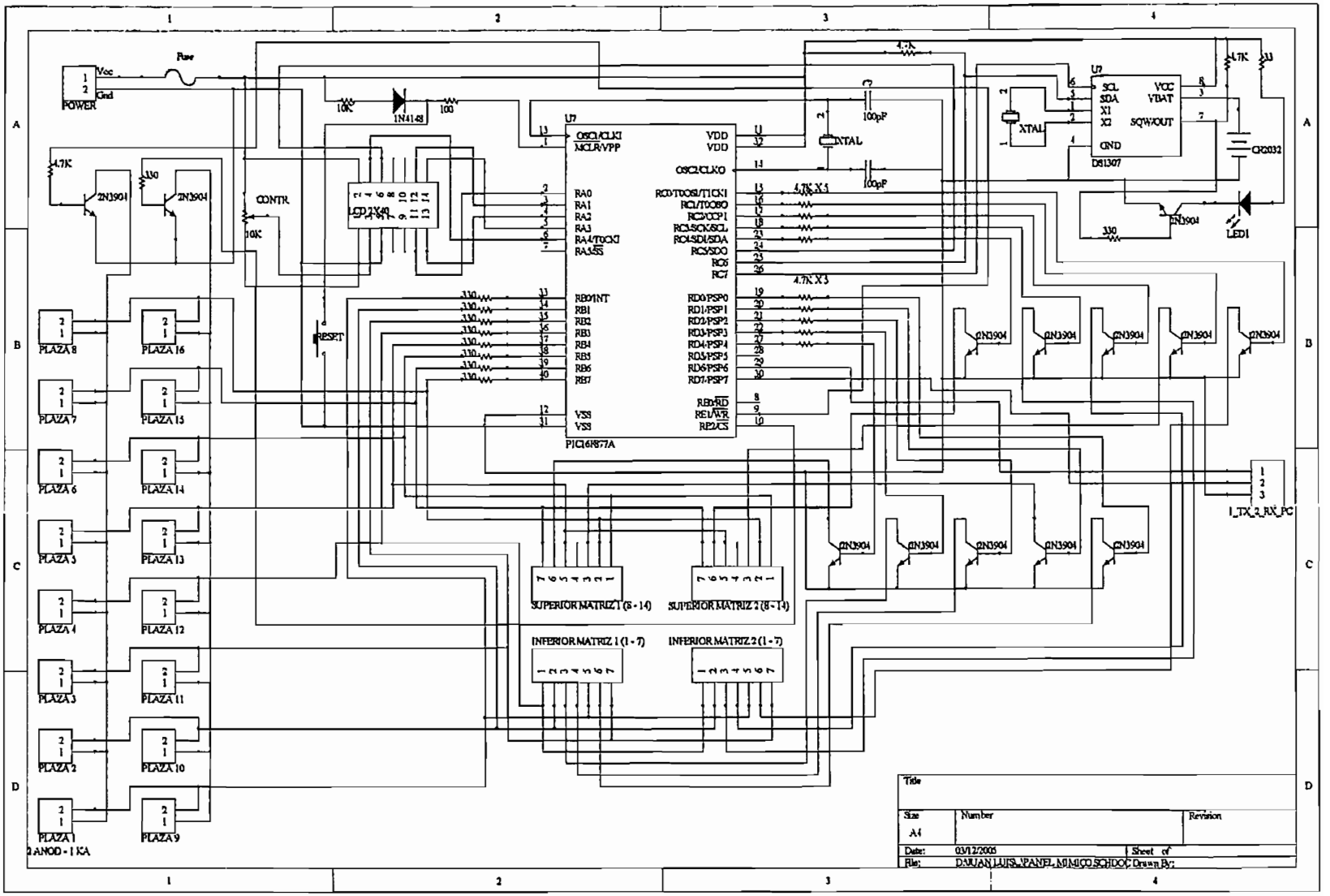
El PIC 16F877A posee una capacidad de 8192 palabras de programación, lo que combinado con el gran número de entradas / salidas, lo convierte en un dispositivo con muchas prestaciones para almacenar una gran cantidad de programa y comunicarse con el medio externo con un gran número de dispositivos.

Posee ocho pines con conversores análogo / digitales, lo que facilita la utilización de dispositivos que entreguen información analógica, como son los sensores y sin dispositivos adicionales poder generar una señal digital y utilizarla para diferentes procesos dentro del PIC y generar resultados al medio externo.

La compatibilidad entre el compilador PBP y el PIC 16F877A, permiten realizar una programación de mayor nivel, logrando programas más manejables para el programador y obteniendo resultados más rápidos, consiguiendo una optimización de tiempo.

Al conocer la forma de trabajo y programación del PIC 16F84A que es un dispositivo relativamente de menor capacidad, lo que implica un menor número de instrucciones y de registros de programación, facilitan al programador un aprendizaje en un menor tiempo, y la familiaridad con el PIC 16F877A, permite que los conocimientos sean aplicados a un PIC de mayor tamaño y que genera un mayor número de prestaciones.

3.1.2.2 Esquemático.



3.1.2.3 Justificación del Diseño.

En este módulo se disponen de dos matrices de 8 x 5, las cuales se las maneja conectando las ocho filas a todo un puerto del PIC que en este caso es el puerto B, y las cinco columnas se conectan a diferentes puertos, para que mediante la multiplexación de las columnas se puedan manejar las diferentes matrices, es así que el PIC podría manejar hasta un total de seis matrices, cabe indicar que para el encendido de las matrices no se debe sobrepasar un tiempo de 20 ms, para que no se note que las matrices se van prendiendo en diferente tiempo.

Para el manejo de más de una matriz se utiliza el principio de la multiplexación, es decir que los puertos de las filas son los mismos para todas las matrices, pero se va variando los puertos de las columnas y así lograr encender una matriz a la vez.

Es necesario colocar transistores en cada una de las columnas para que este cumpla la función de un switch al recibir una señal de los puertos del PIC, y poder generar la suficiente corriente para un encendido óptimo de cada uno de los elementos de la matriz.

Es así que se han colocado en cada una de las columnas transistores 2N3904 logrando una buena iluminación de la matriz, cabe indicar que se pueden utilizar transistores TIP 110 que generan una mayor corriente, aunque el costo se elevaría.

Para el manejo del LCD es necesario cambiar la configuración predeterminada por el compilador PBP, en vista de que se utilizo el puerto B.3 para el manejo de las matrices. Por ello se utilizó el puerto C.5 para manejar el ENABLE de la LCD, para realizar este cambio se debe utilizar la declaración **DEFINE** en conjunto con los diferentes registros de manejo del LCD.

El cambio se realiza mediante software, razón por la cual estas varias variaciones se estudiarán en el diseño de software. El contraste del LCD se maneja mediante un potenciómetro.

Con el objeto de que el módulo PMI no dependa de la computadora central, se ha instalado el RTC (DS1307) que entrega al PIC la información de la hora y fecha la misma que se mostrará en el LCD. Es necesario indicar que para el correcto funcionamiento del RTC se debe recurrir a la hoja de datos y configurar correctamente los diferentes pines; se debe cuidar el tipo de cristal a instalar, ya que el fabricante recomienda para el correcto funcionamiento un cristal de 32.768 Hz.

Tomando en cuenta que este dispositivo se comunica serialmente utilizando el protocolo I2C, es necesario colocar resistencias de Pull Up tanto en la línea SDA que es la línea de datos, como en la línea SCL que corresponde al reloj.

En el diseño se ha omitido la resistencia de Pull Up de la línea SCL, en vista de que mediante software se puede configurar para que no sea necesaria, esto se logra mediante la declaración **DEFINE** I2C_SCLOUT 1, esta declaración se explicará en la parte de software.

Para darle confiabilidad al reloj del PMI se ha colocado una batería CR2032 que sirve de respaldo para cuando la energía del módulo se desconecte o se sufra de interrupciones eléctricas.

Para la señalización de las diferentes plazas dentro del PMI se utiliza LEDS, que se manejan como un reflejo de los leds que se instalan en cada MODEMAG.

3.1.3 DISEÑO DEL MODULO DE CONTROL – MODCON.

3.1.3.1 Microcontrolador Utilizado.

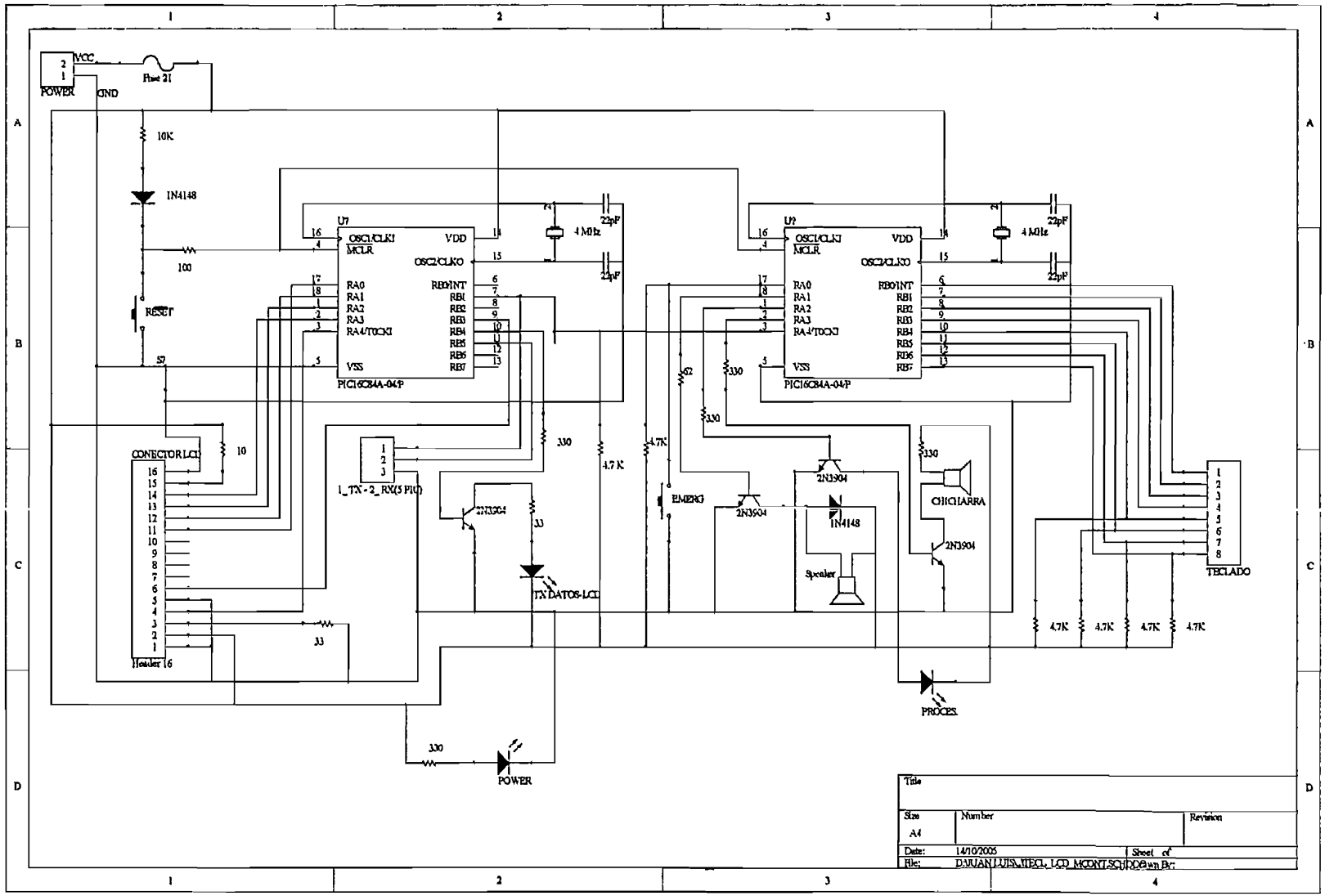
Se ha utilizado el PIC 16F84A, por ser un PIC pequeño pero con grandes prestaciones. En este caso para dar la característica de modularidad se conectan serialmente dos microcontroladores, lo que permite manejar individualmente las dos partes principales de este módulo, la primera parte que controla el teclado y las alarmas, y la segunda parte que controla el LCD, que constituye la interfaz con el usuario.

El PIC 16F84A posee el número necesario de E/S, para manejar cada una de las partes principales del módulo, como son el teclado y el LCD y los demás servicios adicionales que posee como son alarmas y leds indicadores.

El compilador Pic Basic Pro posee declaraciones como LDCIN o LDCOUT, que unido con las conexiones adecuadas entre el LCD y el PIC, permiten un manejo de la información de entrada y salida al LCD de una forma fácil y ágil con tan solo colocar la declaración, los parámetros de posición y la información.

El PIC 16F84A esta diseñado para este tipo de declaraciones lo que permite una mayor facilidad en la programación y manejo de la información a través del LCD.

3.1.3.2 Esquemático.



3.1.3.3 Justificación del Diseño.

Para el manejo del teclado es necesario colocar resistencias de Pull UP en las columnas, excepto cuando el teclado posee resistencias de Pull Up internas. Los valores de estas resistencias se basan en la cantidad de corriente que soportan los puertos del PIC, que es de 25 mA. Es decir:

$$V = RI$$

$$R = \frac{V}{I}$$

$$R = \frac{5V}{0.025A}$$

$$R = 200\Omega \approx 220\Omega$$

La resistencia limite seria de 220Ω, lo que implicaría trabajar con el peligro de dañar el puerto, se maneja un rango de $1K\Omega$ a $10K\Omega$, es por esta razón que los valores mas utilizados son las resistencias de $4.7K\Omega$, que permiten trabajar en un rango intermedio.

Se ha colocado una chicharra, y un led indicador junto al PIC que maneja el teclado para dar al usuario un indicativo sonoro y otro luminoso al momento de manipular el teclado, lo que permite saber que se ha presionado una tecla o se ha producido algún proceso en el modulo como puede ser un reset, el ingreso al menú de mensajes, activación de una alarma o el cambio de clave.

La generación de la alarma se realiza por medio de un pulsador de llave, lo que permite darle mayor seguridad al módulo, el cerrar el pulsador se realiza un cambio de 1 lógico a 0 lógico, y esto detiene el funcionamiento normal del módulo y provoca

la salida de tonos por un puerto del PIC; en este puerto se realiza un amplificación con un transistor 2N3904 y así tener una salida adecuada apara la colocación de un parlante.

A diferencia de las E/S del puerto B, en el puerto A es necesario colocar resistencias de Pull Up, para que el pin funcione como entrada o salida, es así que el pin conectado al pulsador de emergencia, y el pin que sirve para la comunicación serial con el otro PIC se encuentran conectados a resistencia de Pull Up, cuyo valor se especifica en base a la corriente que soporta el puerto.

En la parte del módulo que maneja el teclado, se encuentran tres led, cuya funcionalidad es la siguiente:

LED ROJO: o led de power, indica que el modulo se encuentra energizado.
LED AMARILLO: indica que se ha pulsado alguna o algunas teclas.
LED VERDE: indica que se están enviando datos al PIC que maneja el LCD o directamente al MINT – PC.

En el PIC que maneja el LCD es necesario cumplir con la configuración que se muestra en la siguiente figura, para que las instrucciones como **LCDOUT** o LCDIN funciones sin problema. En caso de ser necesario que se altera la configuración predeterminado por el compilador, se debe usar la declaración **DEFINE** para alterar los registros de los bits de datos, R/W y el Enable, estos cambios se explicaran en el diseño del software.

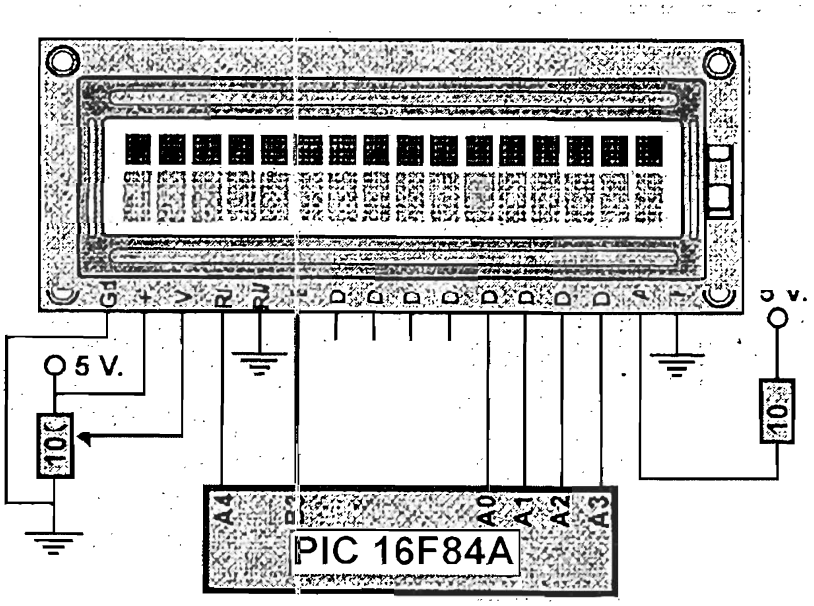


Figura 3.1

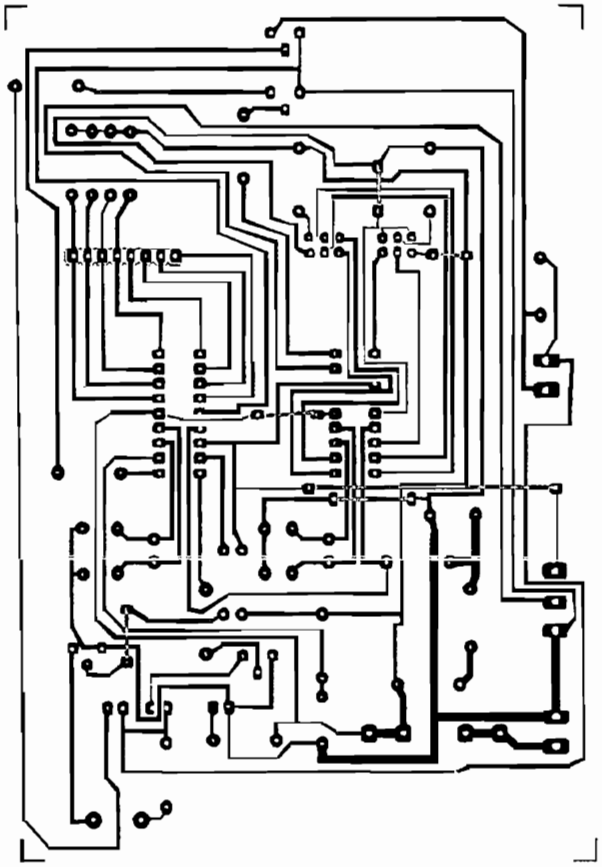
Conexión de un LCD, predefinido por el compilador PBP

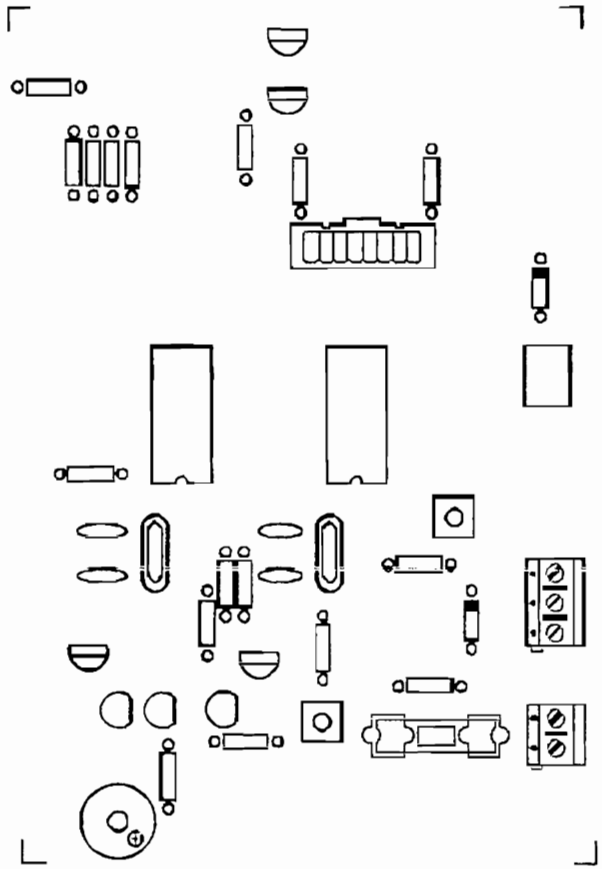
La conexión anterior es para trabajar a cuatro bits , la resistencia de 10Ω conectada a la alimentación del back Light, sirve para evitar las altas temperaturas, además el bit R/W se encuentra conectado a tierra, lo que permite escribir en el LCD, mediante la instrucción **LCDOUT**.

Es necesario que el cristal del PIC sea de 4 MHz para que las instrucciones, en este caso que manejan la información hacia el LCD, funcionen adecuadamente.

En esta parte del módulo se dispone de un led de color verde que indica la transmisión de datos hacia el LCD.

3.1.3.4 Layout (Ruteado).





Se muestra el ruteado de la placa y la distribución de los elementos. Las líneas azules indican los diferentes puentes utilizados en la construcción de la placa.

3.1.4 DISEÑO DEL MODULO DE INTERFAZ CON LA PC – MINT – PC.

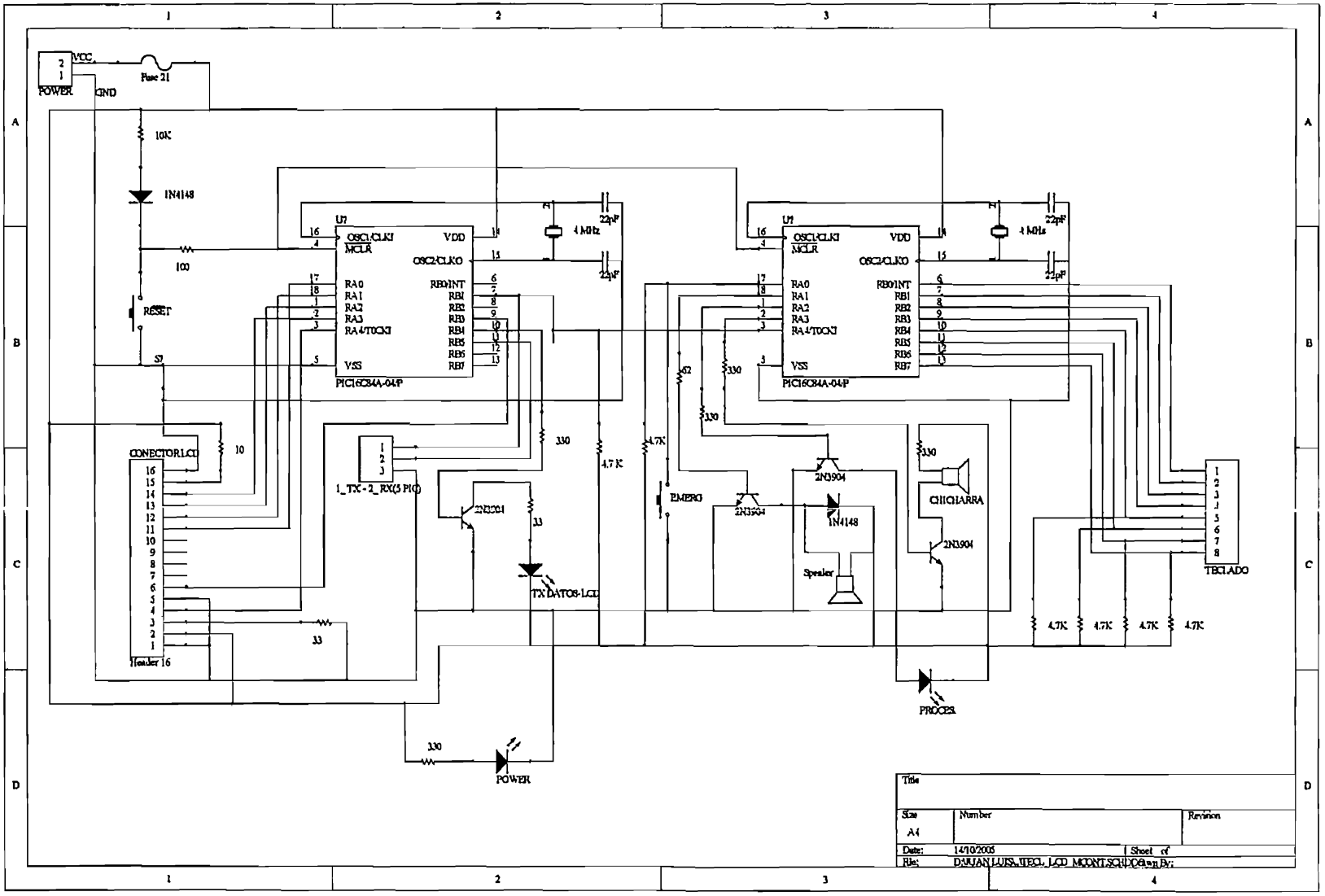
3.1.4.1 Microprocesador Utilizado.

Se ha utilizado el PIC 16F84A, por disponer de un número de E/S suficientes para manejar la comunicación con los diferentes módulos del PRINMOP.

Gracias a su tamaño reducido permite diseñar este módulo de una forma compacta, lo que facilita la instalación y cambio de ser necesario en caso de algún fallo del módulo.

El PIC 16F84A permite la comunicación serial con los demás módulos de una forma sencilla, en este caso la mayoría de elementos como los MODEMAG, MODCON y el PMI, se encuentra relativamente cerca del MINT – PC, lo que permite que la comunicación serial de los de los diferentes módulos se realice incluso sin la ocupación de dispositivos como el MAX 232, que permiten la comunicación del estándar RS232 con la PC a mayores distancias; esto se debe a la compatibilidad del compilador PBP con el PIC 16F84A, puesto que existen declaraciones como **SEROUT** puerto,N2400,["información"], que facilitan la comunicación omitiendo dispositivos de interfaz entre la salidas del PIC que generalmente entrega niveles TTL y la entrada al PC que poseen niveles RSR 232, cabe indicar que el compilador PBP también posee instrucciones como **SEROUT** puerto,T2400,["información"], que permiten utilizar el max 232 entre el PIC y la PC, proporcionando una mayor distancia de comunicación entre dispositivos.

3.1.4.2 Esquemático.



3.1.4.3 Justificación del Diseño.

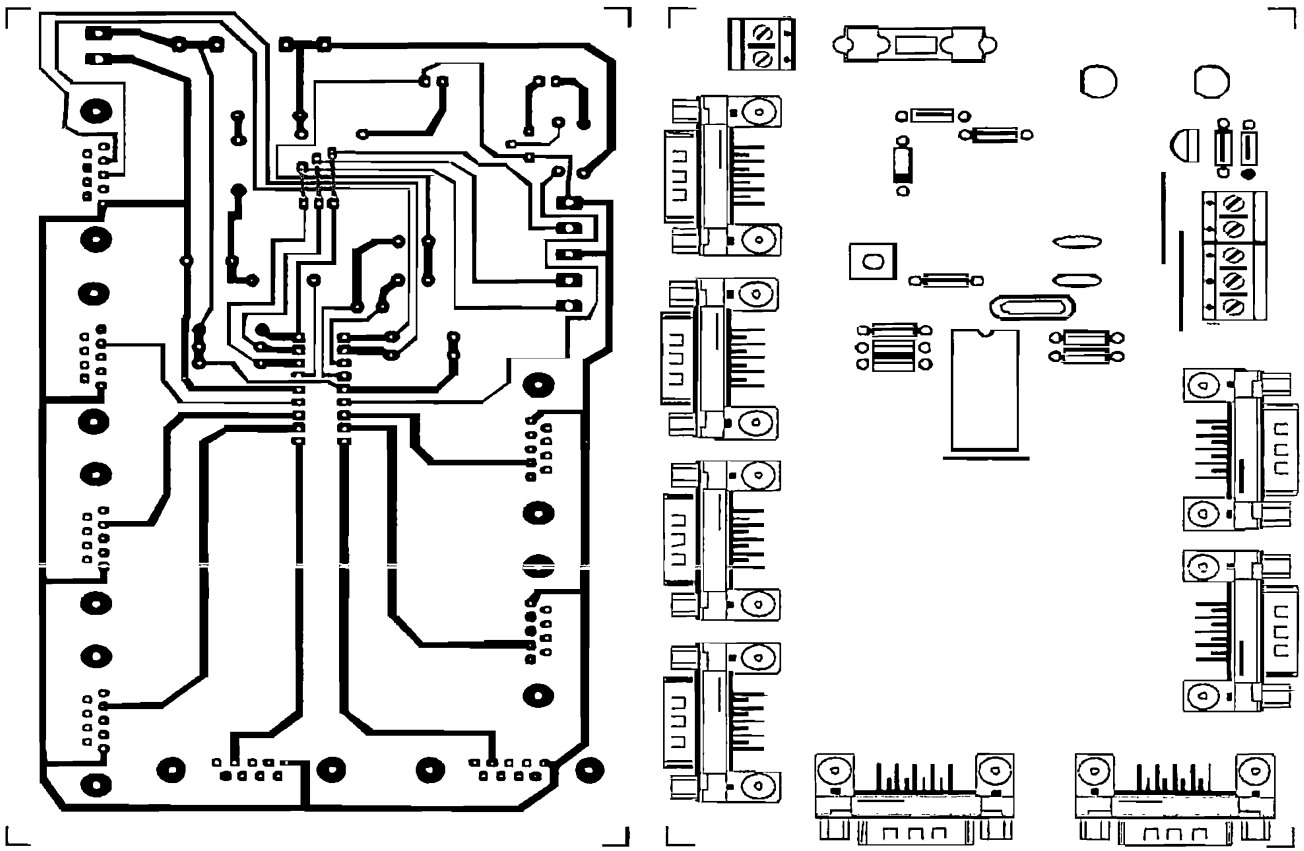
El puerto A se ha distribuido de la siguiente manera:

Puerto A.0 enciende el led indicador
Puerto A.1 recibe los datos del MODCON
Puerto A.2 recibe del panel mímico
Puerto A.3 transmite hacia el PMI – I2C
Puerto A.4 Transmite la información hacia el PC.

Todo el puerto B se encuentra diseñado para recibir la información de los 7 MODEMAG que se pueden conectar al MINT – PC.

Posee un led de status, que nos ayuda a identificar el funcionamiento normal del módulo, mediante un destello que nos indica que el programa internamente en el PIC se encuentra corriendo y tomando datos de los diferentes módulos y enviándolos hacia la PC. Además posee un LED de power que nos indica que el módulo está energizado. Para facilitar el diseño se ha tomado todo el puerto B para la conexión con los MODEMAG, en cambio para la comunicación con el MINT – PC se ha tomado dos pines del puerto A, uno para transmisión y otro para recepción; para la comunicación para el MODCON se ha tomado un solo pin del puerto A que se configura como recepción, puesto que solo nos interesa que el módulo nos entregue datos; para la comunicación con el PMI se utiliza los dos pines sobrantes del puerto A, uno para recepción y otro para transmisión, en vista de que podemos obtener datos del módulo, y necesitamos enviar información al LCD.

3.1.4.4 Layout (Ruteado).

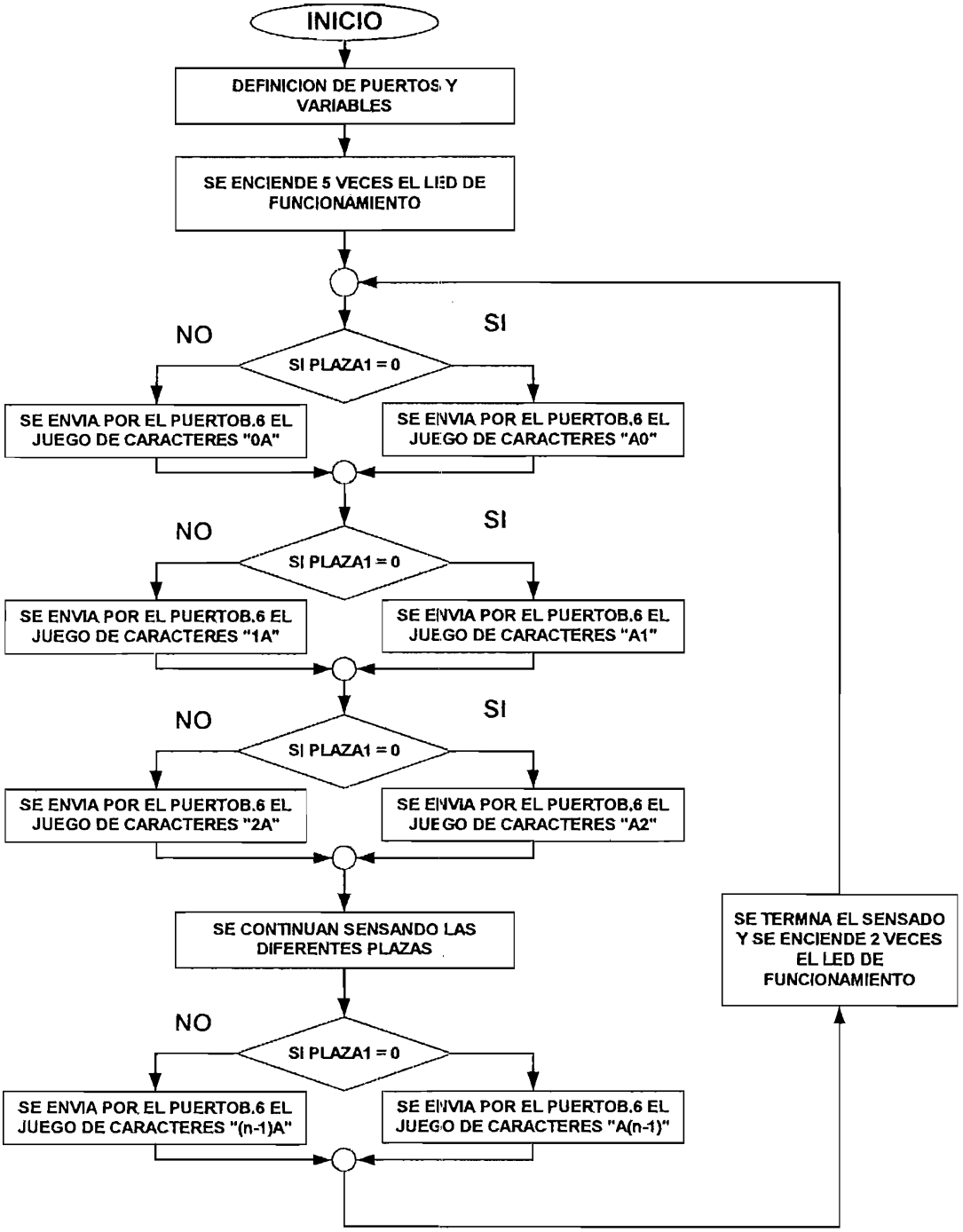


Se muestra el ruteado de la placa y la distribución de los elementos del MINT PC. Las líneas azules indican los diferentes puentes utilizados en al construcción de las placa.

3.2 DISEÑO DEL SOFTWARE

3.2.1 MODULO DE DETECCION MAGNETICA - MODEMAG.

3.2.1.1 Diagrama de Flujo



3.2.1.2 Subrutinas Importantes.

3.2.1.2.1 Modedefs.bas

INCLUDE "modedefs.bas"; librerías que contienen los modos de comunicación

SE deben incluir esta línea para poder utilizar la declaración **SEROUT**, que sirve para enviar datos seriales asincrónicos usando 8 bits de dato, sin paridad y 1 bit de stop (8N1).

Esto significa incluir el programa modedefs.bas. El comando **INCLUDE** se puede utilizar para insertar nuestros propios programas.

3.2.1.2.2 Etiquetado de los puertos.

```
plaza1 var porta.0    ;se renombra cada puerto con una variable
plaza2 var porta.1    ; por ejemplo, el puerto A.0 se llamará plaza1
plaza3 var porta.2
plaza4 var porta.3
plaza5 var porta.4
plaza6 var portb.1
plaza7 var portb.2
plaza8 var portb.3
plaza9 var portb.4
plaza10 var portb.5
```

Se redefinen los puertos del PIC conectados a los loops magnéticos, para facilitar el manejo dentro del programa, es así que el puerto A.0 se llamará plaza1, el puerto A.1 se llamará plaza2, y así sucesivamente.

3.2.1.2.3 Condición para cada puerto

```
IF plaza1=0 then          ; si cumple la condición va a envioA
    CALL envioA           ;se llama a la subrutina envio A
else                      ; si no cumple la condición del IF, va a envio1
    CALL envio1           ; se llama a la subrutina envio1
endif                     ; se finaliza el IF
```

A cada puerto conectado a un loop magnético, se le inserta una condición, puesto que estos puertos están recibiendo un 1 lógico de la salida de la compuerta, mientras no se active la plaza de forma manual o automática, es así que la condición indica que si existe un cambio a 0 lógico se llame a la subrutina enviaA0, caso contrario de no cumplirse la condición, se llama a la subrutina envio0A, y se finaliza la condición.

3.2.1.2.4 Subrutinas de Envio

```
envioA0:                          ; subrutina de envio si plaza 1 se ocupa
  SEROUT portb.6,N2400,["A0"]     ; se envía los caracteres A0 por el puerto b.6
  PAUSE 100                       ; se realiza una pausa
  RETURN                          ; regresa a la parte del programa donde se
                                  ;realizó la llamada a la subrutina


envio0A:                          ;subrutina de envio si plaza 1 esta libre
  SEROUT portb.6,N2400,["0A"]     ; se envía los caracteres 0A por el puerto b.6
  PAUSE 100
  RETURN                          ; regresa a la parte del programa donde se
                                  ;realizó la llamada a la subrutina
```

La subrutina envioA0, saca por el puerto B.6, el juego de caracteres A0, cuando la plaza 1 del MODEMAG A se encuentra ocupada, luego de enviar los caracteres serialmente, retorna para seguir sensando los demás puertos del módulo, lo mismo ocurre con la subrutina envio0A, con la variación que se invierte el orden de los caracteres, es decir envía 0A, para indicar que la plaza esta libre.

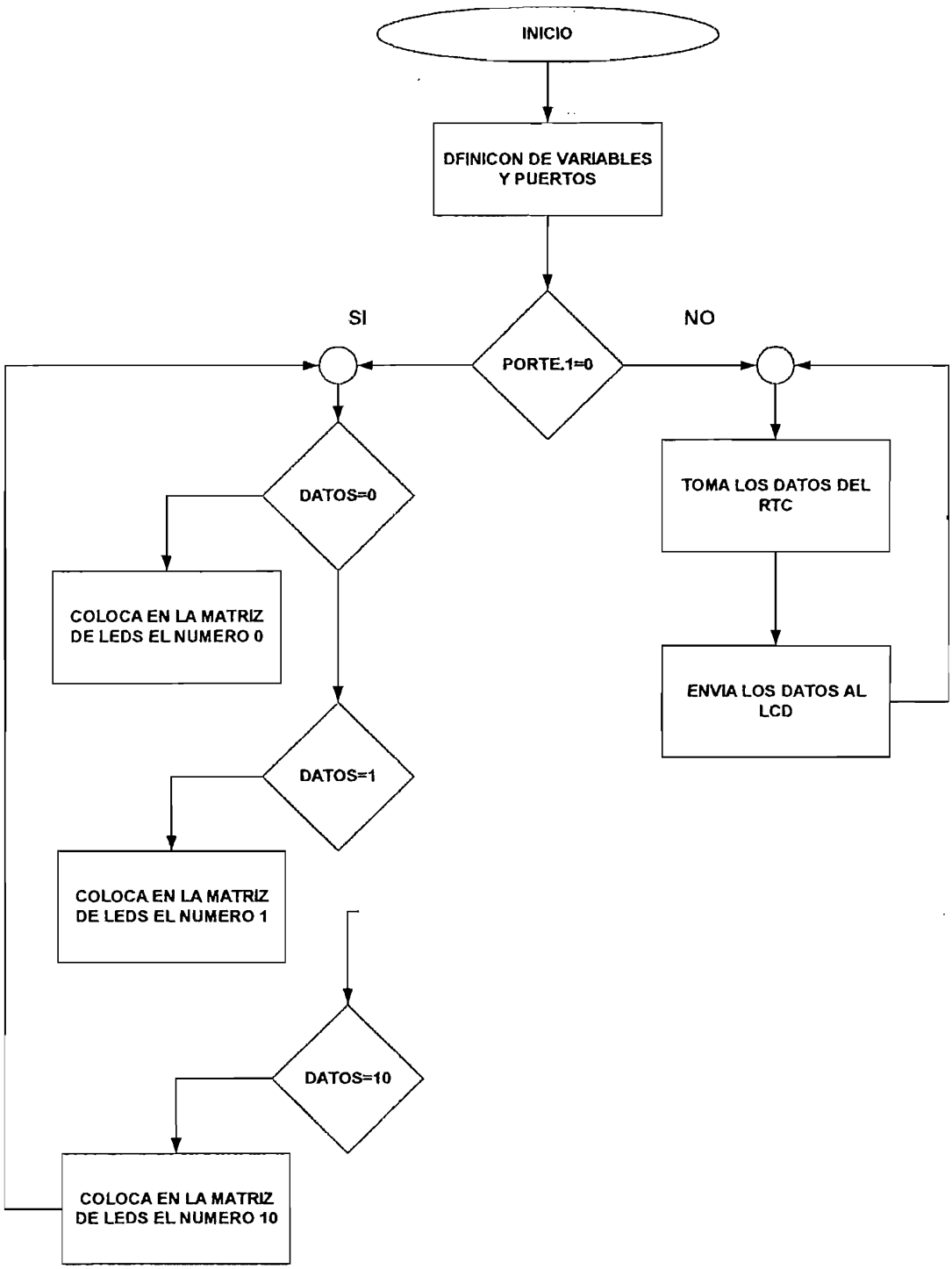
Cada MODEMAG en los caracteres que envía, se encuentra el estado de las plazas y su identificación propia, es así que para dos diferentes MODEMAG tenemos la siguiente nomenclatura de los caracteres a enviarse.

MODEMAG 1			MODEMGA 2		
PLAZA	LIBRE	OCUPADA	PLAZA	LIBRE	OCUPADA
1	0A	A0	1	0B	B0
2	1A	A1	2	1B	B1
3	2A	A2	3	2B	B2
4	3A	A3	4	3B	B3
5	4A	A4	5	4B	B4
6	5A	A5	6	5B	B5
7	6A	A6	7	6B	B6
8	7A	A7	8	7B	B7
9	8A	A8	9	8B	B8
10	9A	A9	10	9B	B9

Tabla 3.5

3.2.2 PANEL MIMICO INFORMATIVO - PMI

3.2.2.1 Diagrama de Flujo.



3.2.2.2 Subrutinas Importantes.

3.2.2.2.1 Reasignación Pin Enable LCD.

```
DEFINE LCD_EREG PORTC        ; define pin para conectar el bit Enable
DEFINE LCD_EBIT 5            ; en el puerto C5
```

En esta parte usando al declaración **DEFINE** se modifica el registro que maneja el Enable del LCD, en vista de que puerto B.3 que es el puerto por defecto que asigna el PBP se utilizó para el manejo de las matrices de leds. Primero se indica el puerto, en este caso es el C, y luego se asigna un pin.

3.2.2.2.2 Toma de Información del Puerto de Datos.

```
SERIN porte.1,N2400,datos

inicio2:

IF datos = 0 THEN numero0
IF datos = 1 THEN numero1
IF datos = 2 THEN numero2
IF datos = 3 THEN numero3
IF datos = 4 THEN numero4
IF datos = 5 THEN numero5
IF datos = 6 THEN numero6
IF datos = 7 THEN numero7
IF datos = 8 THEN numero8
```

```
IF datos = 9 THEN
   numero9
ELSE
   reloj
ENDIF
GOTO inicio4
```

Se toma los datos del puerto E.1 y dependiendo de la información que se almacene en la variable datos, el programa se direcciona a la subrutina adecuada para enviar la información hacia al matriz de leds,

3.2.2.2.3 Envio de información a las matrices.

```
numero0:
FOR x=1 TO 40                                   ; subrutina genera cero - cero
portd=%00000001:portb=%11111111:PAUSE 2
portd=%00000010:portb=%10000001:PAUSE 2
portd=%00000100:portb=%10000001:PAUSE 2
portd=%00001000:portb=%10000001:PAUSE 2
portd=%00010000:portb=%11111111:PAUSE 2
portd=%00000000:
portc=%00000001:portb=%11111111:PAUSE 2
portc=%00000010:portb=%10000001:PAUSE 2
portc=%00000100:portb=%10000001:PAUSE 2
portc=%00001000:portb=%10000001:PAUSE 2
portc=%00010000:portb=%11111111:PAUSE 2
portc=%00000000
NEXT

GOTO inicio2
```

En esta subrutina se maneja las ocho filas, median te el puerto B, y las columnas de las matrices, mediante el puerto C y D,. Se activan las columnas de la matriz que se requiera y se realiza el barrido de las filas y así se logra mostrar el carácter que se desee.

3.2.2.2.4 Asignación de líneas I2C.

```
DEFINE I2C_SCLOUT 1          ; para que no necesite resistencia pull up en SCL
CPIN VAR Portc.7             ; pin señal de reloj I2C
DPIN VAR PORTC.6             ; pin de datos I2C
```

Mediante I2C_SCLOUT 1, se evita colocar resistencias de Pull UP en la línea de reloj. Y se asigna el puerto C.7 y C.8, para reloj y datos respectivamente.

3.2.2.2.5 Lectura del RTC.

```
inicio3:

   I2CREAD DPIN,CPIN,%11010000,0,[segu]
   I2CREAD DPIN,CPIN,%11010000,1,[minu]
   I2CREAD DPIN,CPIN,%11010000,2,[hora]
   I2CREAD DPIN,CPIN,%11010000,3,[diaS]
   I2CREAD DPIN,CPIN,%11010000,4,[diaF]
   I2CREAD DPIN,CPIN,%11010000,5,[mes]
   I2CREAD DPIN,CPIN,%11010000,6,[anio]

LCDOUT $FE,1,hex2 hora,":",hex2 minu,":",hex2 segu
```

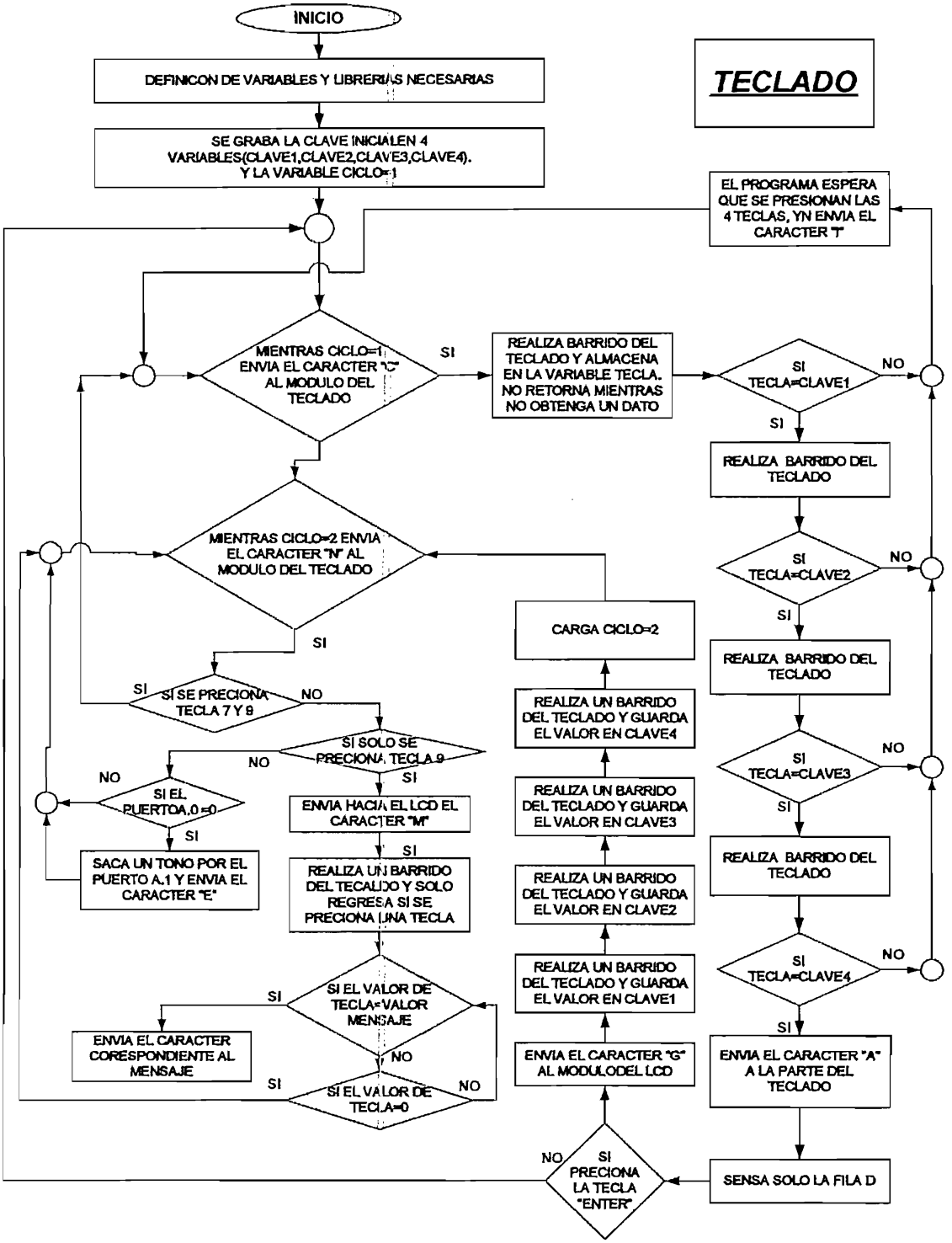
Se utiliza la declaración **I2CREAD,** para leer las diferentes secciones del RTC, se coloca la declaración, el pin de datos, el pin de reloj, la dirección del dispositivo I2C, en este caso es %11010000, cuyos cuatro bits menos significativos son dados por el fabricante, y al final se indica la variable donde se almacena la información leída.

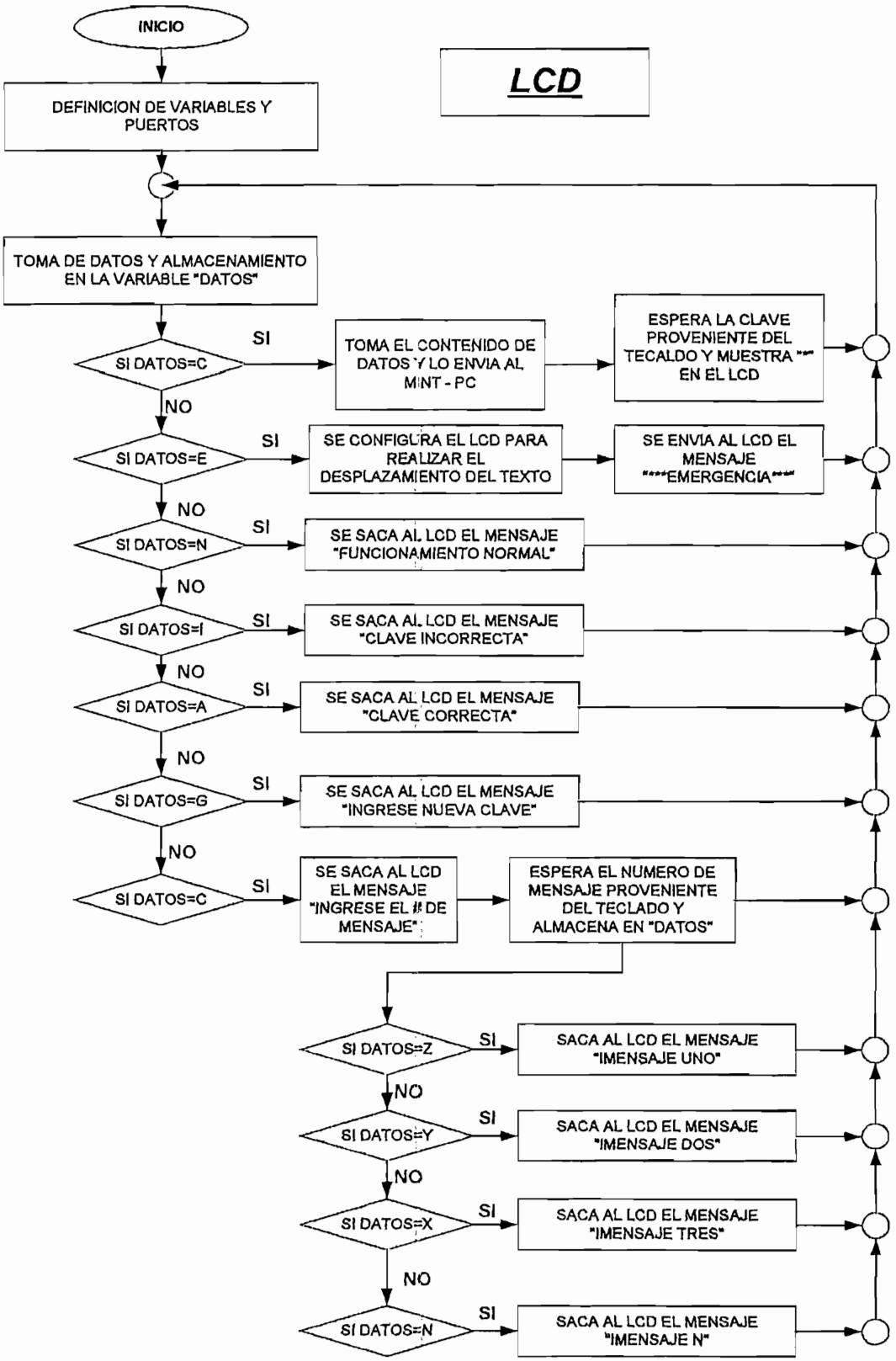
Luego se toma la información de las diferentes variables y se envía a mostrar en el LCD.

El programa completo se encuentra en la parte de Anexos.

3.2.3 MODULO DE CONTROL MODCON.

3.2.3.1 Diagrama de Flujo.




INICIO
LCD
DEFINICION DE VARIABLES Y PUERTOS
TOMA DE DATOS Y ALMACENAMIENTO EN LA VARIABLE "DATOS"
SI DATOS=C
SI
TOMA EL CONTENIDO DE DATOS Y LO ENVIA AL MINT - PC
ESPERA LA CLAVE PROVENIENTE DEL TECALDO Y MUESTRA "*" EN EL LCD
NO
SI DATOS=E
SI
SE CONFIGURA EL LCD PARA REALIZAR EL DESPLAZAMIENTO DEL TEXTO
SE ENVIA AL LCD EL MENSAJE "****EMERGENCIA****"
NO
SI DATOS=N
SI
SE SACA AL LCD EL MENSAJE "FUNCIONAMIENTO NORMAL"
NO
SI DATOS=I
SI
SE SACA AL LCD EL MENSAJE "CLAVE INCORRECTA"
NO
SI DATOS=A
SI
SE SACA AL LCD EL MENSAJE "CLAVE CORRECTA"
NO
SI DATOS=G
SI
SE SACA AL LCD EL MENSAJE "INGRESE NUEVA CLAVE"
NO
SI DATOS=C
SI
SE SACA AL LCD EL MENSAJE "INGRESE EL # DE MENSAJE"
ESPERA EL NUMERO DE MENSAJE PROVENIENTE DEL TECLADO Y ALMACENA EN "DATOS"
SI DATOS=Z
SI
SACA AL LCD EL MENSAJE "IMENSAJE UNO"
NO
SI DATOS=Y
SI
SACA AL LCD EL MENSAJE "IMENSAJE DOS"
NO
SI DATOS=X
SI
SACA AL LCD EL MENSAJE "IMENSAJE TRES"
NO
SI DATOS=N
SI
SACA AL LCD EL MENSAJE "IMENSAJE N"

3.2.3.2 Subrutinas Importantes.

3.2.3.2.1 Definición de variables.

```
TECLA VAR BYTE          ; variable para almacenar la tecla pulsada
CICLO VAR BYTE          ; variable para determinar la salida de un proceso
R var BYTE              ; variable para repeticiones

BIP VAR PORTA.3         ; se renombra el puertoa.3 como BIP
LED VAR PORTA.2         ; se renombra el puertoa.2 como LED
SALIDA VAR PORTA.4      ; define el puerto de la salida serial

A VAR PORTB.0           ; nombre de los pines de las filas
B VAR PORTB.1
C VAR PORTB.2
D VAR PORTB.3

UNO VAR PORTB.4         ; nombre de los pines para las columnas
DOS VAR PORTB.5
TRES VAR PORTB.6
CUATRO VAR PORTB.7

CLAVE1 VAR BYTE         ; variable para almacenar la primera clave
CLAVE2 VAR BYTE         ; variable para almacenar la segunda clave
CLAVE3 VAR BYTE         ; variable para almacenar la tercera clave
CLAVE4 VAR BYTE         ; variable para almacenar la cuarta clave
```

Se asignan diferentes variables como TECLA CICLO y R para realizar varios procesos repetitivos dentro del programa que maneja el teclado.

Se renombra algunos puertos, con nombres mas fáciles de recordar y fáciles de incluir dentro del programa, como por ejemplo el puerto A.3, se denomina LED y cuando se lo necesito, tan solo se coloca el nombre LED y una declaración como HIGH o **LOW** y el puerto se colocara en Alto o Bajo respectivamente, y mediante el hardware instalado se puede manejar un led por ejemplo.

A cada fila y a cada columna que permiten manejar el teclado, se le asigna una letra y un número respectivamente, para que sea más fácil el manejo en el momento de realizar el barrido.

Se definen variables para el almacenamiento de la clave inicial o de fábrica, y que luego podrá ser cambiada por el usuario.

3.2.3.2.2 Grabación de la clave inicial.

```
CLAVE1="1"
CLAVE2="2"
CLAVE3="3"
CLAVE4="4"
```

El módulo viene con la clave 1,2,3,4 pre grabada, al momento de encender el módulo se activa esta clave y permanece mientras el MODCON esta energizado o hasta cuando el usuario decida cambiarla.

3.2.3.2.3 Subrutina Reset.

```
RESET:
   FOR R=1 TO 5
      HIGH LED:HIGH BIP
      PAUSE 50
      LOW LED :LOW BIP
      PAUSE 50
   NEXT
CICLO=1
```

Luego de haber ingresado al clave inicial y que el módulo se encuentre en funcionamiento normal, tenemos al posibilidad de resetear el módulo para bloquearlo o realizar un cambio de clave, la subrutina RESET alerta de este proceso que se inicia al presionar las teclas 7 y 9 juntas, mediante 5 pitidos de la chicharra y 5 destellos del led.

3.2.3.2.4 Parte inicial del programa.

```
PRINCIPAL:

WHILE ciclo =1                          ; inicio de la condición WHILE
   FOR R=1 TO 5
   SEROUT salida,n2400,["C"]: PAUSE 400
   NEXT
   GOTO TECLAUNO                        ;ir a comparar claves
WEND.                                   ; finalización del WHILE
```

Antes de ingresar a este segmento del programa se carga la variable CICLO con el valor de 1, se utiliza la declaración **WHILE**, para que permanezca dentro de un lazo

infinito mientras no se ingrese la clave, y envía el carácter "C" hacia el LCD, y se muestre el mensaje ingrese la clave.

3.2.3.2.5 Funcionamiento Normal.

```
NORMAL:
WHILE CICLO=2
   PAUSE 500
   SEROUT salida,n2400,["N"]: PAUSE 400
   HIGH A: HIGH B: LOW C: HIGH  D :PAUSE 200
   IF (TRES=0)AND(UNO=0)THEN RESET              ;corresponde a las teclas 7 y 9
   HIGH A: HIGH B: HIGH C: LOW  D :PAUSE 200    ; sensar solo la fila C
   IF uno = 0 THEN MENSAJES                     ;corresponde a la tecla 9 para ir a
mensajes

   ALARMA:
      IF PORTA.0=0 THEN
         SOUND porta.1,[100,10,50,10]
         PAUSE 100
         SEROUT salida,n2400,["E"]: PAUSE 400
         HIGH LED:HIGH BIP
         PAUSE 50
         LOW LED :LOW BIP
         PAUSE 50
         GOTO ALARMA
      ELSE
         GOTO NORMAL
      ENDIF
WEND
GOTO NORMAL
```

En esta parte del programa se ingresa luego de haber ingresado al clave inicial, con la posibilidad de haber cambiado la clave pre grabada y cargando el valor 2 en la variable CICLO.

Se utiliza la declaración **WHILE** para que permanezca en un lazo infinito mientras CICLO es igual a 2. Mientras permanece en el lazo se envía el carácter "N" hacia el LCD, que es interpretado como Funcionamiento Normal. Luego se sensa únicamente la fila D y se pone la condición que sean las teclas 7 y 9, para provocar un RESET, si esto no ocurre, se censa únicamente la fila D, y se condiciona que si se presiona la tecla 0, vaya al menú de mensajes; si no se ha presionado ninguna de esta teclas, ingresa a la parte de la alarma.

En la alarma se esta monitoreando continuamente el puerto A.0, par ver si ha cambiado de 1 a 0, si esto se produce se saca por el puerto A.1 tonos que provocan una alarma sonora en el Hardware, y envía el carácter "E" al LCD, además cambian de estado los puertos asignados a LED y BIP. Si el cambio de estado del puerto A.0 no se produce, regresa a la parte inicial del funcionamiento normal.

3.2.3.2.6 Comparación de Claves.

```
TECLAUNO:
   GOSUB BARRIDO                ; ir a barrido y retornar con un valor
   GOSUB PTECLA                 ; envía aun programa antirrebote para soltar tecla
   IF TECLA = CLAVE1 THEN TECLADOS          ; si el numero es igual a
CLAVE1
   GOTO FALSO                               ; caso contrario ir a alzo FALSO
TECLADOS:
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   GOSUB BARRIDO:GOSUB PTECLA
   IF TECLA=CLAVE2 THEN TECLATRES
   GOTO FALSO1
```

```
TECLATRES:
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   GOSUB BARRIDO:GOSUB PTECLA
   IF TECLA=CLAVE3 THEN TECLACUATRO
   GOTO FALSO2
TECLACUATRO:
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   GOSUB BARRIDO:GOSUB PTECLA
   IF TECLA=CLAVE4 THEN MENSAJE
   GOTO FALSO3
MENSAJE:
  SEROUT salida,n2400,[TECLA]: PAUSE 100
  FOR R=1 TO 3                          ; Dos pitido indica clave correcta
     PAUSE 100
     HIGH LED:HIGH BIP
     PAUSE 100
     LOW LED: LOW BIP
  NEXT
FOR r=1 TO 4
   SEROUT salida,n2400,["A"]: PAUSE 400
NEXT
   HIGH A: HIGH B: HIGH C: LOW  D :PAUSE 300     ; sensar solo la fila C
   IF tres = 0 THEN GRABAUNO                ; corresponde a la tecla enter para ir a
grabar
CICLO=2

GOTO NORMAL
```

En esta parte el programa compara las claves pregrabadas con las que ingresa el usuario, y si una de ellas es falsa, tan solo espera que ingresa los cuatro caracteres y va la subrutina FALSO, en caso de las claves ser correctas, produce un tres pitidos y

tres destellos en el módulo y envía el carácter "A" hacia le LCD. Luego da la posibilidad de que si se presiona la tecla ENTER, se vaya a la subrutina de grabar una nueva clave. Si el usuario no decide cambiar la clave retorna a la parte de funcionamiento normal.

3.2.3.2.7 Subrutina de Grabado.

```
GRABAUNO:                          ;programa para cambiar la clave
  FOR r=1 TO 4
    SEROUT salida,n2400,["G"]: PAUSE 100
  NEXT

  GOSUB PTECLA:HIGH LED            ;espera a que suelte las teclas
  GOSUB BARRIDO:GOSUB PTECLA       ;ir a barrido y retornar a un antirrebote
  HIGH LED                         ; mantiene encendido el led
  CLAVE1=TECLA                     ;guarda en CLAVE1
GRABADOS:
  GOSUB BARRIDO:GOSUB PTECLA  ;ir a barrido y retornar con a un antirrebote
  HIGH LED                         ;mantiene encendido el led
  CLAVE2=TECLA                     ;guarda en CLAVE2
GRABATRES:
  GOSUB BARRIDO:GOSUB PTECLA  ;ir a barrido y retornar a un antirrebote
  HIGH LED                         ; mantiene encendido el led
  CLAVE3=TECLA                     ;guarda en valor de tecla
GRABACUATRO:
  GOSUB BARRIDO:GOSUB PTECLA  ;ir a barrido y retornar con a un antirrebote
  HIGH LED                         ;mantiene encendido el led
  CLAVE4=TECLA                     ; guarda en valor de tecla
RETURN                             ; retorna donde se realizo la llamada
```

Cuando ingresa a esta subrutina envía el carácter "G hacia el LCD, luego realiza un barrido del teclado y retorna con un valor, y ese lo graba en la variable CLAVE1, el mismo proceso se cumple para los demás caracteres de la clave, una vez completadas las 4 claves, el programa retorna donde se realizo la llamada.

3.2.3.2.8 Barrido del teclado.

```
BARRIDO:

   LOW A                                  ;sensar fila A
IF UNO =0 THEN TECLA= "1":RETURN          ; tecla pulsada retorne cargada con 1
IF DOS =0 THEN TECLA="2":RETURN           ; tecla pulsada retorne cargada con 2
IF TRES =0 THEN TECLA="3":RETURN          ; tecla pulsada retorne cargada con 3
IF CUATRO =0 THEN TECLA="A":RETURN        ; tecla pulsada retorne cargada con A
   HIGH A

   LOW B                                  ;sensar fila B
IF UNO =0 THEN TECLA="4":RETURN           ; tecla pulsada retorne cargada con 4
IF DOS =0 THEN TECLA="5":RETURN           ; tecla pulsada retorne cargada con 5
IF TRES =0 THEN TECLA="6":RETURN          ; tecla pulsada retorne cargada con 6
IF CUATRO =0 THEN TECLA="B":RETURN        ; tecla pulsada retorne cargada con B
   HIGH B

   LOW C
IF UNO =0 THEN TECLA="7":RETURN           ; tecla pulsada retorne cargada con 7
IF DOS =0 THEN TECLA="8":RETURN           ; tecla pulsada retorne cargada con 8
IF TRES =0 THEN TECLA="9":RETURN          ; tecla pulsada retorne cargada con 9
IF CUATRO =0 THEN TECLA="C":RETURN        ; tecla pulsada retorne cargada con C
   HIGH C
```

```
    LOW D
IF UNO =0 THEN TECLA="*":RETURN        ; tecla pulsada retorne cargada con *
IF DOS =0 THEN TECLA="0":RETURN        ; tecla pulsada retorne cargada con 0
IF TRES =0 THEN TECLA="#":RETURN       ; tecla pulsada retorne cargada con #
IF CUATRO =0 THEN TECLA="D":RETURN     ; tecla pulsada retorne cargada con B
   HIGH D

GOTO BARRIDO
```

Para el barrido se van activando fila por fila y se realiza el barrido por columnas, si durante la activación de la fila se presiona la tecla correspondiente a la columna, se carga el valor asignado en la variable TECLA, y retorna a la parte del programa donde se solicito el barrido del teclado, luego de terminar con todas las filas y columnas, el programa permanece dentro del barrido mientras no se presione una tecla.

3.2.3.2.9 Subrutina Para Claves Falsas.

```
FALSO:
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   GOSUB BARRIDO: GOSUB PTECLA          ; estas teclas no comparan ninguna

FALSO1:                                 ; clave solo espera que termine de
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   GOSUB BARRIDO: GOSUB PTECLA          ; pulsar las 4 teclas y no hace nada

FALSO2:
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   GOSUB BARRIDO: GOSUB PTECLA
```

```
FALSO3:
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   FOR   R=1 TO 10                          ; 10 pitidos indica clave incorrecta
      PAUSE 150
      HIGH LED: HIGH BIP
      PAUSE 150
      LOW LED : LOW BIP
   NEXT

SEROUT salida,n2400,["I"]: PAUSE 4000
CICLO=1

GOTO PRINCIPAL
```

Cuando una de las claves ingresadas es incorrecta ingresa esta subrutina, el programa espera que se ingresen las 4 claves y luego envía el carácter "F" al LCD, se producen 10 pitidos y 10 destellos del led, y retorna a esperar una nueva clave.

3.2.3.2.10 Condición para el puerto del LCD.

```
inicio:

SERIN portb.1,N2400,datos          toma los datos del portb.1 y almacena en
datos
IF datos="C" THEN clave          ; si el contenido de datos es "C" vamos a clave
IF datos="E" THEN emergencia     ; si datos es "E" vamos a emergencia
IF datos="N" THEN normal          si datos es "N" vamos a normal
IF datos="I" THEN incorrecto     ; si datos es "I" vamos a incorrecto
```

```
IF datos="A" THEN autorizado    si datos es "A" vamos a autorizado
IF datos="G" THEN grabar        ;si datos es "G" vamos a grabar
IF datos="M" THEN mensajes      ; si datos es "M" vamos a mensajes
IF datos="Z" THEN mensaje1      ; si datos es "Z" vamos a mensaje1
IF datos="Y" THEN mensaje2      si datos es "Y" vamos a mensaje2
IF datos="X" THEN mensaje3      si datos es "X" vamos a mensaje3

GOTO inicio
```

Se utiliza la declaración **IF THEN**, para colocar una determinada condición al puerto B.1 del PIC que maneja el LCD, es así que dependiendo del carácter que ingrese por el puerto, el programa se direcciona a una determinada subrutina, que se encarga de enviar el mensaje correspondiente para que se muestre en el LCD.

3.2.3.2.1Subrutinas de Envío de Mensajes.

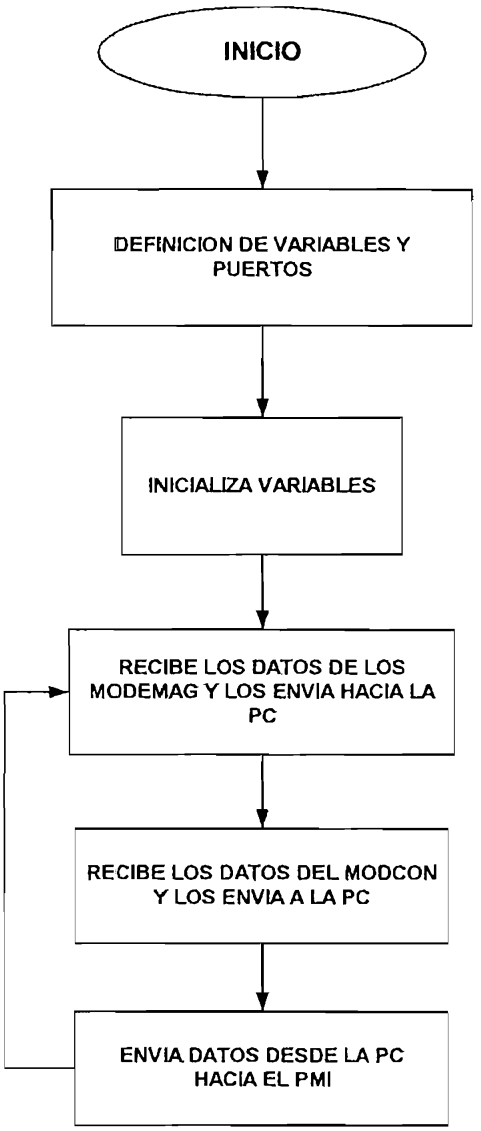
```
normal:                          ; inicio subrutina normal
   LCDOUT $fe,1 ," Funcionamiento "
   LCDOUT $fe,$C0 ,"    Normal    "
   SEROUT SALIDA,N2400,[datos]
   PAUSE 2500
   LCDOUT $FE,1                  ; Limpia la pantalla
   GOTO inicio
```

Las subrutinas se encuentran configuradas para enviar el mensaje correspondiente, ya sea en una línea o en doble línea hacia el LCD , termina de enviar el mensaje y retorna a continuar censando el puerto B.1, para tomar el carácter y llamar a la subrutina adecuada.

El programa completo se encuentra en la parte de ANEXOS

3.2.4 MODULO DE INTERFAZ CON LA PC – MINT – PC.

3.2.4.1 Diagrama de Flujo.



3.2.4.2 Subrutinas Importantes.

3.2.4.2.1 Definición de Puertos.

```
TXPC VAR PORTA.0
RXPC VAR PORTA.1
RXMCONTROL VAR PORTA.2
RXPMIMICO VAR PORTA.3
TXPMIMICO VAR PORTA.4
RXMODEMAG1 VAR PORTB.0
RXMODEMAG2 VAR PORTB.1
```

Se renombran cada uno de los puertos para facilitar el manejo el momento de receptar o enviar información desde o hacia los diferentes módulos.

3.2.4.2.2 Validación de la Clave

```
lazo:
SERIN RXMCONTROL,N2400,datos1        ; validación clave
  IF datos1 ="N"  THEN
     GOTO inicio
  ELSE
     GOTO lazo
  ENDIF
```

El programa al ingresar a esta parte, toma los datos provenientes del modulo de control, que si es igual al carácter "N" el programa continua, caso contrario el programa se mantiene en un lazo mientras en el módulo de control no se haya ingresado la clave correcta y se reciba el caracter "N" de funcionamiento normal

3.2.4.2.3 Subrutinas de Transmisión y Recepción.

recibir:

```
  FOR R=1 TO 10                    ; lazo de 1 a 10

    SERIN RXMODEMAG1,N2400,datos0          ; toma los datos del modemag
    SERIN RXMODEMAG1,N2400,datos1          ; toma los datos del modemag
    IF datos0 = "0" THEN contador =1                ; si la plaza 1 esta libre
aumenta
    IF datos0 = "1" THEN contador = contador +1    ; el contador, si la plaza2
    IF datos0 = "2" THEN contador = contador +1    ; esta libre aumenta el
    IF datos0 = "3" THEN contador = contador +1    ; contador y así
    IF datos0 = "4" THEN contador = contador +1    ; sucesivamente con las
    IF datos0 = "5" THEN contador = contador +1    ; demás plazas.
    IF datos0 = "6" THEN contador = contador +1
    IF datos0 = "7" THEN contador = contador +1
    IF datos0 = "8" THEN contador = contador +1
    IF datos0 = "9" THEN contador = contador +1

    SEROUT TXPC,N2400,[datos0]             ; transmite los datos hacia la PC
    SEROUT TXPC,N2400,[datos1]

  NEXT

IF contador = 10 THEN
  PAUSE 300
  SEROUT TXPMIMICO,N2400,["P"]
ELSE
  PAUSE 300
  SEROUT TXPMIMICO,N2400,[#contador]
```

```
ENDIF

SERIN RXMCONTROL, N2400,datos2
  FOR R= 1 TO 3
  IF datos2 = "X" THEN  SEROUT TXPMIMICO,N2400,[datos2]
        IF datos2 = "Y" THEN  SEROUT TXPMIMICO,N2400,[datos2]
        IF datos2 = "Z" THEN  SEROUT TXPMIMICO,N2400,[datos2]
        IF datos2 = "E" THEN  SEROUT TXPMIMICO,N2400,[datos2]
  NEXT

  GOTO inicio
```

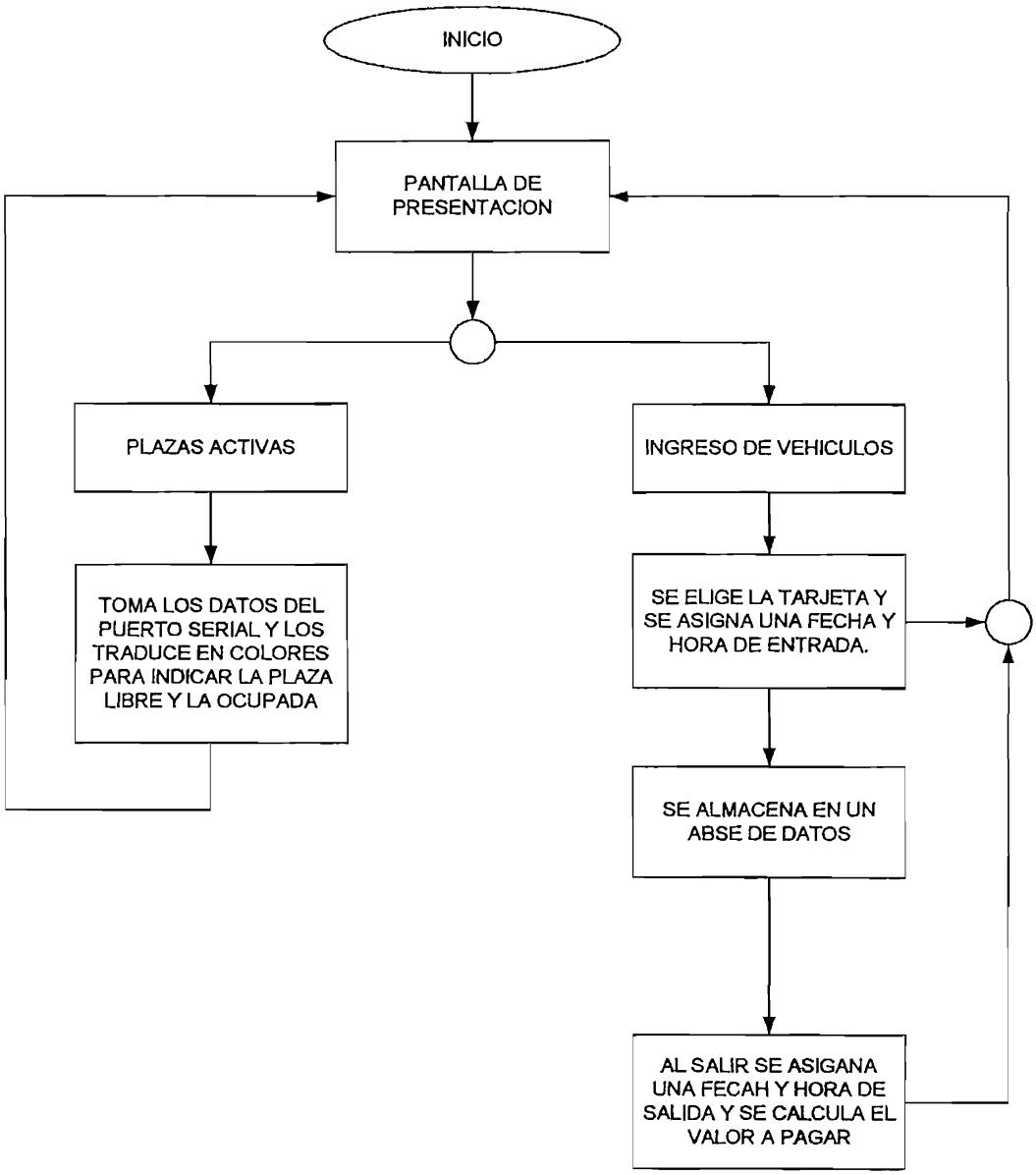
En esta subrutina se encuentra constantemente tomando datos del medio externo, que llegan de los MODEMAG, y los almacena en la variables datos0 y datos1 y envía esta información por el pin que se comunica con la PC, además dependiendo de si la plaza esta libre u ocupada se incrementa la variable CONTADOR, para al final de terminado el lazo de 10 ciclos, enviar el valor de la variable CONTADOR al panel mímico y que este valor se refleje en las matrices, para indicar el número de plazas libres.

Luego el programa realiza un barrido del pin de recepción del panel de control, y almacena esta información en la variable datos2, para luego dependiendo de la información que le llega, enviarla hacia el panel mímico y poder mostrar la información al usuario.

El programa completo se encuentra en los Anexos.

3.2.5 SOFTWARE DE VISUALIZACION Y REPORTE - SVR.

3.2.5.1 Diagrama de Flujo.



3.2.5.2 Subrutinas Importantes.

3.2.5.2.1 Pantalla principal.

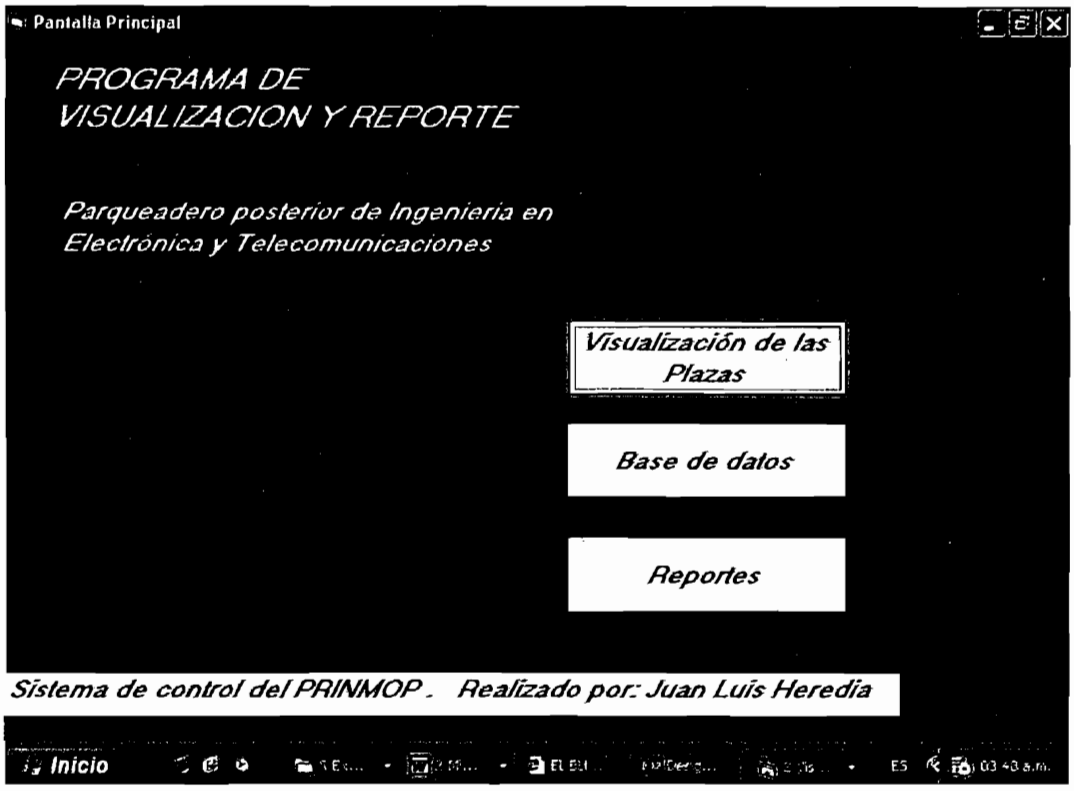


Figura 3.2

La figura muestra la pantalla de presentación del SVR, y cada uno de sus componentes. A continuación se muestra el programa para cada uno de los Command Button.

```
Private Sub Cmd_ingreso_Click ()
   Frm_base.Show
END Sub
```

```
Private Sub Cmd_plazas_Click ()
 Frm_plazas.Show
END Sub
```

Esta pantalla es una presentación del prototipo, y lo que contiene es tres botones (Command Button) para mostrar los formularios adicionales, el primero sobre plazas activas, el segundo para ingresar los datos de las tarjetas entregadas a los usuarios y el tercero para obtener estadísticas del sistema.

3.2.5.2.2 Pantalla de plazas activas.

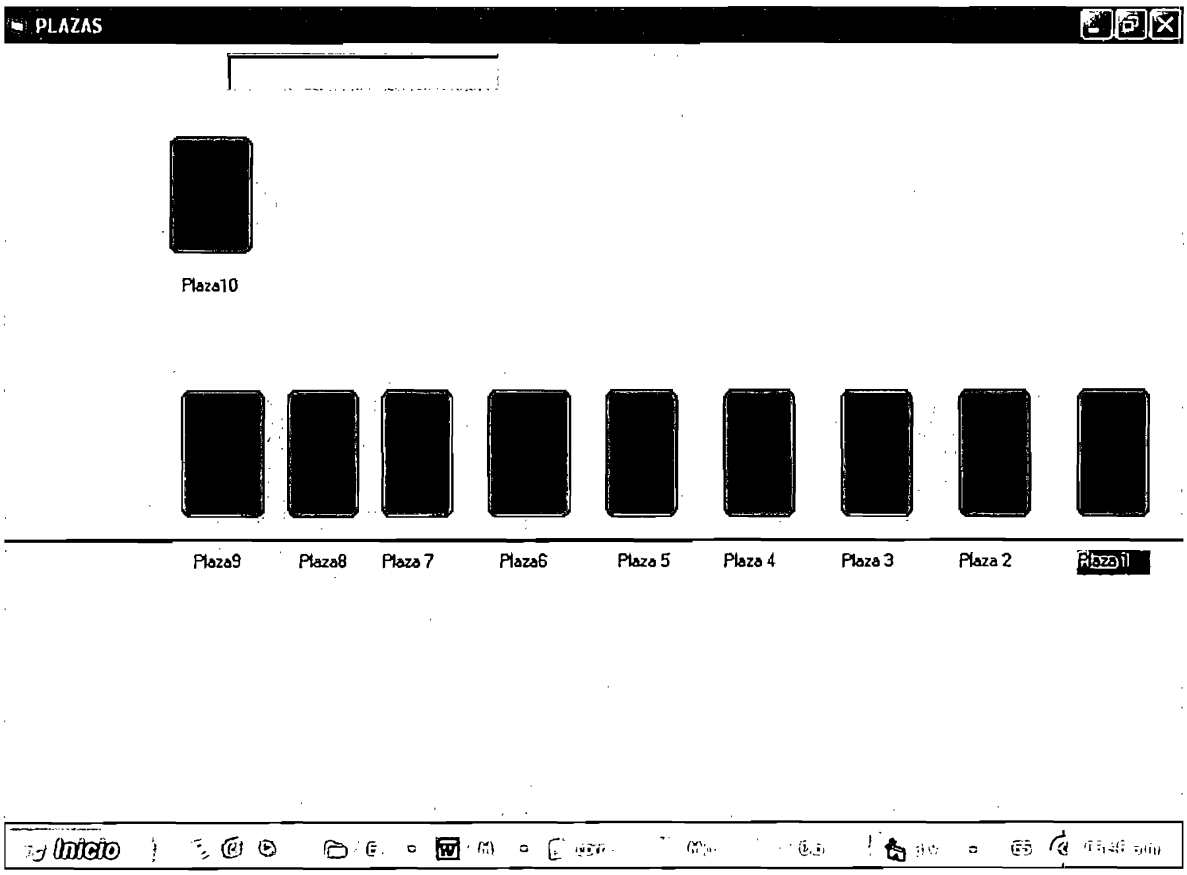


Figura 3.3

```
Private Sub Form_Load ()
MSComm1.PortOpen = True
END Sub

Private Sub MSComm1_OnComm ()

Dim PLAZA As String

   PLAZA = MSComm1.Input
   txtPantalla.Text = PLAZA

   IF PLAZA = "A0" THEN
      ShpPLaza1.FillColor = QBColor(12)
   END IF

   IF PLAZA = "0A" THEN
      ShpPLaza1.FillColor = QBColor(10)
   END IF

END Sub
```

Para esta sección se abre el puerto serial mediante MSComm1.PortOpen = True. Luego se indica que al ingresar datos por el puerto serial, los almacene en la variable PLAZA que previamente fue definida como una cadena de caracteres.

Se usa la condición **IF ... THEN ....ELSE**, para generar dos condiciones, si la cadena de caracteres es "A0" el color del Shape Plaza 1 es verde, caso contrario es rojo, la misma lógica se maneja para las demás plazas..

3.2.5.2.3 Pantalla base de datos.

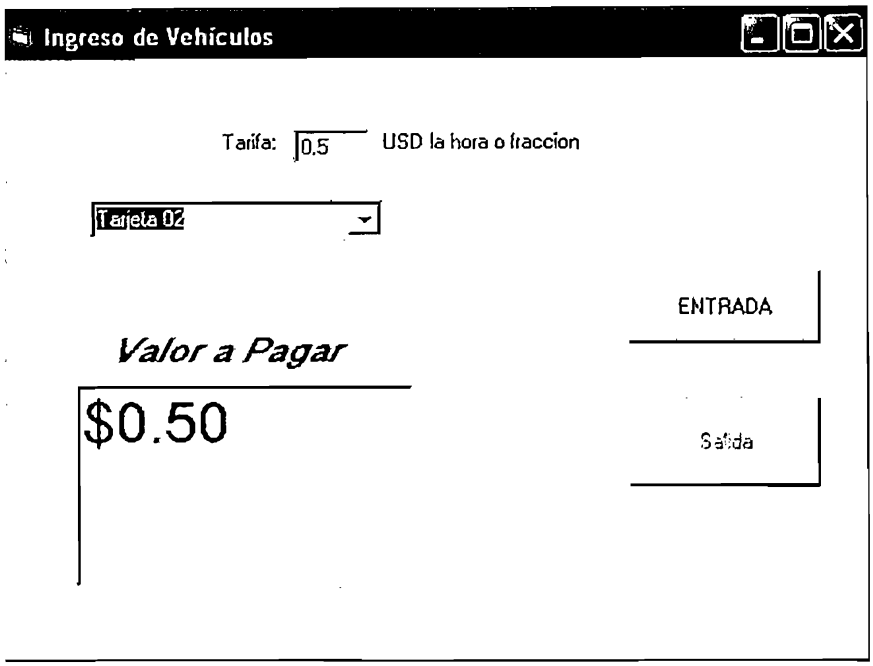


Figura 3.4

Para manejar la base de datos denominada Prinmop.mdb, se usa un ADODC que permite conectar permanente la pantalla de visual con la base de datos. Se utiliza un Combo box para tener almacenadas las diferentes tarjetas que se pueden manejar, se selecciona la tarjeta que se asigne en ese momento y se presiona el botón ENTRADA, de esta manera se almacena en la base de datos la identificación de l a tarjeta , fecha y hora de entrada.

Cuando el usuario requiere salir se selecciona la tarjeta y se presiona Salida, de esta forma se asigna una fecha y hora de salida y se calcula el valor a pagar que se muestra en el Text Box.

Para proporcionar al formulario, cierta seguridad, se establece, que si una tarjeta ya fue ingresada, el botón de ENTRADA se desactiva, y al seleccionar la tarjeta de un usuario que desea salir, luego de terminar el valor a pagar, se desactiva el botón SALIDA.

3.2.5.2.4 Pantalla de Reportes.

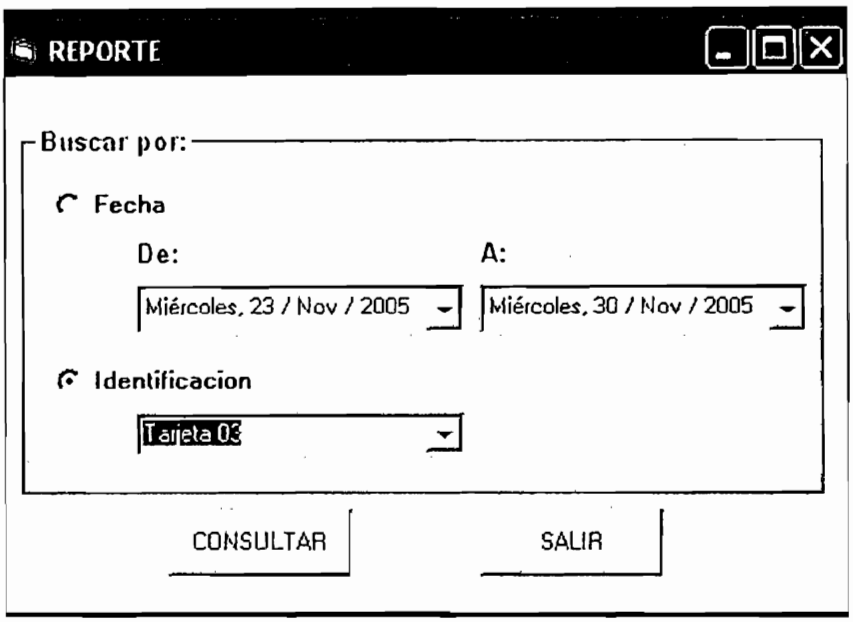
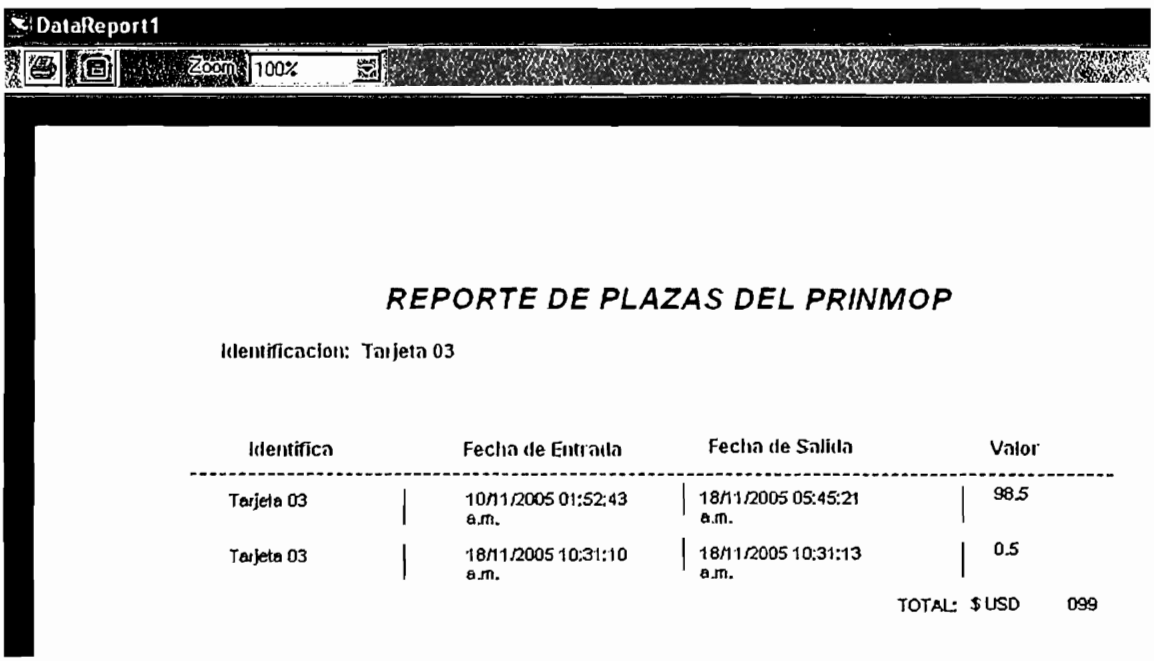


Figura 3.5

En esta sección se obtiene un reporte de, tiempo de ocupación y valor facturado de cada una de las plazas. Se selecciona la fecha inicial y la fecha hasta la cual se desea el reporte, luego se selecciona la tarjeta, y se presiona el botón consultar, y se obtiene el siguiente formato de reporte.



CAPITULO 4

4. CONSTRUCCION, PRUEBAS Y COSTOS

4.1 ENSAMBLADO

4.1.1 MODULO DE DETECCION MAGNETICA – MODEMAG

4.1.1.1 Primera tarjeta MODEMAG.

En la siguiente figura se muestra la primera tarjeta del MODEMAG.

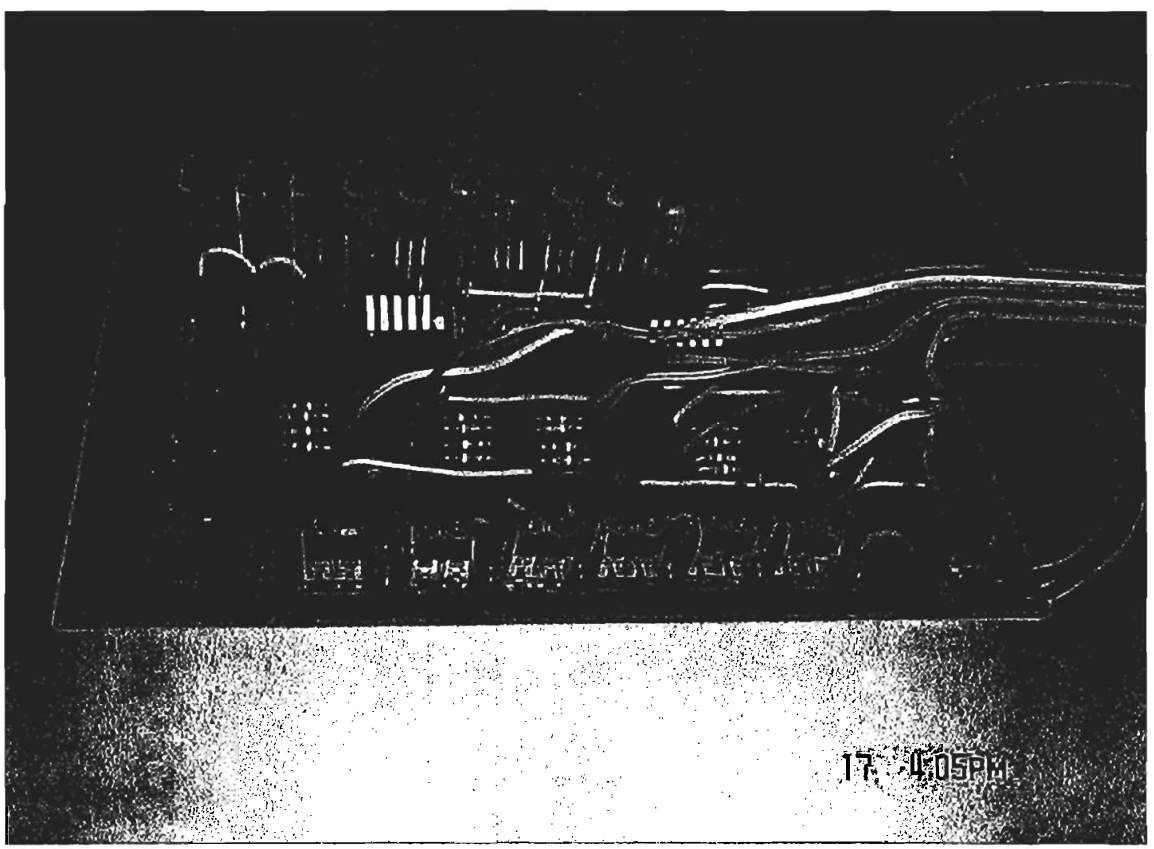


Figura 4.1

Esta tarjeta contiene las borneras para la conexión de los Loops, las compuertas lógicas, los dip switch para la reserva manual de las plazas o test del módulo, y los leds bicolor para indicar que tipo de activación se ha utilizado, manual o automática.

4.1.1.2 Segunda tarjeta MODEMAG.

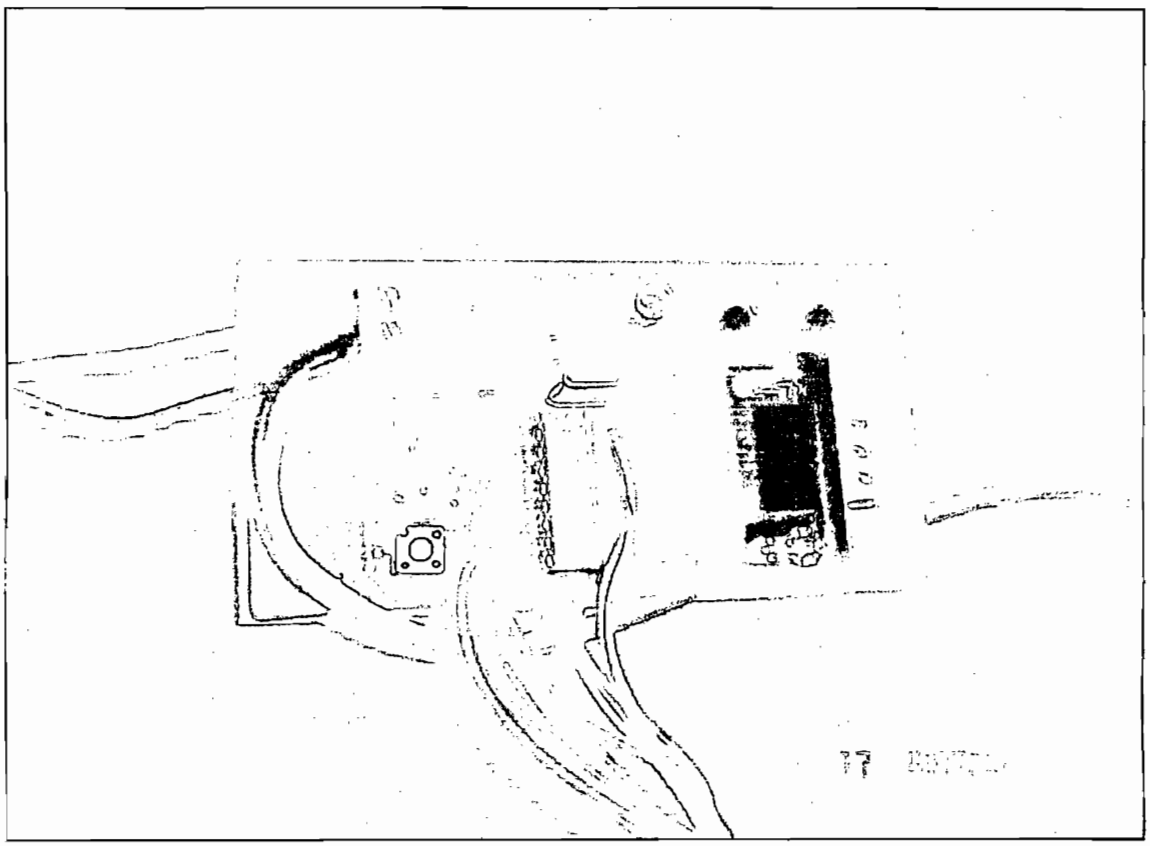


Figura 4.2

A esta tarjeta se une la salida de las compuertas lógicas, a los pines del PIC, además contiene los leds indicadores del MODEMAG, y el conector DB 9 , para la transmisión de datos hacia el MINT - PC

4.1.1.3 Modemag Completo.

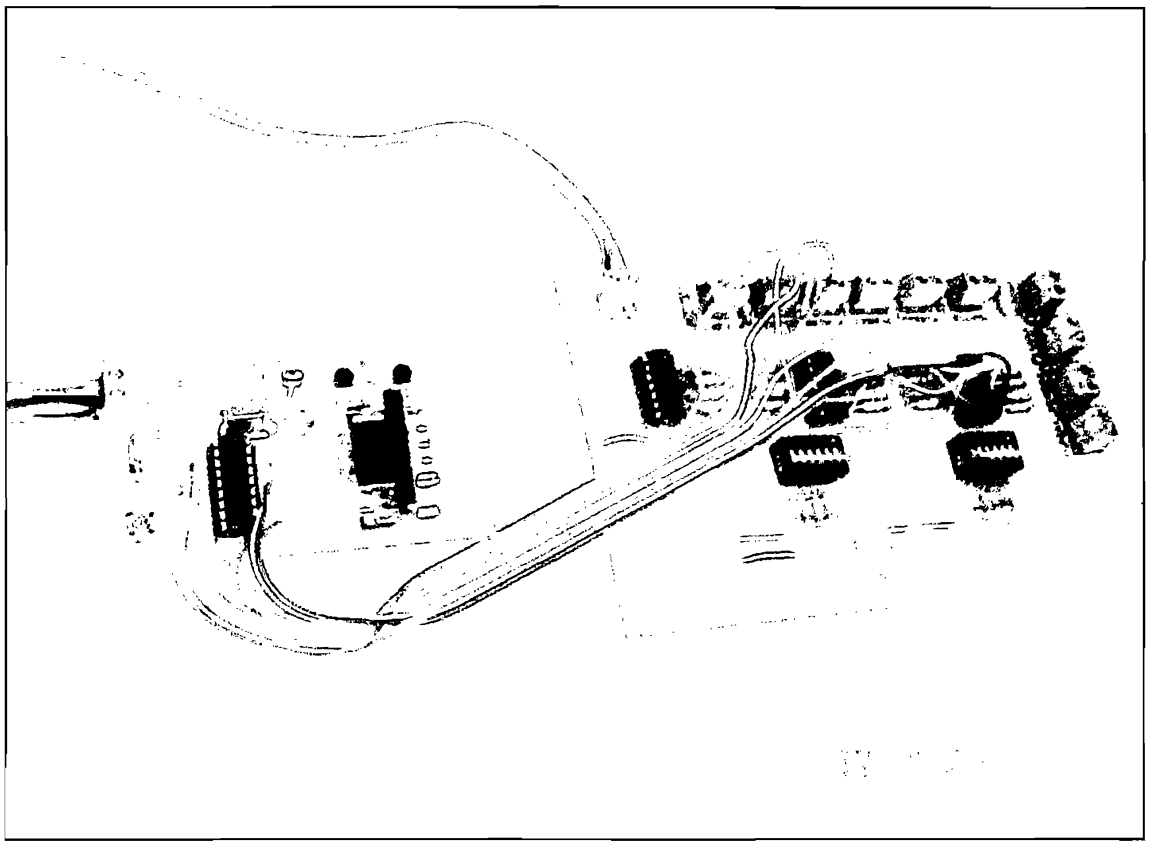


Figura 4.3

En la figura se muestra el MODEMAG completo, tanto la tarjeta que recepta los datos de los loops magnéticos, como la tarjeta que procesa la información y la envía hacia el MINT – PC.

4.1.1.4 Vista Frontal Modemag Funcionando.

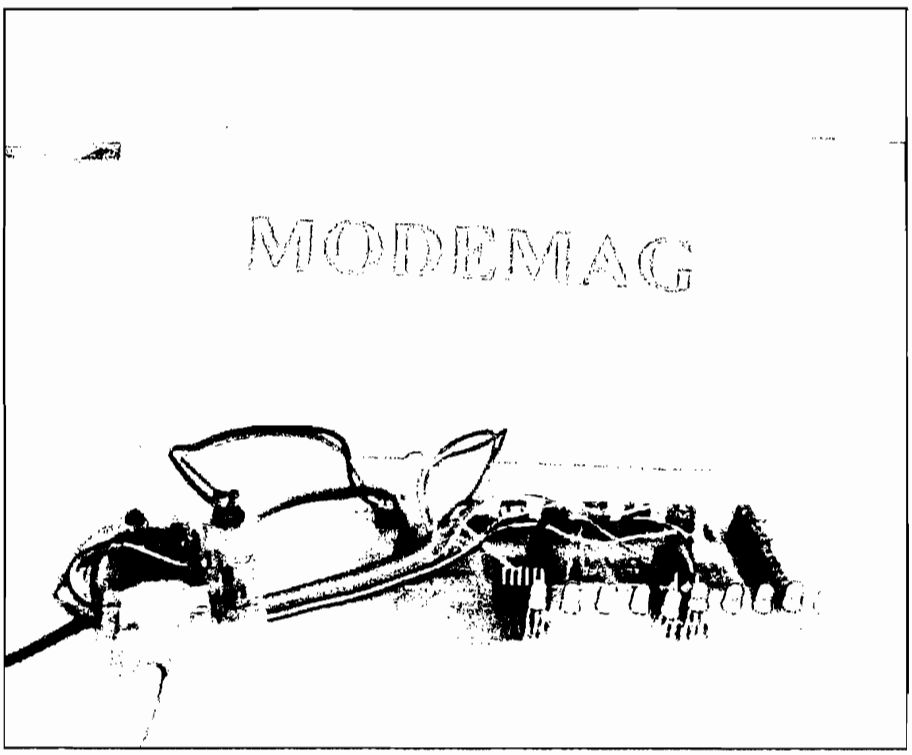


Figura 4.4

En la figura se muestra el MODEMAG energizado, y con algunas de sus plazas ocupadas, unas en forma manual y otras en forma automática (leds de color rojo en forma manual y leds de color verde en forma automática), también se puede observar el funcionamiento de los leds indicadores.

4.1.2 MODULO DE CONTROL – MODCON

4.1.2.1 Vista Frontal MODCON

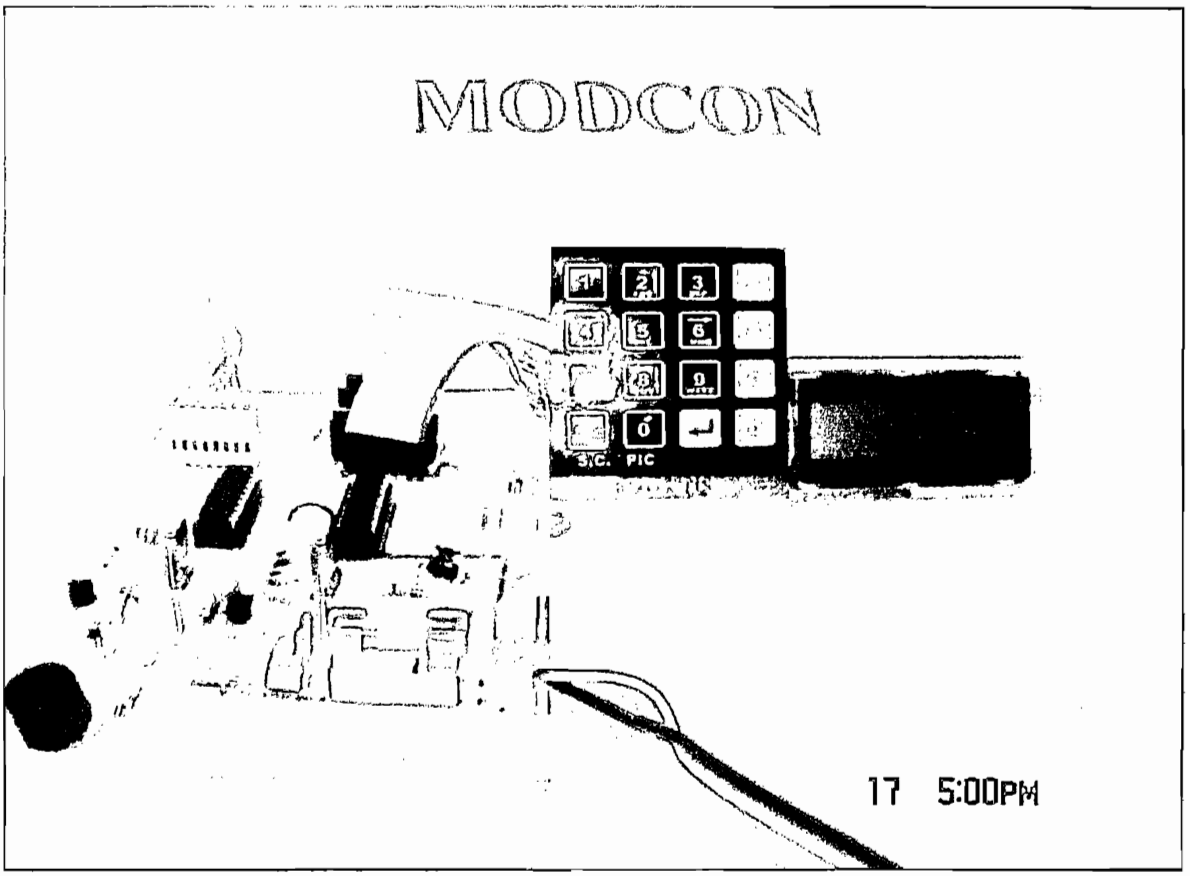


Figura 4.5

En la figura se observa los diferentes componentes del MODCON, como son el teclado, LCD, chicharra, leds indicadores, PICs, borneras de poder y de comunicación con el MINT – PC.

4.1.3 PANEL MIMICO INFORMATICO - PMI

4.1.3.1 Vista Frontal Panel Mimico

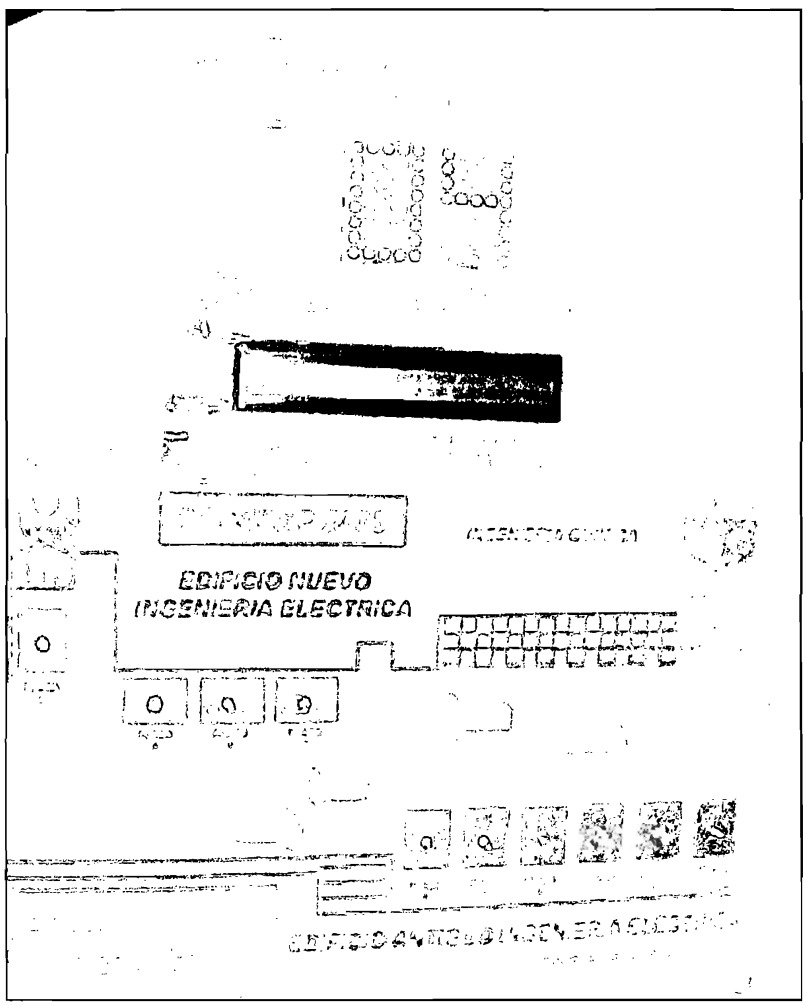


Figura 4.6

La figura muestra un esquemático del parqueadero a monitorear, aquí se puede observar que las plazas ocupadas tienen encendido su led respectivo, el LCD para la mensajería y las matrices indicando el número de plazas libres, este número es igual al numero de leds apagados en el esquemático del parqueadero,.

4.1.4 MODULO DE INTERFAS CON LA PC – MINT – PC

4.1.4.1 Vista Frontal Mint - PC

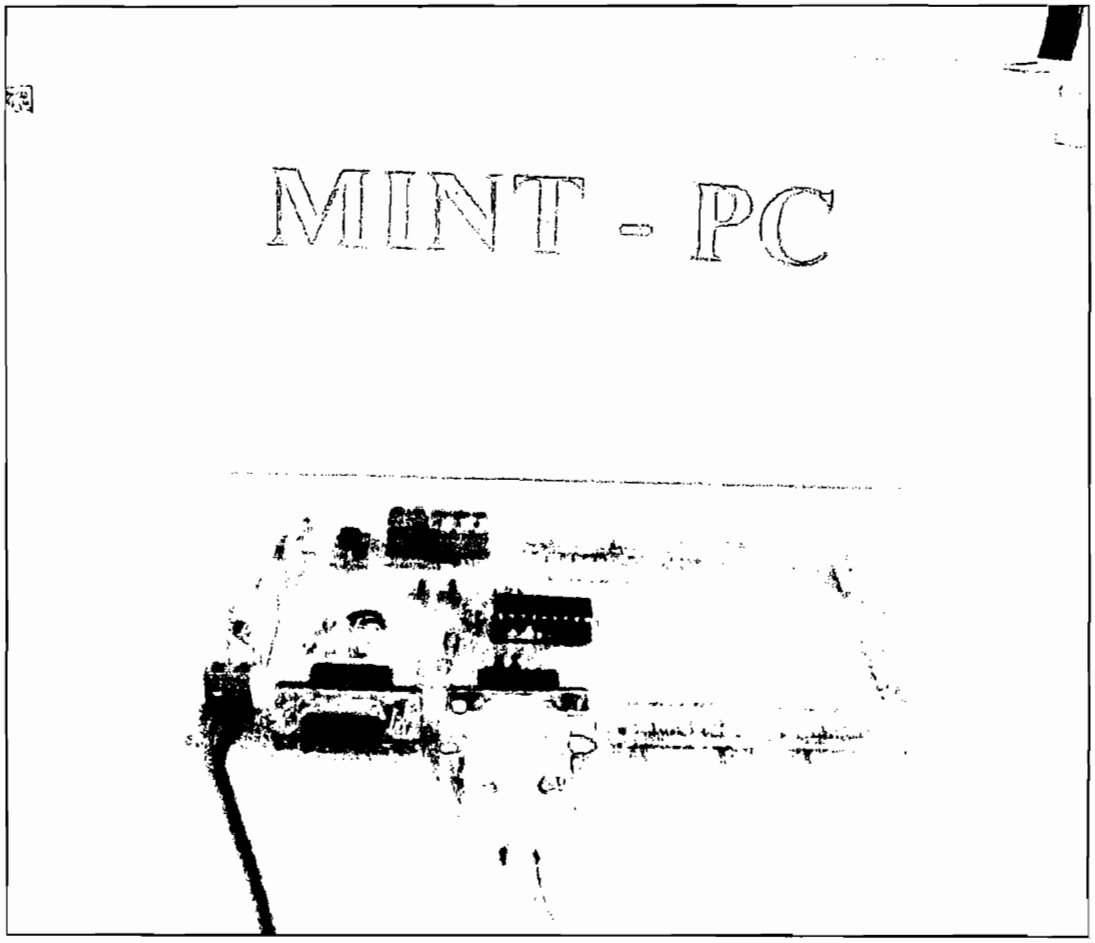


Figura 4.7

La figura muestra los conectores DB9 para la comunicación con los diferentes MODEMAG y con la PC, los leds indicadores, el PIC, y las borneras para la comunicación con el MODCON y con el Panel Mímico.

4.1.4.2 Vista superior MINT – PC.

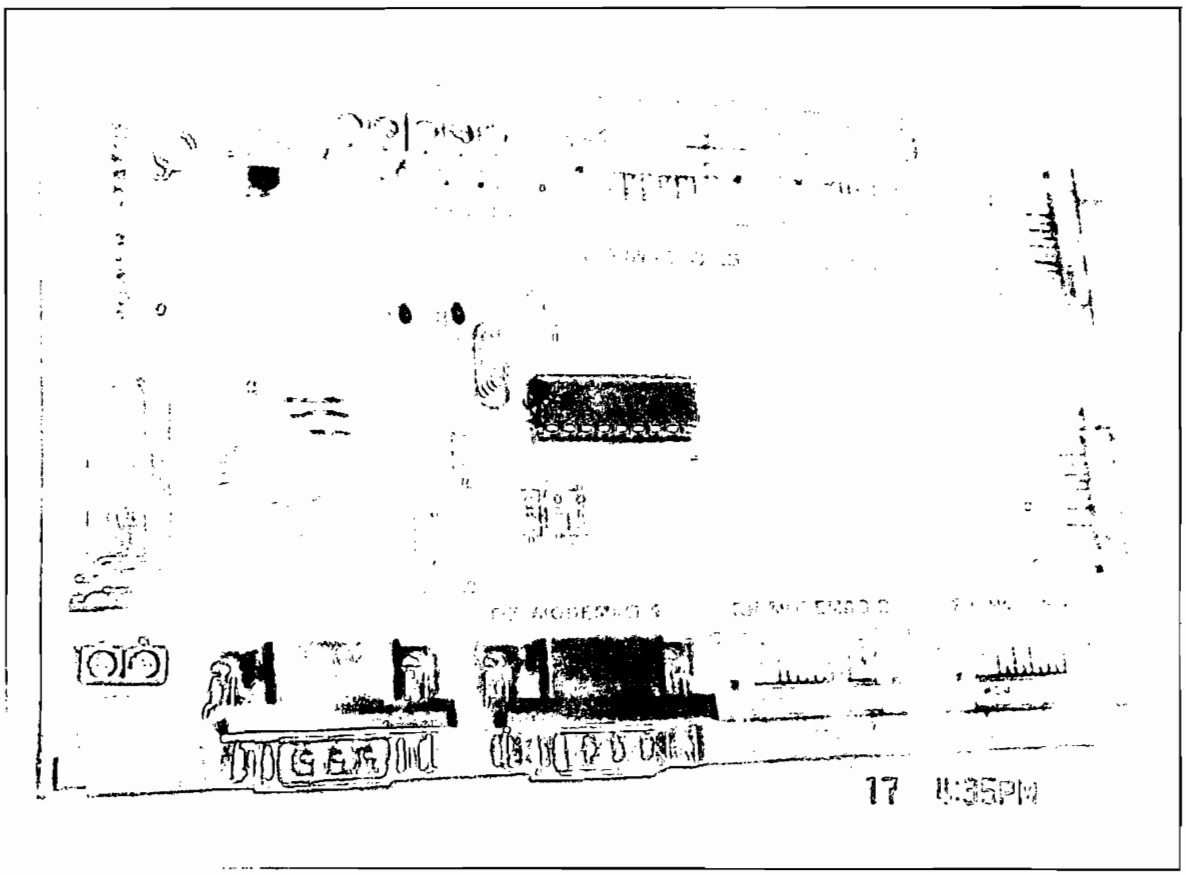


Figura 4.8

La figura muestra con mayor detalle cada unos de los elementos que conforman el MINT – PC.

4.1.5 PRINMOP COMPLETO.

4.1.5.1 Partes del PIRNMOP.

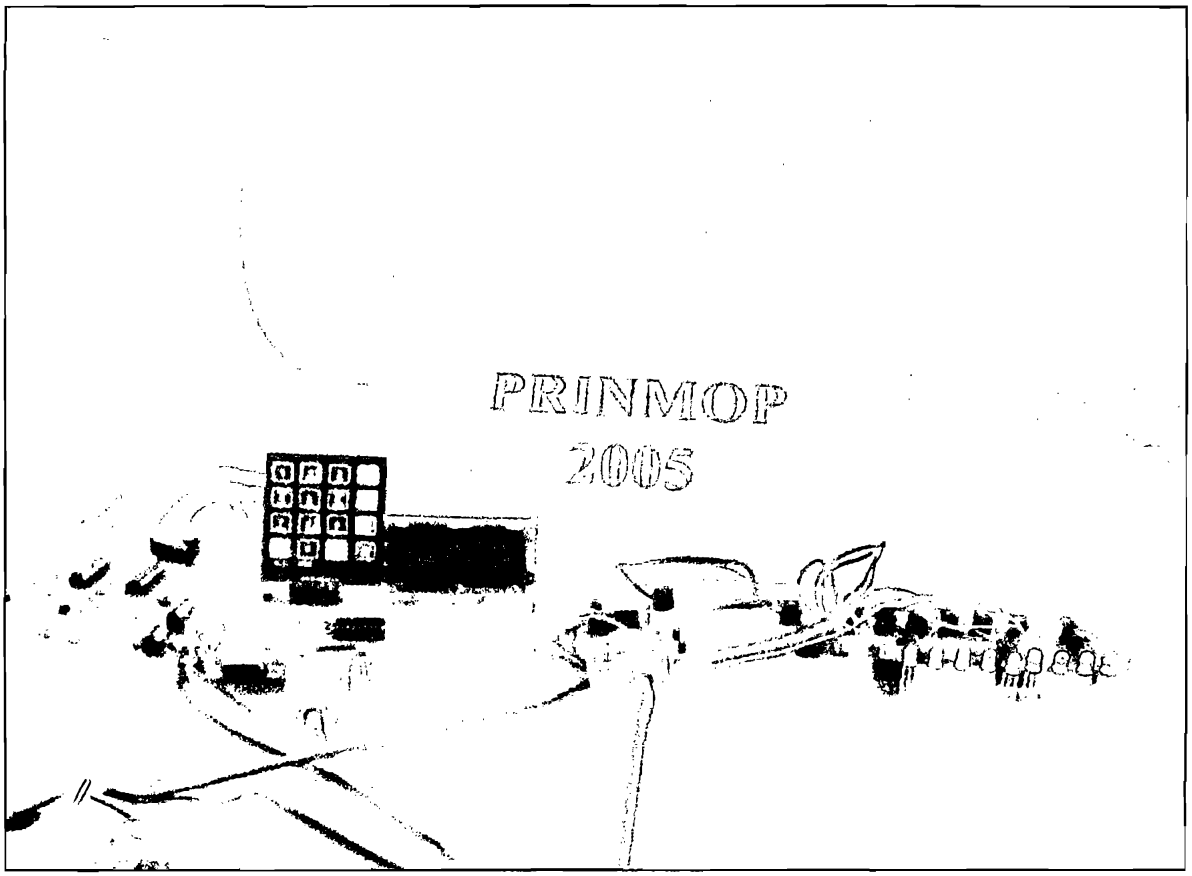


Figura 4.9

La figura muestra las tarjetas conectadas y ensambladas del PRINMOP.

4.1.5.2 PIRNMOP Ensamblado

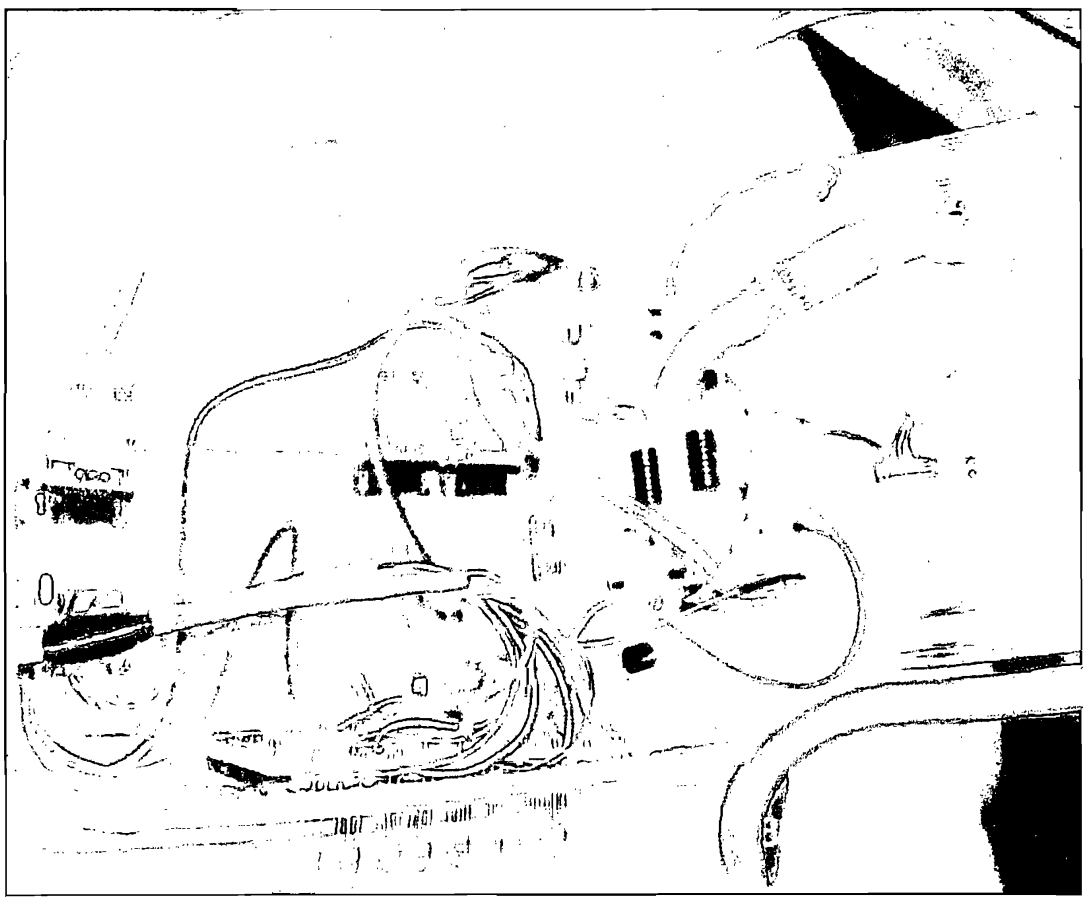


Figura 4.10

4.2 MANUAL DE USUARIO

4.2.1 CONFIGURACION DEL PRINMOP

4.2.1.1 Instalación De Los Loop Magnéticos.

En el mercado se pueden encontrar loops con diferente tipo de alimentación, que puede ser de 24 V DC o 110 V AC, por facilidad de instalación se recomienda usar loops con alimentación de 110V AC, en vista de que la mayoría de instalaciones tienen puntos de conexión eléctrica que suministran este tipo de voltaje.

El loop debe tener un diámetro de 1m de ancho por 1.5 de largo, para cubrir la mayor parte del vehículo, las espiras deben colocarse una sobre otra para lograr buenos resultados en la detección y evitar lecturas falsas.

En la siguiente figura se muestra la forma de instalación de las espiras.

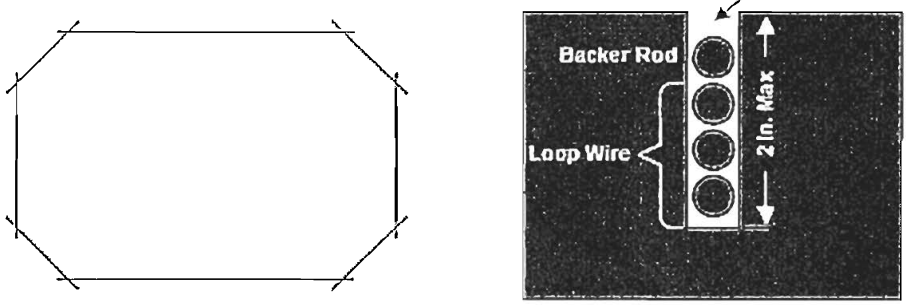


Figura 4.11

Se debe tener cuidado de redondear las esquinas para evitar lecturas falsas debido a una curvatura excesiva del cable.

Es recordable que el sensor se encuentre lo mas cerca del loop, para evitar interferencias en el momento de instalar otros dispositivos similares.

4.2.1.2 Instalación De Los Modemag.

El MODEMAG se encuentra constituido por dos placas, la primera y la más grande, posee una bornera de energía , y las 10 restantes borneras se encuentra etiquetadas desde la plaza 1 hasta la plaza 10 , en cada una de estar borneras se colocan los terminales del sensor de cada loop magnético. En la misma placa posee dos dip switch que sirven para realizar test o reserva manual de plazas, al momento de instalar el MODEMAG deben estar abiertos los dip switch, para no provocar daño a los led bicolor, en caso de una mala conexión del loop.

En la segunda placa, que es la mas pequeña, pose una bornera para la energía, los leds indicadores y un conector DB9 para la conexión con el MINT – PC, en esta placa se instala la alimentación de 5 V DC y el cable de conexión con el MINT –PC..

Una vez conectados los loop magnéticos, se energiza el módulo, y se debe encender el led de power y el de status debe titilar 5 veces, luego de unos segundos el led de TX de datos también debe empezar a titilar y el led de status titila dos veces durante el funcionamiento normal.

Luego se puede manipular el dip switch para verificar que los leds bicolor se encuentran en buen estado, si solo se activa manualmente, el led es de color rojo, si solo se activa mediante el loop el color es verde, pero si se activan de las dos formas anteriores tenemos una combinación de rojo y verde.

4.2.1.3 Instalación Del Modcon.

En este módulo se debe tener cuidado que se encuentren conectados a la placa el teclado, el conector para un parlante, el pulsador, y el LCD.

Posee una bornera de dos pines para la energía de 5V DC y una bornera de tres pines para la comunicación con el MINT –PC. Se energiza el módulo y se debe encender el led de power, y el led de procesamiento debe titilar, así también se debe escuchar un pitido que indica la inicialización del módulo.

En el LCD se debe visualizar un mensaje de Bienvenido y luego el mensaje de Ingrese la Clave, se debe ingresar la clave preconfigurada de fábrica y si es correcto se escucha un pitido y se visualiza el mensaje de Clave correcta, en caso contrario se escuchan 10 pitidos y se visualiza el mensaje clave incorrecta y luego de unos segundos vuelve a mostrar el mensaje Ingrese la clave. Este mensaje permanecerá mientras no se ingrese la clave correcta.

Una vez ingresada la clave correcta, el usuario tiene 5 segundos para presionar la tecla # y poder grabar una nueva clave, en caso de no hacerlo, ingresa al modo de funcionamiento normal y la clave será la pregrabada.

4.2.1.4 Instalación Del Pmi

En este módulo se debe colocar la pila de 3 V para el respaldo del RTC, conectar los cables que salen desde los MODEMAG, para obtener un reflejo de las plazas libres y ocupadas, verificar que el cable de transmisión de datos con el MINT – PC se encuentre conectado. Al energizarlo debe mostrar en el LCD el mensaje de Bienvenida, y luego se pone a esperar los datos del MINT – PC. El módulo no mostrara ninguna información, tanto en las matrices de plazas libres como en la parte de mensajería, mientras en el MODCON no se ingrese la clave correcta.

4.2.1.5 Instalación Del Mint – Pc

Posee una bornera de tres pines, donde el pin1 es la TX hacia el PMI y el pin2 es la RX desde el PMI; en la bornera de dos pines es la RX desde el MODCON; y posee una bornera de POWER, el DB9 etiquetado como TX_RX_PC, es la conexión hacia la computadora, los demás DB9 son la conexión para los diferentes MODEMAG.

Una vez energizado y realizadas la conexiones de los demás módulos, se debe encender el led de power y titilar el led de status, cada vez que se produzca intercambio de información el led de estatus titilará.

4.2.1.6 Instalación De SVR.

El software viene en un archivo ejecutable denominado PRINMOP.exe, se pone acorrer este archivo y se instala en el lugar de nuestra preferencia.

Al momento de ejecutarlo de se debe tener cuidado, que ningún otro programa se encuentre ocupando el puerto serial, puesto que no podrá tomar los datos provenientes del MINT – PC.

4.2.2 OPERACIÓN DEL PRINMOP

4.2.2.1 Configuración Inicial.

Una vez conectados todos los módulos del PRINMOP , se enciende el módulo y el MODCON entregara una alarma visual y una sonora, esperamos que termine al inicialización del módulo y se visualiza un mensaje de bienvenida y luego pide ingresar la clave, que se encuentra configurada como 1234 (esta clave es la preconfigurada y se debe colocar cuando se enciende el módulo o si ha existido falla

de energía), luego de recibir el mensaje de clave correcta, el usuario tiene 5 segundos, para presionar la tecla ENTER y poder grabar una nueva clave.

Luego de ingresar la clave y grabar una nueva, se muestra en el LCD un mensaje de Funcionamiento Normal, que indica el modo de funcionamiento.

4.2.2.2 Bloqueo del Módulo.

En caso de ser necesario que ninguna otra persona puede manipular el MODCON, existe la posibilidad de bloquearlo, presionando simultáneamente durante 2 segundos las teclas 7 y 9, el sistema genera una alarma sonora y luego de unos segundos pide ingresar la clave. Mientras la clave no sea ingresada el módulo se quedará en el estado de bloqueo.

4.2.2.3 Mensajes del Modcon.

Para el envío de mensajes pregrabados, luego de que el módulo se encuentra en el modo de funcionamiento normal, se debe presionar la tecla ESC durante 3 segundos, y se obtiene un mensaje de ingrese el número de mensaje; se ingresa el numero de mensaje que se desee, el módulo se quedara mostrando el mensaje hasta que se decida pararlo, presionando la tecla 0, luego el módulo vuelve al modo de funcionamiento normal.

4.2.2.4 Utilización del SVR.

Luego de instalar el programa de visualización y reporte, buscamos el ejecutable y se obtiene la siguiente pantalla

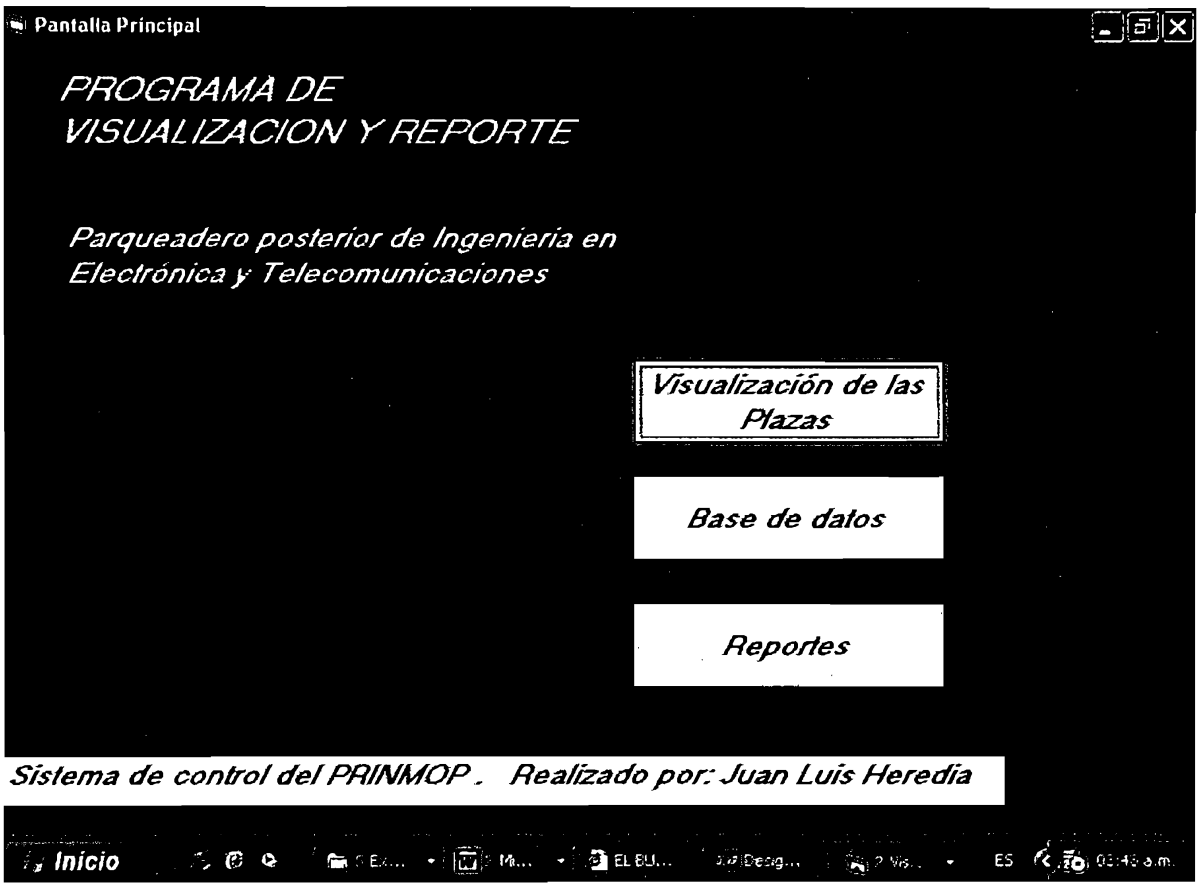


Figura 4.12

En esta pantalla se puede acceder a tres funciones principales.

Dando un clic en el botón Visualización de las plazas, obtenemos la pantalla que muestra el estado de cada una de las plazas del parqueadero.

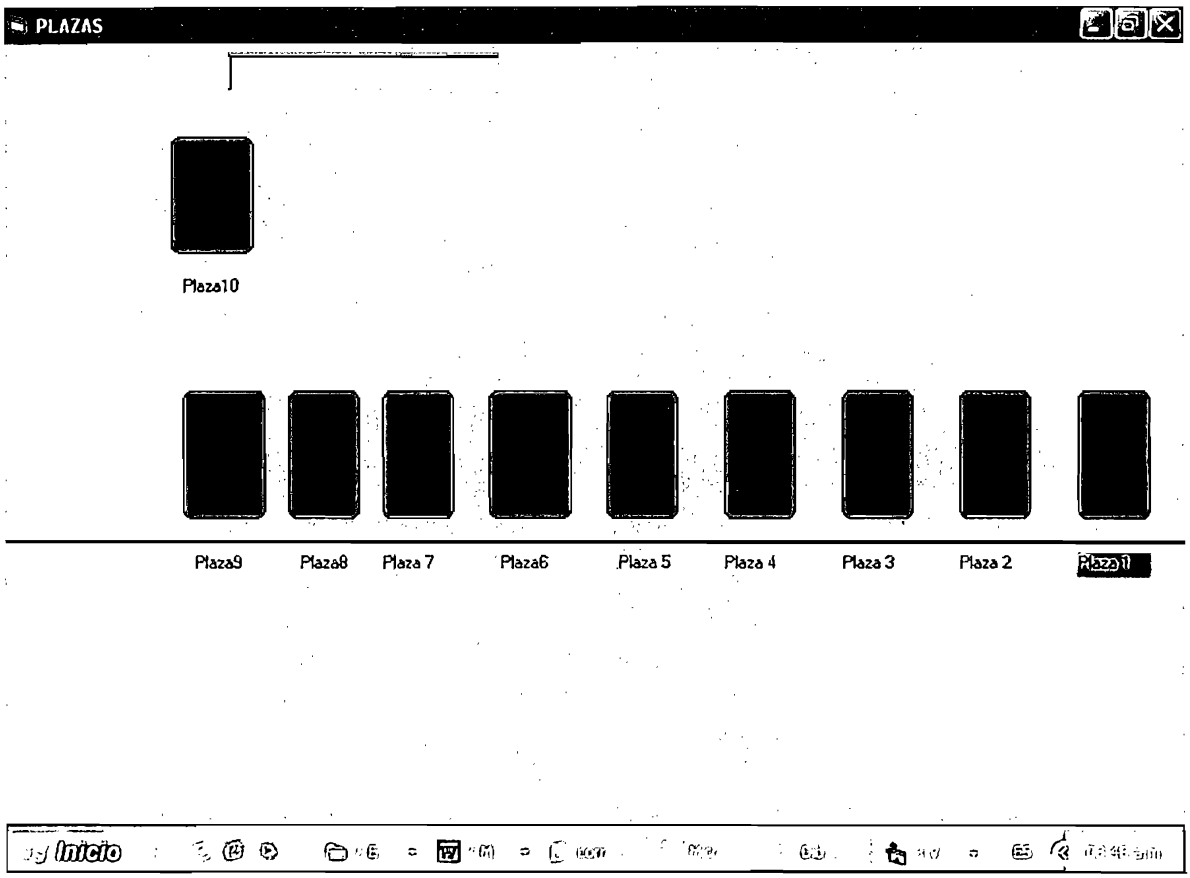


Figura 4.13

En esta pantalla el color verde indica plaza libre y el color rojo plaza ocupada, a medida que las plazas sean ocupadas o desocupadas por los usuarios el programa mostrara el estado actual de cada una de las plazas.

Dando un click sobre el botón Base de datos, obtenemos la pantalla que nos permite ingresar la hora y fecha de la tarjeta entrega a cada uno de los usuarios, y estos datos se ingresan a una base de datos diseñada en Access, como se muestra en la siguiente figura:

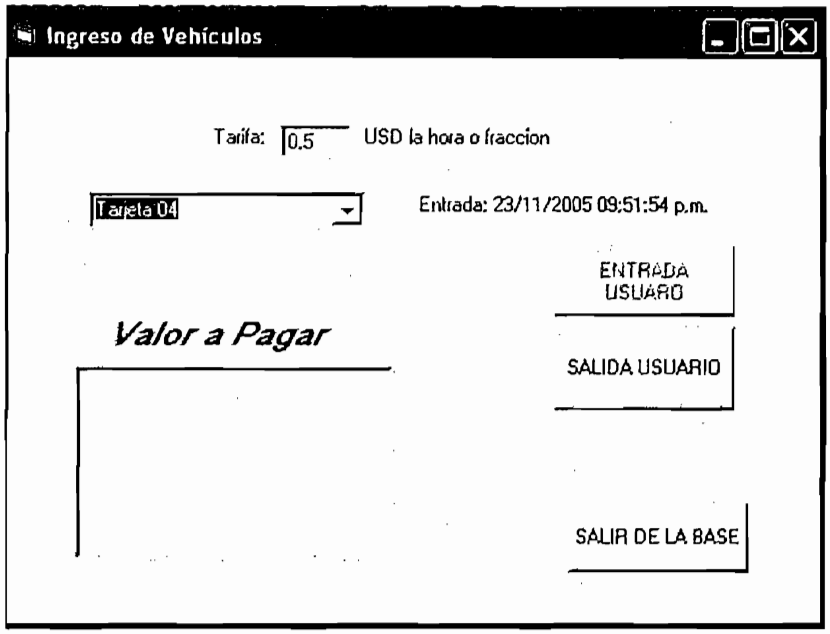


Figura 4.14

Cuando el usuario abandona el parqueadero, se debe seleccionar la tarjeta qu8e se entregó al usuario, y dar un clic sobre el botón Salida Usuario, así se registrará la fecha y hora de salida y el programa calculará el valor a pagar, como se muestra a continuación.

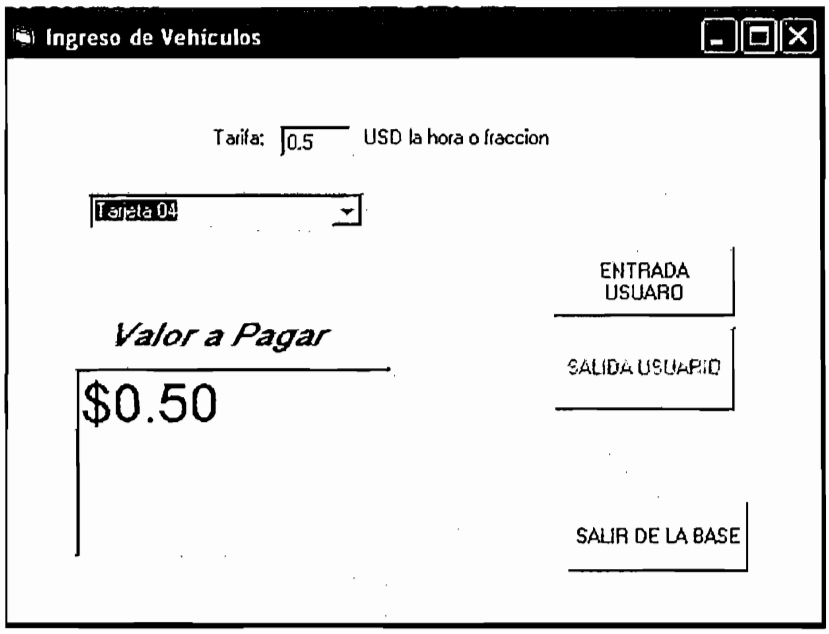


Figura 4.15

Dando un clic sobre el botón Reportes, obtenemos una pantalla donde se elige la fecha de inicio y fecha de fin del reporte, así también se selecciona el número de tarjeta, como se muestra a continuación.

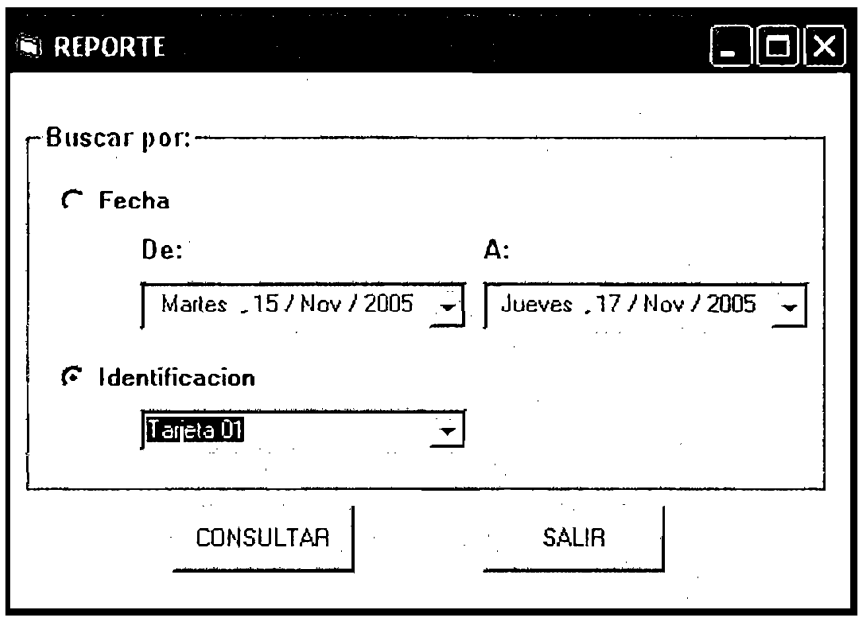


Figura 4.16

Luego se da un click sobre el botón Consultar y se obtiene el reporte de la tarjeta seleccionada, como se muestra.

REPORTE DE PLAZAS DEL PRINMOP

Identificacion: Tarjeta 01

Identifica	Fecha de Entrada	Fecha de Salida	Valor
Tarjeta 01	09/11/2005 07:10:00 a.m.	09/11/2005 03:23:23 p.m.	4.5
Tarjeta 01	10/11/2005 01:52:38 a.m.	14/11/2005 02:00:53 a.m.	49
Tarjeta 01	18/11/2005 05:45:05 a.m.	18/11/2005 10:31:01 a.m.	3
Tarjeta 01	18/11/2005 05:45:46 p.m.		0

TOTAL: $ USD 057

Figura 4.17

4.3 PRUEBAS

4.3.1 OPERACIÓN DEL MODEMAG.

El módulo posee leds bicolor para la señalización de plazas libres y ocupadas. Si se encuentran ocupadas por detección del lazo se enciende de color verde, si es ocupada por configuración manual, el led se enciende de color rojo, y en caso de activarse de las dos formas simultáneamente, se enciende una combinación de los dos colores, como se muestra en la siguiente figura.

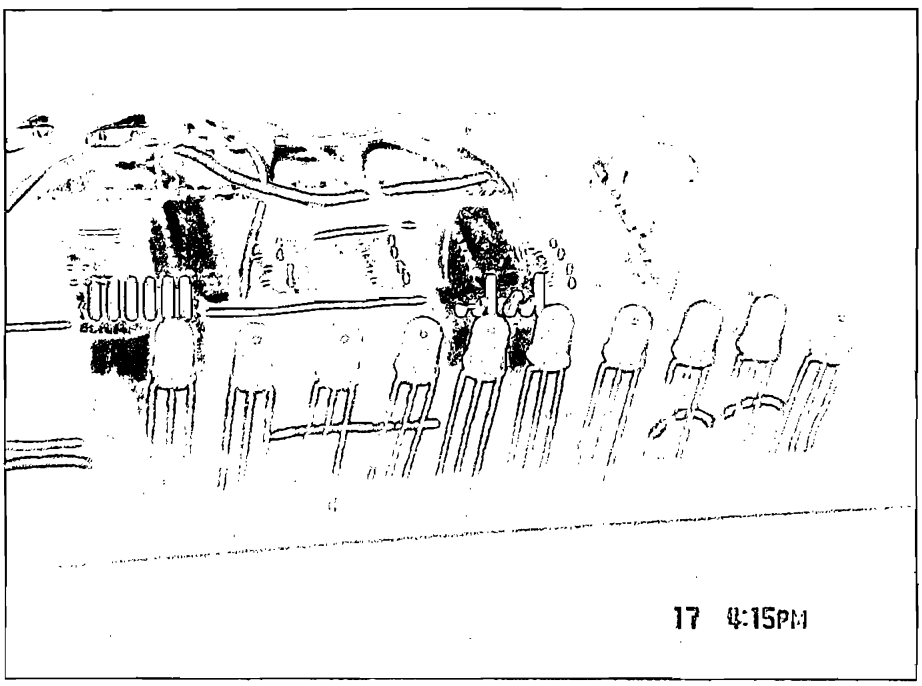


Figura 4.18

La segunda placa del MODEMAG, posee algunos leds que indican power, estatus y transmisor de datos, se conecto el MODEMAG individualmente para comprobar el funcionamiento de los des indicadores, este resultado se muestra a continuación.

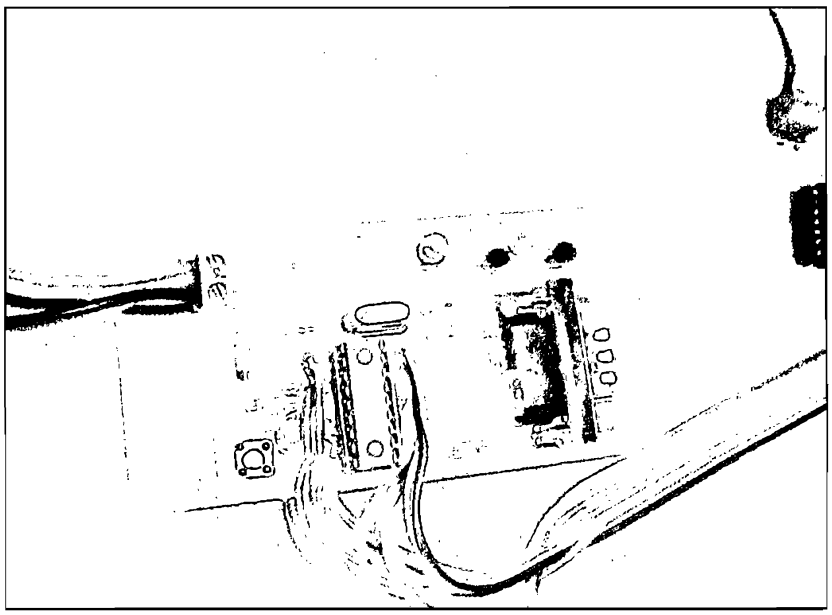


Figura 4.19

Se conecto serialmente al MODEMAG con la PC y se comprobó que el módulo entrega los datos para las plazas ocupadas bajo el formato (# de plaza, id del módulo), y plazas libres bajo el formato (id de módulo, # de plaza), como se muestra en al siguiente pantalla.

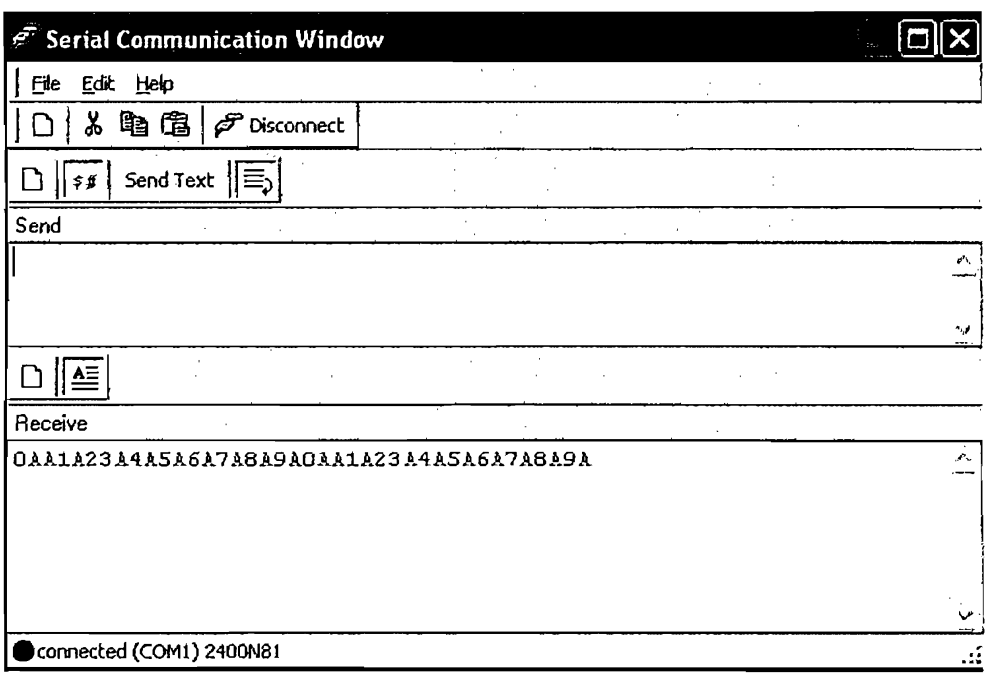


Figura 4.20

En este caso se encontraban ocupadas la plaza 1 y la plaza 4.

4.3.2 FUNCIONAMIENTO MODCON.

Se conecta el módulo para poder visualizar el mensaje inicial de bienvenida, como se muestra a continuación.

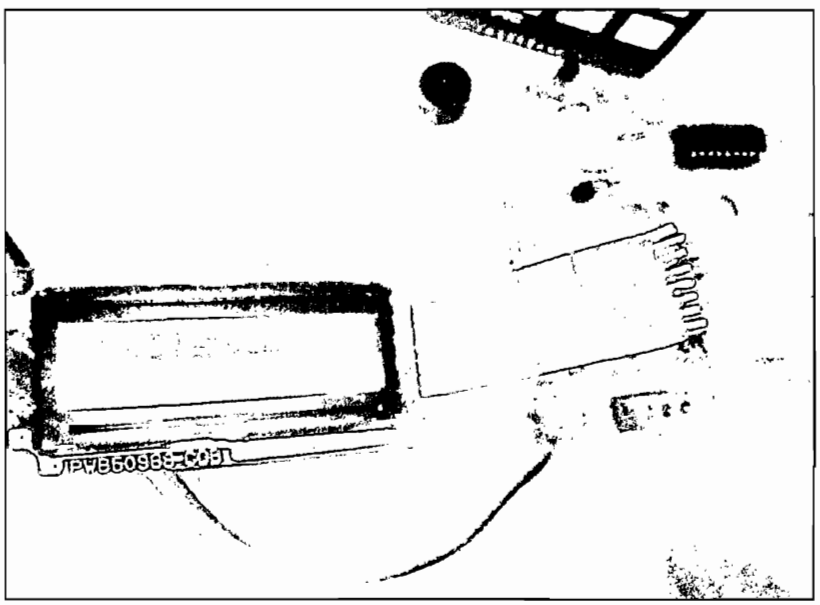


Figura 4.21

4.3.3 FUNCIONAMIENTO PANEL MIMICO.

4.3.3.1 Funcionamiento de la Mensajería

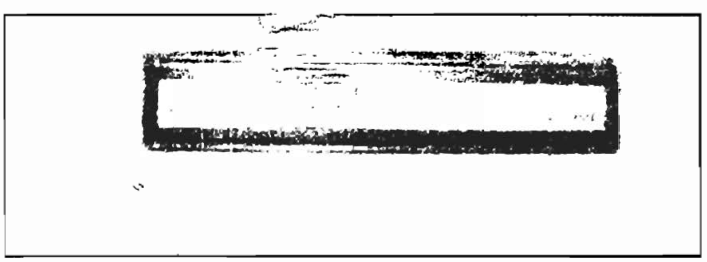


Figura 4.22

En la figura se visualiza el mensaje de bienvenida que se produce al inicializar el módulo, y permanecerá visible mientras el operador del sistema requiera mostrar otro mensaje.

4.3.3.2 Funcionamiento de las Matrices y leds indicadores.

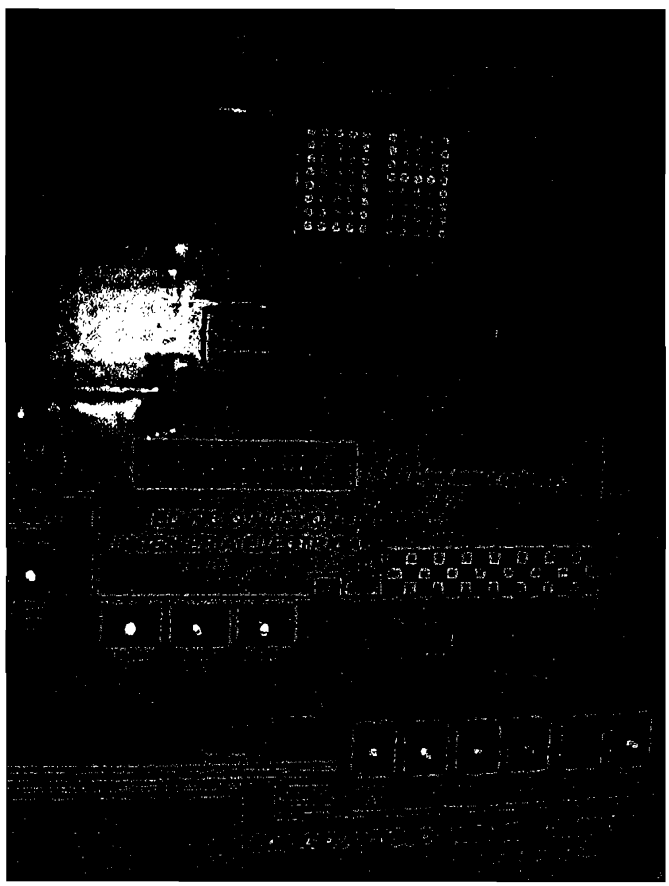


Figura 4.23

Como se visualiza en las matrices existen 4 plazas libres, lo mismo se observa en los leds del esquemático del parqueadero, los leds apagados indican que sus plazas respectivas se encuentran libres.

4.4 COSTOS DEL PROTOTIPO

4.4.1 TABLA 1

MODEMAG			
ELEMENTO	CANTIDAD	COSTO UNITARIO	COSTO TOTAL
Resistencia de 3.9 KO a 1/4 W	20	0.02	0.40
Compuertas 74LS02	3	0.40	1.20
PIC 16F84A	1	6.50	6.50
Cristal 4MHz	1	0.80	0.80
Capacitores cerámicos de 22pF	2	0.10	0.20
Conector DB9	1	0.45	0.45
DIP switch	2	0.85	1.70
Resistencias de 1 KO	10	0.02	0.20
Resistencias de 330	3	0.02	0.06
Resistencias de 33	2	0.02	0.04
Borneras de 2 pines	10	0.45	4.50
Borneras de 3 pines	10	0.75	7.50
Diodo 1N4148	1	0.05	0.05
Pulsador	1	0.50	0.50
Circuito impreso	1	30.00	30.00
LOOP Magnético	10	160.00	1,600.00
TOTAL			1,654.10

4.4.2 TABLA 2

PMI			
ELEMENTO	**CANTIDAD**	**COSTO UNITARIO**	**COSTO TOTAL**
Bornera de 2 pines	11	0.45	4.95
Bornera de 3 pines	1	0.75	0.75
Transistores 2N3904	12	0.25	3.00
Conectores de 8 pines	4	0.85	3.40
Pulsador	1	0.50	0.50
Potenciómetro de 10KO	1	0.90	0.90
Conector de 14 pines	1	0.75	0.75
PIC 16F877A	1	15.00	15.00
Resistencias de 330	8	0.02	0.16
Resistencias de 4.7KO	12	0.02	0.24
Cristal 4 MHz	1	0.80	0.80
Capacitores cerámicos de 22pF	2	0.10	0.20
Pila CR2032	1	1.20	1.20
RTC DS1307	1	6.40	6.40
Cristal 32768	1	1.30	1.30
Circuito Impreso	1	18.00	18.00
LEDS	10	0.10	1.00
LCD 2 X 40	1	18.00	18.00
Matrices de 8 x 5	2	3.00	6.00
Construcción Panel Informativo	1	8.00	8.00
TOTAL			**90.55**

4.4.3 TABLA 3

MODCON			
ELEMENTO	CANTIDAD	COSTO UNITARIO	COSTO TOTAL
Bornera de 2 pines	1	0.45	0.45
Bornera de 3 pines	1	0.75	0.75
Transistores 2N3904	4	0.25	1.00
Pulsador de llave	1	1.80	2.80
LCD 16 x 2	1	9.00	9.00
Teclado	1	6.00	6.00
Resistencias de 4.7 KO	6	0.02	0.12
Resistencias de 330	3	0.02	0.06
LEDS	2	0.10	0.20
LED alta velocidad	1	0.20	0.20
Cristales de 4 MHz	2	0.80	1.60
Capacitores cerámicos de 22pF	4	0.10	0.40
Pulsador	1	0.50	0.50
Fusible	1	0.80	0.80
Portafusible	1	0.20	0.20
Diodos 1N4148	2	0.05	0.10
PIC 16F84A	2	6.50	13.00
Circuito impreso	1	18.00	18.00
TOTAL			55.18

4.4.4 TABLA 4

MINT-PC			
ELEMENTO	CANTIDAD	COSTO UNITARIO	COSTO TOTAL
Bornera de 2 pines	2	0.45	0.90
Bornera de 3 pines	1	0.75	0.75
Transistor 2N3904	1	0.25	0.25
LED	2	0.10	0.20
Resistencias de 4.7 KO	5	0.02	0.10
Resistencias de 330	2	0.02	0.04
Resistencia de 33	1	0.02	0.02
Resistencia de 10 KO	1	0.02	0.02
Fusible	1	0.80	0.80
Portafusible	1	0.20	0.20
Cristal de 4 MHz	1	0.80	0.80
Capacitores cerámicos de 22pF	2	0.10	0.20
Pulsador	1	0.50	0.50
Diodo 1N4148	1	0.05	0.05
Conectores DB9	8	0.45	3.60
PIC 16F84A	1	6.50	6.50
Circuito impreso	1	18.00	18.00
TOTAL			32.93

4.4.5 TOTAL COSTO DEL PROTOTIPO

TOTAL GENERAL		
Módulo	Tabla	Valor
MODEMAG	TABLA 1	1,654.10
PMI	TABLA 2	90.55
MODCON	TABLA 3	55.18
MIT-PC	TABLA 4	32.93
VARIOS		50.00
TOTAL		1,882.76

CAPITULO 5

5. CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

Para el diseño e implementación de un sistema para monitorear las plazas de un parqueadero, se requieren de varias áreas de trabajo; es así, que para el PRINMOP, se necesitó de conocimientos de electrónica, manejo de pics, diseño de esquemáticos y PCB, además para lograr un sistema integral se realizó un programa en lenguaje de alto nivel (visual basic) el cual permite al usuario controlador del sistema tener una idea más clara de las plazas ocupadas y libres, así como también posee un registro del tiempo de ocupación.

La idea fundamental del PRINMOP es cumplir con la característica de modularidad, esto quiere decir que si el espacio físico del parqueadero crece, el sistema puede seguir funcionando con la inserción de nuevas tarjetas Modemag; además cada parte del prototipo está diseñada para cumplir tareas en forma independiente; por ejemplo: si el Modcon deja de funcionar los demás módulos no se ven afectados.

Para el diseño de los Modemag fue necesario implementar circuitería en base a compuertas lógicas, con la finalidad de proporcionarle características de seguridad, test, y activación o reserva manual. Esta circuitería adicional permite que si los loops magnéticos sufren daños o se encuentran en mantenimiento, el administrador del sistema pueda seguir controlando manualmente. Para una identificación visual de las plazas activadas mediante loop o manualmente se conecto leds bicolor que permiten mediante colores diferentes saber si la activación fue mediante loop (color rojo) o manualmente (color verde), o una combinación de los dos mediante la activación de los dos colores.

El módulo Modcon posee varias características de seguridad para el sistema en general, es así que al energizar el sistema, primero se debe ingresar la clave pregrabada para poder ingresar al sistema, luego existe la posibilidad de cambiar la clave para personalizar el ingreso al sistema. Otra característica de seguridad es la activación de la alarma sonora mediante un pulsador de llave; en caso de que el operador necesite abandonar su lugar de trabajo mediante la combinación de algunas teclas (tecla 7 y tecla 9) puede bloquear el módulo y para poder ingresar debe digitar la clave correspondiente.

En cuanto al PMI fue necesario la utilización de un PIC de mayor tamaño puesto que fue necesario manejar un mayor número de entradas – salidas y sistemas de mensajería con mayor tamaño.

Para la implementación en un área física de aproximadamente 50 vehículos la instalación de los módulos no llevaría más de 2 horas, pero los lazos magnéticos requieren de mayores cuidados para su instalación por lo que el tiempo aproximado de un solo loop magnético es de 3 horas, en vista de que se debe tener mayor prolijidad con la distribución de las espiras dentro del canal que alberga al loop, ya que si son mal distribuidas se presentan lecturas erradas y esto provoca fallas en el sistema en general.

Con el objetivo de abaratar costos se intentó construir el sensor de loop magnético, mediante la utilización de un oscilador con el XR2206, el cual nos proporcionó la frecuencia adecuada que recomiendan los fabricantes de este tipo de dispositivos, pero en el momento de unir el oscilador con el loop la señal oscilante perdía potencia; consultando manuales y realizando visitas a sitios donde se encontraban instalados este tipo de módulos, se determinó que el acoplamiento entre el sensor y el loop, no se lo realizaba por alguna red de acoplamiento o circuitos tanques como se lo había intentado, sino mediante un transformador de alta frecuencia, el cual no es comercializado en el país.

El Prinmop al ser un prototipo se desarrollo con las características mínimas necesarias razón por la cual se utilizó pics pequeños. Para un sistema de mayor envergadura se podría utilizar de la gama alta como el 16F877A que nos permite almacenar una gran cantidad de información y le daría al sistema una mayor versatilidad en cuanto a funciones adicionales. También ayudaría al manejo de sistemas adicionales como cámaras de video, sistemas contra incendio, sistemas antirrobo y ahorro de la iluminación.

Para la programación de los PICs se maneja un compilador en BASIC denominado PIC Basic Pro, el cual es una herramienta, que permite realizar programas con gran facilidad, ahorrando tiempo; por ejemplo para manejar el LCD mediante asembler, es necesario realizar varias subrutinas como la de inicialización, luego de transmisión de datos y la forma de mostrar los datos, lo que implicaba gran cantidad de código, en cambio con el compilador solo es necesario colocar la declaración **LCDOUT** , la operación a realizar y la información a mostrar.

Para lograr un sistema con mejores características comerciales, es necesaria la participación de profesionales de diferentes áreas, que permitan mejorar la presentación del sistema para atraer a los compradores, tanto en la parte física como en la parte de software.

5.2 RECOMENDACIONES

Para la instalación de los sensores magnéticos se recomienda la utilización de dispositivos que se alimenten de energía a 110 Voltios AC, en vista de que todas las instalaciones poseen puntos eléctricos de este tipo, lo que evitaría realizar un nuevo tendido eléctrico y la adquisición de fuentes adicionales, para dispositivos que trabajen con 24 Voltios DC por ejemplo.

Se recomienda que para que la señal sensada tenga un correcto funcionamiento es necesario cumplir con las normas adecuadas de instalación, como son: número de vueltas, forma y distribución de las espiras, y recubrir con el material adecuado para evitar futuros daños producidos por el medio ambiente.

El ingreso de los vehículos puede ser sensado mediante sistemas de detección utilizando cámaras digitales lo que nos permitiría saber con mayor exactitud las características de los vehículos que ingresen, siendo este un costo extra a la seguridad del sistema por una parte; pero sin embargo también representaría un mecanismo extra de control de los vehículos.

Es recomendable la utilización de pics con memoria EEPROM, para tener la capacidad de cambiar la clave pregrabada de fábrica y que permanezca si el fluido eléctrico se corta.

Para que el usuario no solo tenga información sobre las plazas del parqueadero sino también información adicional del lugar que visita se recomienda que los paneles informativos sean de mayor tamaño con ello mostrarán un mayor número de mensajes para el usuario.

El software de visualización puede ser depurado para hacer los ingresos de forma automática y no manual, utilizando por ejemplo un sistema de códigos de barras; la pantalla correspondiente a la visualización de las plazas puede contener varias pantallas de los diferentes lugares que se sensen con el sistema.

La implementación del sistema en un área de gran tamaño, requiere de la participación de profesionales no solo en el área de electrónica sino también en el área de sistemas para que el hardware pueda proporcionar datos al software y así el sistema genere más beneficios al usuario y al administrador.

Se recomienda para que el sistema tenga un menor costo, el diseño y construcción de los loops magnéticos, en vista de que este elemento es uno de los que encarecen el sistema en general. Para lograr este objetivo es recomendable trabajar con un sistema microcontrolado, un oscilador y un transformador de alta frecuencia, que una los dos electos anteriores. Por otro lado se podría utilizar otro tipo de sensor de menor costo, por ejemplo infrarrojos.

Para lugares donde se tenga clientes frecuentes, se recomienda la utilización de la nueva tecnología RFID, que en la actualidad se esta implementando para el control de productos en supermercados, el control de vehículos en lugares de uso frecuente, y también para el control de personas, mediante el implante de este tipo de tarjetas en el cuerpo humano.

REFERENCIAS BIBLIOGRAFICAS.

- GARCIA Javier, RODRIGUEZ José, BRAZALEZ Alfonso, Escuela Superior de Ingenieros Industriales, Universidad de Navarra" Aprenda Visual Basic 6.0 como si estuviera en primero", Navarra España.
- Micro Engineering Labs, "Manual Pic Basic PRO Compiler".
- GARCIA Jorge, GARCIA Juan, "Dispositivos y Componente Electrónicos", Lima Perú.
- Altium Limited, "Design capture, simulation and layout – an introduction"., 2002.
- ANGULO José M., "Microcontroladores PIC Diseño práctico de aplicaciones", Segunda Edición, Mc Graw Hill, España, 1999.
- REYES Carlos., "Aprenda rápidamente a programar Microcontroladores PIC" , Gráficas Ayerve c.a., Quito Ecuador, 2004.
- MALVINO Albert., "Principios de Electrónica", Quinta Edición, Mc Graw Hill, México, 1994.
- SAVANT C.J., RODEN Martin., CARPENTER Gordon., "Diseño Electrónico Circuitos y Sistemas ", Segunda Edición, Addison – Wealey Iberoamericana S.A., U.S.A. , 1992.

DIRECCIONES ELECTRONICAS.

- MECANIQUE. Página para descargar el programa MICROCODE
  http://www.mecanique.co.uk
- Programador de PICs IC Prog.
  http://www.IC-prog.com
- Herramientas de Programación.
  http://www.talkingelectronics.com
- Descarga del compilador Pic Basic Pro.
  http://www.sonsivri.com
- Microchip Electronics.
  http://www.microchip.com
- Ejemplos de programación.
  http://lawebdelprogramador.com
- Información sobre Microconroladores.
  http://www.todopic.com.ar
- Información de proyectos con PICs.
  http://www.pablin.com.ar
- Información sobre LCD
  http://wwwx-robotics.com
- Peoria del bus I2C.
  http://wwwcipres.cec.uchile.cl

ANEXOS

ANEXO 1 PROGRAMAS DE LOS DIFERENTES MODULOS

PROGRAMA MODEMAG.

```
'*****************************************************************
'* Name    : MODEMAG.BAS                                         *
'* Author  : Juan Luis Heredia                                   *
'* Notice  : Copyright (c) 2005 [select VIEW...EDITOR OPTIONS] *
'*         : All Rights Reserved                                 *
'* Date    : 10/05/2005                                          *
'* Versión : 1.0                                                 *
'*****************************************************************
INCLUDE "modedefs.bas"          ; librerias que contienen los modos de comunicación

;**************DEFINICION DE VARIABLES ****************************
plaza1 VAR porta.0              ;se renombra cada puerto con una variable
plaza2 VAR porta.1              ; por ejemplo, el puerto A.0 se llamará plaza1
plaza3 VAR porta.2
plaza4 VAR porta.3
plaza5 VAR porta.4
plaza6 VAR portb.1
plaza7 VAR portb.2
plaza8 VAR portb.3
plaza9 VAR portb.4
plaza10 VAR portb.5

led VAR portb.0                 ; El puerto b.0 se llamará "led"
x VAR BYTE                      ; se define al variable x
```

```
test1:                  ; inicio del test #1, cuando el módulo se conecta a
x=0                     ; la energía o se resetea

FOR x=1 TO 5            ; el led de estatus se encenderá 5 veces.
  HIGH led
  PAUSE 600
  LOW led
  PAUSE 600
NEXT

inicio:                 ; inicio del programa
  IF plaza1=0 THEN      ; si cumple la condición va a envioA
    CALL envioA         ; se llama a la subrutina envio A
  ELSE                  ; si no cumple la condición del if, va a envio1
    CALL envio1         ; se llama a la subrutina envio1
  ENDIF                 ; se finaliza el IF

  IF plaza2=0 THEN      ; si cumple la condición va a envioB
    CALL envioB         ; se llama a la subrutina envio A
  ELSE                  ; si no cumple la condición del IF, va a envio1
    CALL envio2         ; se llama a la subrutina envio1
  ENDIF                 ; se finaliza el IF

  IF plaza3=0 THEN      ; si cumple la condición va a envioC
    CALL envioC         ; se llama a la subrutina envio A
  ELSE                  ; si no cumple la condición del IF, va a envio1
    CALL envio3         ; se llama a la subrutina envio1
  ENDIF                 ; se finaliza el IF

  IF plaza4=0 THEN      ; si cumple la condición va a envioD
    CALL envioD         ; se llama a la subrutina envio A
```

```
ELSE                    ; si no cumple la condición del IF, va a envio1
  CALL envio4           ; se llama a la subrutina envio1
ENDIF                   ; se finaliza el IF

IF plaza5=0 THEN        ; si cumple la condición va a envioE
  CALL envioE           ; se llama a la subrutina envio A
ELSE                    ; si no cumple la condición del IF, va a envio1
  CALL envio5           ; se llama a la subrutina envio1
ENDIF                   ; se finaliza el IF

IF plaza6=0 THEN        ; si cumple la condición va a envioF
  CALL envioF           ; se llama a la subrutina envio A
ELSE                    ; si no cumple la condición del IF, va a envio1
  CALL envio6           ; se llama a la subrutina envio1
ENDIF                   ; se finaliza el IF

IF plaza7=0 THEN        ; si cumple la condición va a envioG
  CALL envioG           ; se llama a la subrutina envio A
ELSE                    ; si no cumple la condición del IF, va a envio1
  CALL envio7           ; se llama a la subrutina envio1
ENDIF                   ; se finaliza el IF

IF plaza8=0 THEN        ; si cumple la condición va a envioH
  CALL envioH           ; se llama a la subrutina envio A
ELSE                    ; si no cumple la condición del IF, va a envio1
  CALL envio8           ; se llama a la subrutina envio1
ENDIF                   ; se finaliza el IF

IF plaza9=0 THEN        ; si cumple la condición va a enviol
  CALL enviol           ; se llama a la subrutina envio A
ELSE                    ; si no cumple la condición del IF, va a envio1
```

```
    CALL envio9              ; se llama a la subrutina envio1
  ENDIF                      ; se finaliza el IF

  IF plaza10=0 THEN          ; si cumple la condición va a envioJ
    CALL envioJ              ; se llama a la subrutina envio A
  ELSE                       ; si no cumple la condición del IF, va a envio1
    CALL envio10             ; se llama a la subrutina envio1
  ENDIF                      ; se finaliza el IF


test2:                       ; inicio del test # 2

  FOR x=1 to 2               ; luego del barrido de los puertos, el led de
                             ; estatus se encenderá dos veces, para
    HIGH led                 ; indicar el funcionamiento normal
    PAUSE 200
    LOW led
    PAUSE 300
  NEXT

  GOTO inicio                ; si ningún puerto se ha activado
                             ; hace nuevo barrido

; ******* INICIO SUBRUTINAS DE ENVIO - PLAZAS OCUPADAS ********************

envioA:                      ; subrutina de envio si plaza 1 se ocupa
  SEROUT portb.6,N2400,["A0"] ; se envía los caracteres A0 por el puertob.6
  PAUSE 50                   ; se realiza una pausa
  RETURN                     ; regresa a la parte del programa donde se
                             ; realizo la llamada a la subrutina
envioB:
```

```
   SEROUT portb.6,N2400,["A1"]
   PAUSE 50
   RETURN

envioC:
   SEROUT portb.6,N2400,["A2"]
   PAUSE 50
   RETURN

envioD:
   SEROUT portb.6,N2400,["A3"]
   PAUSE 50
   RETURN

envioE:
   SEROUT portb.6,N2400,["A4"]
   PAUSE 50
   RETURN

envioF:
   SEROUT portb.6,N2400,["A5"]
   PAUSE 50
   RETURN

envioG:
   SEROUT portb.6,N2400,["A6"]
   PAUSE 50
   RETURN

envioH:
   SEROUT portb.6,N2400,["A7"]
```

```
   PAUSE 50
   RETURN

envio1:
   SEROUT portb.6,N2400,["A8"]
   PAUSE 50
   RETURN

envioJ:
   SEROUT portb.6,N2400,["A9"]
   PAUSE 50
   RETURN

; ******* INICIO SUBRUTINAS DE ENVIO  - PLAZAS LIBRES ********************

envio1:                          ; subrutina de envio si plaza 1 esta libre
   SEROUT portb.6,N2400,["0A"]   ; se envía los caracteres 0A por el puertob.6
   PAUSE 50                      ; se realiza una pausa
   RETURN                        ; regresa a la parte del programa donde se
                                 ; realizo la llamada a la subrutina
envio2:
   SEROUT portb.6,N2400,["1A"]
   PAUSE 50
   RETURN

envio3:
   SEROUT portb.6,N2400,["2A"]
   PAUSE 50
   RETURN

envio4:
```

```
   SEROUT portb.6,N2400,["3A"]
   PAUSE 50
   RETURN

envio5:
   SEROUT portb.6,N2400,["4A"]
   PAUSE 50
   RETURN

envio6:
   SEROUT portb.6,N2400,["5A"]
   PAUSE 50
   RETURN

envio7:
   SEROUT portb.6,N2400,["6A"]
   PAUSE 50
   RETURN

envio8:
   SEROUT portb.6,N2400,["7A"]
   PAUSE 50
   RETURN

envio9:
   SEROUT portb.6,N2400,["8A"]
   PAUSE 50
   RETURN

envio10:
   SEROUT portb.6,N2400,["9A"]
```

```
PAUSE 50
RETURN

END                                    ; fin del programa
```

PROGRAMA MODCON TECLADO.

```
'*****************************************************************
'* Name    : TECLADO MODCON.BAS                                  *
'* Author  : Juan Luis Heredia                                   *
'* Notice  : Copyright (c) 2005                                  *
'*         : All Rights Reserved                                 *
'* Date    : 28/05/2005                                          *
'*****************************************************************
INCLUDE "modedefs.bas"          ; librerias que contienen los modos de comunicación

;********** DEFINICION DE VARIABLES *****************
TECLA VAR BYTE                  ; variable para almacenar la tecla pulsada
CICLO VAR BYTE                  ; variable para determinar la salida de un proceso
R VAR BYTE                      ; variable para repeticiones
BIP VAR PORTA.3                 ; se renombra el puertoa.3 como BIP
LED VAR PORTA.2                 ; se renombra el puertoa.2 como LED
SALIDA VAR PORTA.4              ; define el puerto de la salida serial

A VAR PORTB.0                   ; nombre de los pines de las filas
B VAR PORTB.1
```

```
C VAR PORTB.2
D VAR PORTB.3

UNO VAR PORTB.4          ; nombre de los pines para las columnas
DOS VAR PORTB.5
TRES VAR PORTB.6
CUATRO VAR PORTB.7

CLAVE1 VAR BYTE          ; variable para almacenar la primera clave
CLAVE2 VAR BYTE          ; variable para almacenar la segunda clave
CLAVE3 VAR BYTE          ; variable para almacenar la tercera clave
CLAVE4 VAR BYTE          ; variable para almacenar la cuarta clave

; ********** CLAVE PRE GRABADA **************************

CLAVE1="1" : CLAVE2="2" : CLAVE3="3":CLAVE4="4"

INICIANDO:               ; programa del led para ver el funcionamiento
FOR R=1 TO 2
  HIGH LED :HIGH BIP:PAUSE 500
  LOW LED:LOW BIP:PAUSE 300
NEXT
;*****************************************************************
RESET:
  FOR R=1 TO 5
    HIGH LED:HIGH BIP:PAUSE 50
    LOW LED :LOW BIP:PAUSE 50
  NEXT
CICLO=1                  ; se carga la variable CICLO

PRINCIPAL:               ; inicio de la parte principal del programa
```

```
WHILE ciclo =1
   FOR R=1 TO 5
   SEROUT salida,n2400,["C"]: PAUSE 400
   NEXT
   GOTO TECLAUNO              ;ir a comparar claves
WEND

NORMAL:
WHILE CICLO=2
   PAUSE 500
   SEROUT salida,n2400,["N"]: PAUSE 400
   HIGH A: HIGH B: LOW C: HIGH  D :PAUSE 200
   IF (TRES=0)AND(UNO=0)THEN RESET          ;corresponde a las teclas 7 y 9
   HIGH A: HIGH B: HIGH C: LOW  D :PAUSE 200  ; sensar solo la fila C
   IF uno = 0 THEN MENSAJES                   ; corresponde a la tecla 9
                                              ; para ir a grabar

ALARMA:                          ; inicio para generar una alarma
   IF Porta.0 = 0 THEN           ; se genera una condición para porta.0
     SOUND porta.1,[100,10,50,10]:PAUSE 100 ; generación de tonos
     SEROUT salida,n2400,["E"]: PAUSE 200
     HIGH LED:HIGH BIP:PAUSE 50
     LOW LED :LOW BIP:PAUSE 50
     GOTO ALARMA
   ELSE
     GOTO NORMAL
   ENDIF
 WEND
GOTO NORMAL
```

```
; ***********COMPARACION DE CLAVES ******************************
TECLAUNO:
   GOSUB BARRIDO                ; ir a barrido y retornar con un valor
   GOSUB PTECLA                 ; envía aun programa antirrebote para soltar
tecla
   IF TECLA = CLAVE1 THEN TECLADOS ; si el numero es igual a CLAVE1
   GOTO FALSO                   ; caso contrario ir a alzo FALSO
TECLADOS:
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   GOSUB BARRIDO:GOSUB PTECLA
   IF TECLA=CLAVE2 THEN TECLATRES
   GOTO FALSO1
TECLATRES:
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   GOSUB BARRIDO:GOSUB PTECLA
   IF TECLA=CLAVE3 THEN TECLACUATRO
   GOTO FALSO2
TECLACUATRO:
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   GOSUB BARRIDO:GOSUB PTECLA
   IF TECLA=CLAVE4 THEN MENSAJE
   GOTO FALSO3
MENSAJE:
  SEROUT salida,n2400,[TECLA]: PAUSE 100
  FOR R=1 TO 3                  ; Dos pitidos indica clave correcta
     PAUSE 100
     HIGH LED:HIGH BIP:PAUSE 100
     LOW LED: LOW BIP
  NEXT
FOR r=1 TO 4
   SEROUT salida,n2400,["A"]: PAUSE 400
```

```
NEXT
   HIGH A: HIGH B: HIGH C: LOW  D :PAUSE 300     ; sensar solo la fila C
   IF tres = 0 THEN GOSUB GRABAUNO  ; corresponde a la tecla enter para ir a
                                    ; grabar
CICLO=2
GOTO NORMAL

; ************************************************
GRABAUNO:                          ;programa para cambiar la clave
   FOR r=1 TO 4
     SEROUT salida,n2400,["G"]: PAUSE 100
   NEXT
   GOSUB PTECLA:HIGH LED           ; espera a que suelte las teclas
   GOSUB BARRIDO:GOSUB PTECLA      ; ir a barrido y retornar a un antirrebote
   HIGH LED                        ; mantiene encendido el led
   CLAVE1=TECLA                    ; guarda el valor de tecla
GRABADOS:
   GOSUB BARRIDO:GOSUB PTECLA  ; ir a barrido y retornar con a un antirrebote
   HIGH LED                    ; mantiene encendido el led
   CLAVE2=TECLA                ; guarda el valor de TECLA
GRABATRES:
   GOSUB BARRIDO:GOSUB PTECLA  ; ir a barrido y retornar a un antirrebote
   HIGH LED                    ; mantiene encendido el led
   CLAVE3=TECLA                ; guarda en valor de tecla
GRABACUATRO:
   GOSUB BARRIDO:GOSUB PTECLA  ; ir a barrido y retornar con a un antirrebote
   HIGH LED                    ; mantiene encendido el led
   CLAVE4=TECLA                ; guarda el valor de tecla
   SEROUT salida,n2400,["D"]: PAUSE 100
RETURN                 ; ir a reset para cambiar el nuevo valor en las variables
BARRIDO:
```

```
   LOW A                                ;sensar  fila A
IF UNO =0 THEN TECLA= "1":RETURN        ; tecla pulsada retorne cargada con 1
IF DOS =0 THEN TECLA="2":RETURN         ; tecla pulsada retorne cargada con 2
IF TRES =0 THEN TECLA="3":RETURN        ; tecla pulsada retorne cargada con 3
IF CUATRO =0 THEN TECLA="A":RETURN      ; tecla pulsada retorne cargada con A
   HIGH A
   LOW B                                ; sensar fila B
IF UNO =0 THEN TECLA="4":RETURN         ; tecla pulsada retorne cargada con 4
IF DOS =0 THEN TECLA="5":RETURN         ; tecla pulsada retorne cargada con 5
IF TRES =0 THEN TECLA="6":RETURN        ; tecla pulsada retorne cargada con 6
IF CUATRO =0 THEN TECLA="B":RETURN      ; tecla pulsada retorne cargada con B
   HIGH B
   LOW C
IF UNO =0 THEN TECLA="7":RETURN         ; tecla pulsada retorne cargada con 7
IF DOS =0 THEN TECLA="8":RETURN         ; tecla pulsada retorne cargada con 8
IF TRES =0 THEN TECLA="9":RETURN        ; tecla pulsada retorne cargada con 9
IF CUATRO =0 THEN TECLA="C":RETURN      ; tecla pulsada retorne cargada con C
   HIGH C
   LOW D
IF UNO =0 THEN TECLA="*":RETURN         ; tecla pulsada retorne cargada con *
IF DOS =0 THEN TECLA="0":RETURN         ; tecla pulsada retorne cargada con 0
IF TRES =0 THEN TECLA="#":RETURN        ; tecla pulsada retorne cargada con #
IF CUATRO =0 THEN TECLA="D":RETURN      ; tecla pulsada retorne cargada con B
   HIGH D
GOTO BARRIDO
;**********PROGRAM DE ANTIRREBOTE DE LAS TECLAS **********

PTECLA:
   HIGH LED: HIGH BIP:                  ; al pulsar la tecla se genera un sonido
   PAUSE 100                            ; pausa de 100 ms
   LOW LED : LOW BIP                    ; apaga el sonido y el led
```

```
ESPACIO:                                    ; programa antirrebotes
   IF UNO=0 THEN ESPACIO                    ; si la tecla sigue pulsada ir a espacio
   IF DOS=0 THEN ESPACIO                    ; si la tecla sigue pulsada ir a espacio
   IF TRES=0 THEN ESPACIO                   ; si la tecla sigue pulsada ir a espacio
   IF CUATRO=0 THEN ESPACIO                 ; si la tecla sigue pulsada ir a espacio
   PAUSE 25
   RETURN                                   ; retorna al soltar las teclas
;******** LAZOS PARA LAS TECLA ERRONEAS **********************
FALSO:
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   GOSUB BARRIDO: GOSUB PTECLA              ; estas teclas no comparan ninguna
FALSO1:                                     ; clave solo espera que termine de
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   GOSUB BARRIDO: GOSUB PTECLA              ; pulsar las 4 teclas y no hace nada
FALSO2:
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   GOSUB BARRIDO: GOSUB PTECLA
FALSO3:
   SEROUT salida,n2400,[TECLA]: PAUSE 100
   FOR   R=1 TO 10                          ; 10 pitos indica clave incorrecta
      PAUSE 150
      HIGH LED: HIGH BIP: PAUSE 150
      LOW LED : LOW BIP
   NEXT
FOR r=1 TO 4
   SEROUT salida,n2400,["I"]: PAUSE 4000
NEXT
CICLO=1                                     ;se carga el valor de CICLO
GOTO PRINCIPAL

sonido:
```

```
  SOUND porta.1,[100,10,50,10]:PAUSE 100
  RETURN

MENSAJES:
FOR R=1 TO 8
  SEROUT salida,n2400,["M"]: PAUSE 200
NEXT
 GOSUB PTECLA:HIGH LED              ;espera a que suelte las teclas
 GOSUB BARRIDO:GOSUB PTECLA         ;ir a barrido y retornar a un antirrebote

REPETIR:
  FOR R=1 TO 3
    IF TECLA="1" THEN SEROUT salida,n2400,["Z"]: PAUSE 100
  NEXT
  FOR r= 1 TO 3
    IF TECLA="2" THEN SEROUT salida,n2400,["Y"]: PAUSE 100
  NEXT
  FOR r= 1 TO 3
    IF TECLA="3" THEN SEROUT salida,n2400,["X"]: PAUSE 100
  NEXT
  GOSUB BARRIDO:GOSUB PTECLA         ;ir a barrido y retornar a un antirrebote
  IF TECLA="0" THEN
    SEROUT salida,n2400,["N"]: PAUSE 100
    GOTO NORMAL
  ELSE
    GOTO REPETIR
  ENDIF
END
```

PROGRAMA MODCON LCD.

```
'****************************************************************
'* Name    : LCD MODCON.BAS                                     *
'* Author  : Juan Luis Heredia                                  *
'* Notice  : Copyright (c) 2005 [select VIEW...EDITOR OPTIONS] *
'*         : All Rights Reserved                                *
'* Date    : 28/05/2005                                         *
'* Version : 1.0                                                *
'* Notes   :                                                    *
'****************************************************************
INCLUDE "modedefs.bas"          ; librerias que contienen los modos de comunicación

PAUSE 200                       ; retardo para esperar que funcione el LCD
;********** DEFINICION DE VARIABLES *****************

datos VAR WORD                  ; crea la variable datos
alarma VAR BYTE                 ; crear la variable alarma de 255
estado VAR BYTE                 ; crear la variable estado de 255
x VAR BYTE                      ; crear la variable x de 255
y VAR BYTE                      ; crear la variable x de 255


ENTRADA VAR PORTB.5
; **********inicio del programa *****************

LCDOUT $fe,1 ,"*** Bienvenido ***"   ; se limpia el LCD y se envía el mensaje
PAUSE 2000                      ; se realiza una pausa
LCDOUT $fe,1                    ; se limpia el LCD
```

```
inicio:
SERIN portb.1,N2400,datos      ; toma los datos del portb.1 y almacena en datos
IF datos="C" THEN clave        ; si el contenido de datos es "C" vamos a clave
IF datos="E" THEN emergencia   ; si datos es "E" vamos a emergencia
IF datos="N" THEN normal       ; si datos es "N" vamos a normal
IF datos="I" THEN incorrecto   ; si datos es "I" vamos a incorrecto
IF datos="A" THEN autorizado   ; si datos es "A" vamos a autorizado
IF datos="G" THEN grabar       ; si datos es "G" vamos a grabar
IF datos="D" THEN grabado      ; si datos es "D" vamos a grabado
IF datos="M" THEN mensajes     ; si datos es "M" vamos a mensajes
IF datos="Z" THEN mensaje1     ; si datos es "Z" vamos a mensaje1
IF datos="Y" THEN mensaje2     ; si datos es "Y" vamos a mensaje2
IF datos="X" THEN mensaje3     ; si datos es "X" vamos a mensaje3

GOTO inicio

clave:                         ; inicio de la subrutina clave
   LCDOUT $fe,1
   LCDOUT $fe,$80 ,"Ingrese la Clave"      ; se coloca el cursor en la posición 80
                                           ; y se envía el mensaje
   PAUSE 2500
   LCDOUT $fe,$C5                  ; se coloca el cursor en la segunda línea
   SERIN portb.1,N2400,datos:PAUSE 10      ; toma los datos del puertob.1
   LCDOUT "*"                              ; escribe el caracter "*"
   SERIN portb.1,N2400,datos:PAUSE 10
   LCDOUT "*"
   SERIN portb.1,N2400,datos:PAUSE 10
   LCDOUT "*"
   SERIN portb.1,N2400,datos:PAUSE 10
   LCDOUT "*"
   PAUSE 100
```

```
   LCDOUT $fe,1                    ; borra el LCD
GOTO inicio                        ; regresa al inicio del programa

normal:                            ; inicio subrutina normal
   LCDOUT $fe,1 ," Funcionamiento "
   LCDOUT $fe,$C0 ,"    Normal    "

   PAUSE 2500
   LCDOUT $FE,1                    ; Limpia la pantalla
   GOTO inicio

emergencia:                        ; inicio subrutina emergencia
   LCDOUT $FE,$7                   ; configura para desplazamiento izquierda
   LCDOUT $FE,1                    ; Limpia la pantalla
   LCDOUT $FE,$90
      FOR x=0 TO 22
         lookup x,["*** EMERGENCIA ***   "],alarma
         LCDOUT, alarma
         PAUSE 150
      NEXT

   LCDOUT $FE,$6                   ; configura para no desplazamiento
   LCDOUT $FE,1                    ; Limpia la pantalla
   GOTO inicio
incorrecto:                        ; inicio subrutina incorrecto
   LCDOUT $fe,1
   LCDOUT $fe,$80 ,"Clave incorrecta"

   PAUSE 2000
   LCDOUT $fe,1
   GOTO inicio
```

```
autorizado:                          ; inicio subrutina autorizado
   LCDOUT $fe,1
   LCDOUT $fe,$80 ,"Clave correcta"

   PAUSE 2000
   LCDOUT $fe,1
   GOTO inicio
grabar:                              ; inicio subrutina grabar
   LCDOUT $fe,1
   LCDOUT $fe,$80 ,"Ingrese nueva clave"

   PAUSE 2000
   LCDOUT $fe,1
   GOTO inicio
grabado:                             ; inicio subrutina grabar
   LCDOUT $fe,1
   LCDOUT $fe,$80 ,"Clave grabada"

   PAUSE 2000
   LCDOUT $fe,1
   GOTO inicio


mensajes:                            ; inicio subrutina mensajes
   LCDOUT $fe,1
   LCDOUT $fe,$80 ,"Ingrese el # mensaje"

   PAUSE 2000
   LCDOUT $fe,$C5
   SERIN portb.1,N2400,datos:PAUSE 10
   LCDOUT datos: PAUSE 100
```

```
   LCDOUT $fe,1
   GOTO inicio
mensaje1:                          ; inicio subrutina mensaje1
   LCDOUT $fe,1
   LCDOUT $fe,$80 ,"Mensaje uno"

   PAUSE 3000
   GOTO inicio
mensaje2:                          ; inicio subrutina mensaje2
   LCDOUT $fe,1
   LCDOUT $fe,$80 ,"Mensaje dos"

   PAUSE 3000
   GOTO inicio

mensaje3:                          ; inicio subrutina mensaje3
   LCDOUT $fe,1
   LCDOUT $fe,$80 ,"Mensaje tres"

   PAUSE 3000
   GOTO inicio
END                                ; fin del programa
```

PROGRAMA MINT – PC

```
'*****************************************************************
'* Name    : MINT - PC.BAS                                       *
'* Author  : Juan Luis Heredia                                   *
'* Notice  : Copyright (c) 2005 [select VIEW...EDITOR OPTIONS] *
```

```
'*       : All Rights Reserved                                *
'* Date   : 09/06/2005                                       *
'* Version : 1.0                                             *
'* Notes  :                                                  *
'*       :                                                   *
'************************************************************
INCLUDE "modedefs.bas"        ; librerias que contienen los modos de comunicación

;********** DEFINICION DE VARIABLES *****************
TXPC VAR PORTA.0              ; se define cada puerto con un nombre
RXPC VAR PORTA.1              ; que identifica la función a cumplir
RXMCONTROL VAR PORTA.2
RXPMIMICO VAR PORTA.3
TXPMIMICO VAR PORTA.4
RXMODEMAG1 VAR PORTB.0
RXMODEMAG2 VAR PORTB.1

LED VAR PORTB.7
datos0 VAR BYTE               ; se definen as variables que almacenan
datos1 VAR BYTE               ; los datos
datos2 VAR BYTE
R VAR BYTE
contador VAR BYTE

lazo:
SERIN RXMCONTROL,N2400,datos1         ; validación clave
  IF datos1 ="N" THEN                 ; si no recibe el caracter N
    GOTO inicio                       ; el programa permanece en un lazo
  ELSE
    GOTO lazo
  ENDIF
```

```
inicio:
FOR R=1 TO 3
   HIGH LED
   PAUSE 100
   LOW LED
   PAUSE 50
NEXT

   datos0=0                    ; se inicializa las variables
   datos1=0
   datos2=0
   contador =0
recibir:

   FOR R=1 TO 10                          ; lazo de 1 a 10
      SERIN RXMODEMAG1,N2400,datos0       ; toma los datos del modemag
      SERIN RXMODEMAG1,N2400,datos1       ; toma los datos del modemag
       IF datos0 = "0" THEN contador =1   ; si la plaza 1 esta libre aumenta
       IF datos0 = "1" THEN contador = contador +1   ; el contador, si la plaza2
       IF datos0 = "2" THEN contador = contador +1   ; esta libre aumenta el
       IF datos0 = "3" THEN contador = contador +1   ; contador y así
       IF datos0 = "4" THEN contador = contador +1   ; sucesivamente con las
       IF datos0 = "5" THEN contador = contador +1   ; demás plazas.
       IF datos0 = "6" THEN contador = contador +1
       IF datos0 = "7" THEN contador = contador +1
       IF datos0 = "8" THEN contador = contador +1
       IF datos0 = "9" THEN contador = contador +1
      SEROUT TXPC,N2400,[datos0]           ; transmite los datos hacia la PC
      SEROUT TXPC,N2400,[datos1]
```

```
   NEXT
IF contador = 10 THEN                          ; si el valor de CONTADOR es  = 10
   PAUSE 300                                   ; enviar el caracter P
   SEROUT TXPMIMICO,N2400,["P"]
ELSE
   PAUSE 300
   SEROUT TXPMIMICO,N2400,[#contador]          ; caso contrario enviar el valor
ENDIF                                          ; decimal de CONTADOR

SERIN RXMCONTROL,N2400,datos2
   FOR R= 1 TO 3
   IF datos2 = "X" THEN  SEROUT TXPMIMICO,N2400,[datos2] ; dependiendo del
                                                         ; valor
   IF datos2 = "Y" THEN  SEROUT TXPMIMICO,N2400,[datos2] ; de datos2 se envía
                                                         ; este valor
   IF datos2 = "Z" THEN  SEROUT TXPMIMICO,N2400,[datos2] ; al panel
   IF datos2 = "E" THEN  SEROUT TXPMIMICO,N2400,[datos2]
   NEXT
   GOTO inicio
END
```

PROGRAMA PMI

```
'*****************************************************************
'* Name    : PMI.BAS                                             *
'* Author  : Juan Luis Heredia                                   *
'* Notice  : Copyright (c) 2005 [select VIEW...EDITOR OPTIONS] *
'*         : All Rights Reserved                                 *
'* Date    : 16/10/2005                                         *
'* Version : 1.0                                                 *
```

```
'* Notes   :                                                    *
'*         :                                                    *
'***************************************************************
INCLUDE "modedefs.bas"          ; librerias que contienen los modos de comunicación

cmcon=7                         ; cambia a modo digital todo el puerto A
;********** DEFINICION DE VARIABLES *****************

trisb=0             ; todo el puerto B es configurado como salida
trisc=0             ; todo el puerto C es configurado como salida
trisd=0             ; todo el puerto D es configurado como salida

x VAR BYTE
datos VAR BYTE

; **** ASIGNACION PIN ENABLE PARA EL LCD ****************
DEFINE LCD_EREG PORTC           ; define pin para conectar el bit Enable
DEFINE LCD_EBIT 5               ; EN EL PUERTO C5


; *************** INICIO DEL PROGRAMA ********************
inicio1:
LCDOUT $fe,1 ,"         *** BIENVENIDO ***"     ; se limpia el LCD y se envía el
                                                ; mensaje
LCDOUT $FE,$C0,"**** PARQUEADERO POSTERIOR ELECTRICA ***"
PAUSE 10000                                     ; se realiza una pausa

LCDOUT $FE,1

inicio4:
```

```
SERIN PORTD.7,N2400,datos              ; se toma los datos del puerto d.7

inicio2:
IF datos = "0" THEN numero0     ; dependiendo de la información de la variable
IF datos = "1" THEN numero1     ; datos, se direcciona a la subrutina adecuada
IF datos = "2" THEN numero2
IF datos = "3" THEN numero3
IF datos = "4" THEN numero4
IF datos = "5" THEN numero5
IF datos = "6" THEN numero6
IF datos = "7" THEN numero7
IF datos = "8" THEN numero8
IF datos = "9" THEN numero9
GOTO inicio4

END


numero0:

FOR x=1 TO 40                   ; subrutina genera cero - cero
portd=%00000001:portb=%11111111:PAUSE 2
portd=%00000010:portb=%10000001:PAUSE 2
portd=%00000100:portb=%10000001:PAUSE 2
portd=%00001000:portb=%10000001:PAUSE 2
portd=%00010000:portb=%11111111:PAUSE 2
portd=%00000000:
portc=%00000001:portb=%11111111:PAUSE 2
portc=%00000010:portb=%10000001:PAUSE 2
portc=%00000100:portb=%10000001:PAUSE 2
portc=%00001000:portb=%10000001:PAUSE 2
```

```
portc=%00010000:portb=%11111111:PAUSE 2
portc=%00000000
NEXT

GOTO inicio2

numero1:

FOR x=1 TO 40                ; subrutina genera cero - uno
portd=%00000001:portb=%11111111:PAUSE 2
portd=%00000010:portb=%10000001:PAUSE 2
portd=%00000100:portb=%10000001:PAUSE 2
portd=%00001000:portb=%10000001:PAUSE 2
portd=%00010000:portb=%11111111:PAUSE 2
portd=%00000000:
portc=%00000001:portb=%11111111:PAUSE 2
portc=%00000010:portb=%01000001:PAUSE 2
portc=%00000100:portb=%00100000:PAUSE 2
portc=%00001000:portb=%00010000:PAUSE 2
portc=%00010000:portb=%00001000:PAUSE 2
portc=%00000000
NEXT

GOTO inicio2

numero2:

FOR x=1 TO 40                ; subrutina genera cero - dos
portd=%00000001:portb=%11111111:PAUSE 2
portd=%00000010:portb=%10000001:PAUSE 2
portd=%00000100:portb=%10000001:PAUSE 2
```

```
portd=%00001000:portb=%10000001:PAUSE 2
portd=%00010000:portb=%11111111:PAUSE 2
portd=%00000000:
portc=%00000001:portb=%11110001:PAUSE 2
portc=%00000010:portb=%10010001:PAUSE 2
portc=%00000100:portb=%10010001:PAUSE 2
portc=%00001000:portb=%10010001:PAUSE 2
portc=%00010000:portb=%10011111:PAUSE 2
portc=%00000000
NEXT

GOTO inicio2


numero3:

FOR x=1 TO 40                    ; subrutina genera cero - tres
portd=%00000001:portb=%11111111:PAUSE 2
portd=%00000010:portb=%10000001:PAUSE 2
portd=%00000100:portb=%10000001:PAUSE 2
portd=%00001000:portb=%10000001:PAUSE 2
portd=%00010000:portb=%11111111:PAUSE 2
portd=%00000000:
portc=%00000001:portb=%11111111:PAUSE 2
portc=%00000010:portb=%10010001:PAUSE 2
portc=%00000100:portb=%10010001:PAUSE 2
portc=%00001000:portb=%10010001:PAUSE 2
portc=%00010000:portb=%10010001:PAUSE 2
portc=%00000000
NEXT
```

```
GOTO inicio2


numero4:

FOR x=1 TO 40                    ; subrutina genera cero - cuatro
portd=%00000001:portb=%11111111:PAUSE 2
portd=%00000010:portb=%10000001:PAUSE 2
portd=%00000100:portb=%10000001:PAUSE 2
portd=%00001000:portb=%10000001:PAUSE 2
portd=%00010000:portb=%11111111:PAUSE 2
portd=%00000000
portc=%00000001:portb=%11111111:PAUSE 2
portc=%00000010:portb=%00010000:PAUSE 2
portc=%00000100:portb=%00010000:PAUSE 2
portc=%00001000:portb=%00010000:PAUSE 2
portc=%00010000:portb=%11110000:PAUSE 2
portc=%00000000
NEXT

GOTO inicio2


numero5:

FOR x=1 TO 40                    ;subrutina genera cero - cinco
portd=%00000001:portb=%11111111:PAUSE 2
portd=%00000010:portb=%10000001:PAUSE 2
portd=%00000100:portb=%10000001:PAUSE 2
portd=%00001000:portb=%10000001:PAUSE 2
portd=%00010000:portb=%11111111:PAUSE 2
```

```
portd=%00000000
portc=%00000001:portb=%10011111:PAUSE 2
portc=%00000010:portb=%10010001:PAUSE 2
portc=%00000100:portb=%10010001:PAUSE 2
portc=%00001000:portb=%10010001:PAUSE 2
portc=%00010000:portb=%11110001:PAUSE 2
portc=%00000000
NEXT

GOTO inicio2


numero6:

FOR x=1 TO 40                    ; subrutina genera cero - seis
portd=%00000001:portb=%11111111:PAUSE 2
portd=%00000010:portb=%10000001:PAUSE 2
portd=%00000100:portb=%10000001:PAUSE 2
portd=%00001000:portb=%10000001:PAUSE 2
portd=%00010000:portb=%11111111:PAUSE 2
portd=%00000000
portc=%00000001:portb=%00011111:PAUSE 2
portc=%00000010:portb=%00010001:PAUSE 2
portc=%00000100:portb=%00010001:PAUSE 2
portc=%00001000:portb=%00000001:PAUSE 2
portc=%00010000:portb=%11111111:PAUSE 2
portc=%00000000
NEXT

GOTO inicio2
```

```
numero7:

FOR x=1 TO 40                ; subrutina genera cero - siete
portd=%00000001:portb=%11111111:PAUSE 2
portd=%00000010:portb=%10000001:PAUSE 2
portd=%00000100:portb=%10000001:PAUSE 2
portd=%00001000:portb=%10000001:PAUSE 2
portd=%00010000:portb=%11111111:PAUSE 2
portd=%00000000
portc=%00000001:portb=%11111111:PAUSE 2
portc=%00000010:portb=%10010000:PAUSE 2
portc=%00000100:portb=%10010000:PAUSE 2
portc=%00001000:portb=%10000000:PAUSE 2
portc=%00010000:portb=%10000000:PAUSE 2
portc=%00000000
NEXT

GOTO inicio2


numero8:

FOR x=1 TO 40                ; subrutina genera cero - ocho
portd=%00000001:portb=%11111111:PAUSE 2
portd=%00000010:portb=%10000001:PAUSE 2
portd=%00000100:portb=%10000001:PAUSE 2
portd=%00001000:portb=%10000001:PAUSE 2
portd=%00010000:portb=%11111111:PAUSE 2
portd=%00000000:
portc=%00000001:portb=%11111111:PAUSE 2
```

```
portc=%00000010:portb=%10010001:PAUSE 2
portc=%00000100:portb=%10010001:PAUSE 2
portc=%00001000:portb=%10010001:PAUSE 2
portc=%00010000:portb=%11111111:PAUSE 2
portc=%00000000
NEXT

GOTO inicio2


numero9:

FOR x=1 TO 40                ; subrutina genera cero - nueve
portd=%00000001:portb=%11111111:PAUSE 2
portd=%00000010:portb=%10000001:PAUSE 2
portd=%00000100:portb=%10000001:PAUSE 2
portd=%00001000:portb=%10000001:PAUSE 2
portd=%00010000:portb=%11111111:PAUSE 2
portd=%00000000:
portc=%00000001:portb=%11111111:PAUSE 2
portc=%00000010:portb=%10010001:PAUSE 2
portc=%00000100:portb=%10010001:PAUSE 2
portc=%00001000:portb=%10010001:PAUSE 2
portc=%00010000:portb=%11110000:PAUSE 2
portc=%00000000
NEXT

GOTO inicio2


FOR x=1 TO 40                ; subrutina genera uno - cero
```

```
portd=%00000001:portb=%11111111:PAUSE 2
portd=%00000010:portb=%01000001:PAUSE 2
portd=%00000100:portb=%00100000:PAUSE 2
portd=%00001000:portb=%00010000:PAUSE 2
portd=%00010000:portb=%00001000:PAUSE 2
portd=%00000000:
portc=%00000001:portb=%11111111:PAUSE 2
portc=%00000010:portb=%10000001:PAUSE 2
portc=%00000100:portb=%10000001:PAUSE 2
portc=%00001000:portb=%10000001:PAUSE 2
portc=%00010000:portb=%11111111:PAUSE 2
portc=%00000000
NEXT

GOTO inicio2


; ***** Programa para el RTC  ***************************
RELOJ:

DEFINE I2C_SCLOUT 1          ; para que no necesite resistencia pull up en SCL
CPIN VAR Portc.7             ; pin señal de reloj I2C
DPIN VAR PORTC.6             ; pin de datos I2C

segu VAR BYTE                ; variable segu de 1 a 255
minu VAR BYTE                ; variable para minutos
hora VAR BYTE                ; variable para las horas
diaS VAR BYTE                ; variable día de la semana
diaF VAR BYTE                ; variable día fecha del mes
mes VAR BYTE                 ; variable mes
anio VAR BYTE                ; variable año de dos dígitos
```

```
dato VAR BYTE                ; variable para almacenar dato leído
actualizado VAR bit          ; variable para almacenar 1 ó 0

eeprom 0,[0]

read 0,actualizado           ; carga el valor de la memoria EEPROM dirección 0

IF actualizado=0 THEN grabarRTC  ; si es la primera vez que corre ir a grabar RTC
                                 ; caso contrario solo leer el RTC
inicio3:
   I2CREAD DPIN,CPIN,%11010000,0,[segu]      ; leer los datos de mem.0
   I2CREAD DPIN,CPIN,%11010000,1,[minu]      ; 1,2,...y guardarlos en sus
   I2CREAD DPIN,CPIN,%11010000,2,[hora]      ; respectivas variables
   I2CREAD DPIN,CPIN,%11010000,3,[diaS]
   I2CREAD DPIN,CPIN,%11010000,4,[diaF]
   I2CREAD DPIN,CPIN,%11010000,5,[mes]
   I2CREAD DPIN,CPIN,%11010000,6,[anio]

LCDOUT $FE,1,hex2 hora,":",hex2 minu,":",hex2 segu    ; mostrar la hora
                                                      ; min y segs. en 2 dígitos (HEX2)
LCDOUT $fe,$c0                                        ; saltar a la segunda línea del LCD
   IF diaS=$1 THEN LCDOUT "Dom."                      ; mostrar día de la semana
   IF diaS=$2 THEN LCDOUT "Lun."
   IF diaS=$3 THEN LCDOUT "Mar."
   IF diaS=$4 THEN LCDOUT "Mie."
   IF diaS=$5 THEN LCDOUT "Jue."
   IF diaS=$6 THEN LCDOUT "Vie."
   IF diaS=$7 THEN LCDOUT "Sab."

LCDOUT $fe,$c5,hex2 diaF,"/"                          ; mostrar el día del mes/
```

```
LCDOUT $fe,$cB,"/20",hex2 anio          ; mostrar año /20+05

LCDOUT $fe,$c8                          ; pasar a la casilla 8
   IF mes=$1 THEN LCDOUT "ene"          ; mostrar el mes
   IF mes=$2 THEN LCDOUT "feb"
   IF mes=$3 THEN LCDOUT "mar"
   IF mes=$4 THEN LCDOUT "abr"
   IF mes=$5 THEN LCDOUT "may"
   IF mes=$6 THEN LCDOUT "jun"
   IF mes=$7 THEN LCDOUT "jul"
   IF mes=$8 THEN LCDOUT "ago"
   IF mes=$9 THEN LCDOUT "sep"
   IF mes=$10 THEN LCDOUT "oct"
   IF mes=$11 THEN LCDOUT "nov"
   IF mes=$12 THEN LCDOUT "dic"
PAUSE 500

GOTO inicio3

; **********subrutina de Grabación. **********

grabarRTC:

I2CWRITE DPIN,CPIN,%11010000,0,[$00]    ; setear 00 segundos
   PAUSE 10                             ; retardo para finalizar grabación
I2CWRITE DPIN,CPIN,%11010000,1,[$45]    ; setear 45 minutos
   PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,2,[$16]    ; setear las 16 horas
   PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,3,[$2]     ; setear dia luens,D=1,L=2, M=3,M=4
   PAUSE 10                             ; J=5,V=6,S=7
```

```
I2CWRITE DPIN,CPIN,%11010000,4,[$18]    ; setear dia  18 del mes
  PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,5,[$10]    ; setear mes octubre
  PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,6,[$05]    ; setear año 05
  PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,7,[$10]    ; control %00010000 para encender
  PAUSE 10                              ; el led cada 1 seg.

WRITE 0,1                   ; escribir en la memoria 0 el valor de 1 para que no
                            ; se vuelva a grabar otra vez estos datos en el RTC
GOTO inicio3                ; ir a presentar los datos en el LCD
```

ANEXO 2 .-HOJA DE DATOS DE ELEMENTOS IMPORTANTES

MICROCHIP

PIC16F84A

18-pin *Enhanced* FLASH/EEPROM 8-Bit Microcontroller

High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single-cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of Data RAM
- 64 bytes of Data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 Special Function Hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt-on-change
  - Data EEPROM write complete

Peripheral Features:

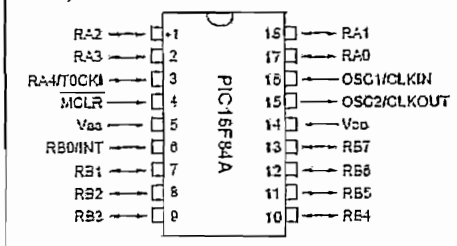
- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Special Microcontroller Features:

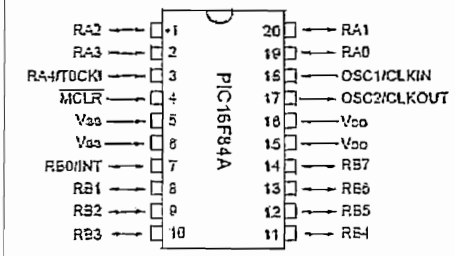
- 10,000 erase/write cycles *Enhanced* FLASH Program memory typical
- 10,000,000 typical erase/write cycles EEPROM Data memory typical
- EEPROM Data Retention > 40 years
- In-Circuit Serial Programming™ (ICSP™) - via two pins
- Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Code protection
- Power saving SLEEP mode
- Selectable oscillator options

Pin Diagrams

PDIP, SOIC



SSOP



CMOS Enhanced FLASH/EEPROM Technology:

- Low power, high speed technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 5.5V
  - Industrial: 2.0V to 5.5V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 15 µA typical @ 2V, 32 kHz
  - < 0.5 µA typical standby current @ 2V

PIC16F84A

1.0 DEVICE OVERVIEW

This document contains device specific information for the operation of the PIC16F84A device. Additional information may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023), which may be downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

The PIC16F84A belongs to the mid-range family of the PICmicro® microcontroller devices. A block diagram of the device is shown in Figure 1-1.

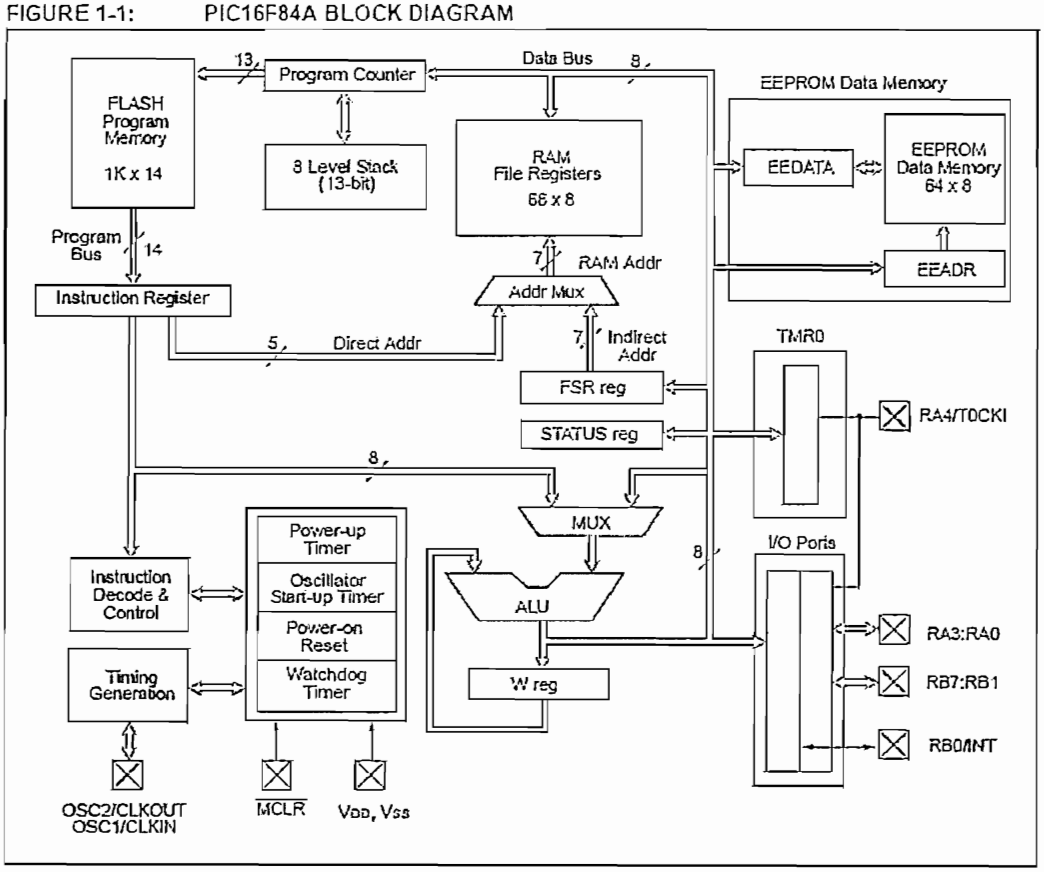
The program memory contains 1K words, which translates to 1024 instructions, since each 14-bit program memory word is the same width as each device instruction. The data memory (RAM) contains 68 bytes. Data EEPROM is 64 bytes.

There are also 13 I/O pins that are user-configured on a pin-to-pin basis. Some pins are multiplexed with other device functions. These functions include:

- External interrupt
- Change on PORTB interrupt
- Timer0 clock input

Table 1-1 details the pinout of the device with descriptions and details for each pin.

FIGURE 1-1: PIC16F84A BLOCK DIAGRAM



PIC16F84A

TABLE 1-1: PIC16F84A PINOUT DESCRIPTION

Pin Name	PDIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	18	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKOUT, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
$\overline{MCLR}$	4	4	4	I/P	ST	Master Clear (Reset) input/programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0	17	17	19	I/O	TTL	
RA1	18	18	20	I/O	TTL	
RA2	1	1	1	I/O	TTL	
RA3	2	2	2	I/O	TTL	
RA4/T0CKI	3	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	7	I/O	TTL/ST(1)	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	8	I/O	TTL	
RB2	8	8	9	I/O	TTL	
RB3	9	9	10	I/O	TTL	
RB4	10	10	11	I/O	TTL	Interrupt-on-change pin.
RB5	11	11	12	I/O	TTL	Interrupt-on-change pin.
RB6	12	12	13	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming clock.
RB7	13	13	14	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming data.
Vss	5	5	5,6	P	—	Ground reference for logic and I/O pins.
VDD	14	14	15,16	P	—	Positive supply for logic and I/O pins.

Legend: I= input O = Output I/O = Input/Output P = Power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

2.3 Special Function Registers

The Special Function Registers (Figure 2-2 and Table 2-1) are used by the CPU and Peripheral functions to control the device operation. These registers are static RAM.

The special function registers can be classified into two sets, core and peripheral. Those associated with the core functions are described in this section. Those related to the operation of the peripheral features are described in the section for that specific feature.

TABLE 2-1: SPECIAL FUNCTION REGISTER FILE SUMMARY

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on RESET	Details on page
Bank 0											
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								---- ----	11
01h	TMR0	8-bit Real-Time Clock/Counter								xxxx xxxx	20
02h	PCL	Low Order 8 bits of the Program Counter (PC)								0000 0000	11
03h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx	11
05h	PORTA(4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	16
06h	PORTB(5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	18
07h	—	Unimplemented location, read as '0'								—	—
08h	EEDATA	EEPROM Data Register								xxxx xxxx	13,14
09h	EEADR	EEPROM Address Register								xxxx xxxx	13,14
0Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10
Bank 1											
80h	INDF	Uses Contents of FSR to address Data Memory (not a physical register)								---- ----	11
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	9
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	11
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	11
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	16
86h	TRISB	PORTB Data Direction Register								1111 1111	18
87h	—	Unimplemented location, read as '0'								—	—
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	13
89h	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	14
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends on condition

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> are never transferred to PCLATH.

2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ Reset.

3: Other (non power-up) RESETS include: external RESET through $\overline{MCLR}$ and the Watchdog Timer Reset.

4: On any device RESET, these pins are configured as inputs.

5: This is the value that will be in the port output latch.

PIC16F84A

2.3.1 STATUS REGISTER

The STATUS register contains the arithmetic status of the ALU, the RESET status and the bank select bit for data memory.

As with any register, the STATUS register can be the destination for any instruction. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper three bits and set the Z bit. This leaves the STATUS register as `000u uluu` (where `u` = unchanged).

Only the `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions should be used to alter the STATUS register (Table 7-2), because these instructions do not affect any status bit.

Note 1: The IRP and RP1 bits (STATUS<7:6>) are not used by the PIC16F84A and should be programmed as cleared. Use of these bits as general purpose R/W bits is NOT recommended, since this may affect upward compatibility with future products.

2: The C and DC bits operate as a $\overline{borrow}$ and digit $\overline{borrow}$ out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

3: When the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. The specified bit(s) will be updated according to device logic

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7-6 Unimplemented: Maintain as '0'

bit 5 RP0: Register Bank Select bits (used for direct addressing)
`01` = Bank 1 (80h - FFh)
`00` = Bank 0 (00h - 7Fh)

bit 4 $\overline{TO}$: Time-out bit
`1` = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
`0` = A WDT time-out occurred

bit 3 $\overline{PD}$: Power-down bit
`1` = After power-up or by the `CLRWDT` instruction
`0` = By execution of the `SLEEP` instruction

bit 2 Z: Zero bit
`1` = The result of an arithmetic or logic operation is zero
`0` = The result of an arithmetic or logic operation is not zero

bit 1 DC: Digit carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions) (for $\overline{borrow}$, the polarity is reversed)
`1` = A carry-out from the 4th low order bit of the result occurred
`0` = No carry-out from the 4th low order bit of the result

bit 0 C: Carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions) (for $\overline{borrow}$, the polarity is reversed)
`1` = A carry-out from the Most Significant bit of the result occurred
`0` = No carry-out from the Most Significant bit of the result occurred

Note: A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

MICROCHIP

PIC16F87XA

28/40-Pin Enhanced FLASH Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High Performance RISC CPU:

- Only 35 single word instructions to learn
- All single cycle instructions except for program branches, which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory, Up to 368 x 8 bytes of Data Memory (RAM), Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8 channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (VREF) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible

Special Microcontroller Features:

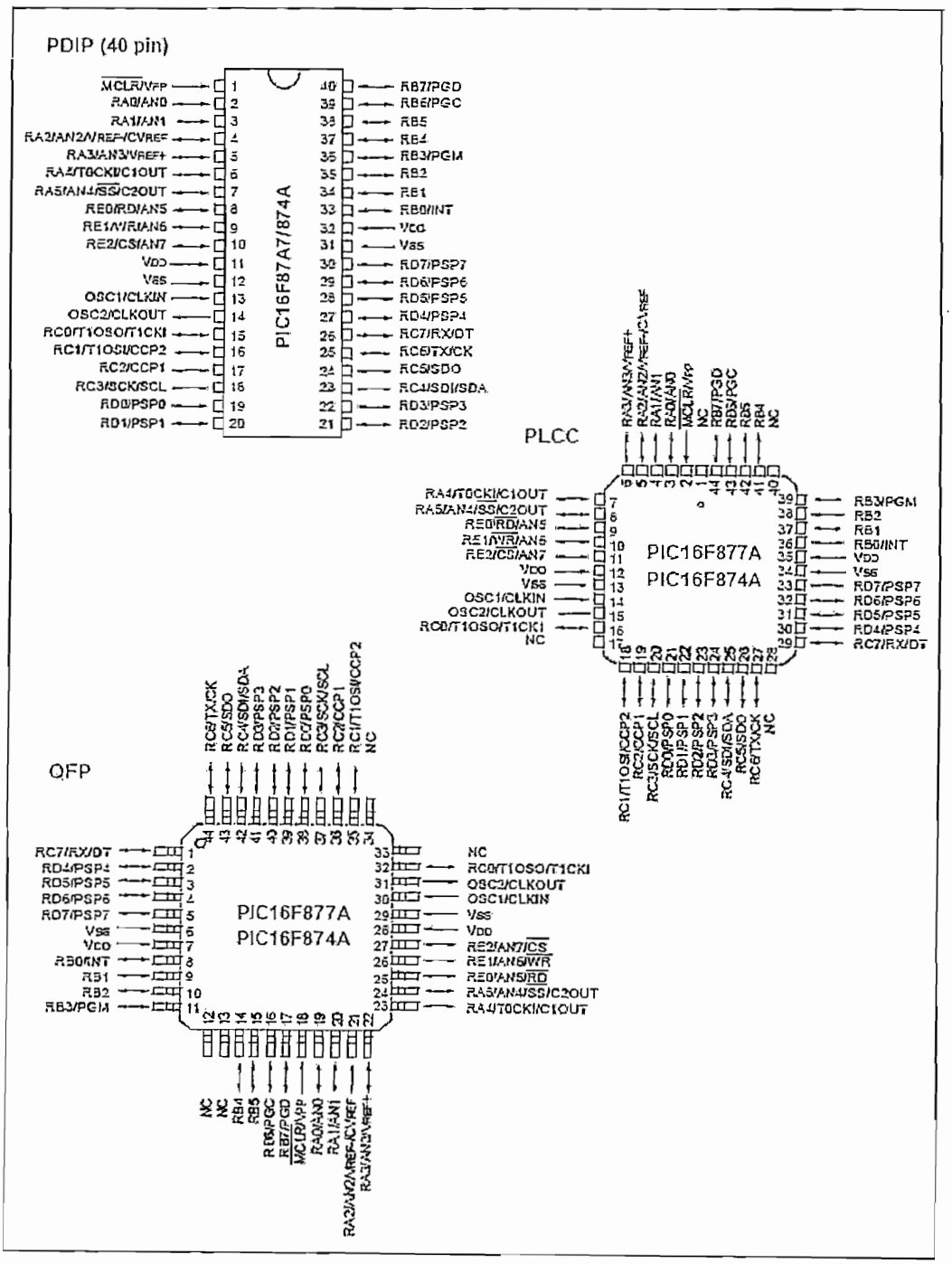
- 100,000 erase/write cycle Enhanced FLASH program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

CMOS Technology:

- Low power, high speed FLASH/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low power consumption

Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^2C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

Pin Diagram


PDIP (40 pin)
PIC16F87A7/874A
PLCC
PIC16F877A
PIC16F874A
QFP
PIC16F877A
PIC16F874A

PIC16F87XA

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKI	13	14	30	I	ST/CMOS(4)	Oscillator crystal or external clock input.
OSC1						Oscillator crystal input or external clock source input. ST buffer when configured in RC mode. Otherwise CMOS.
CLKI						External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins).
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal or clock output.
OSC2						Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO						In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
MCLR/VPP	1	2	18	I/P	ST	Master Clear (input) or programming voltage (output).
MCLR						Master Clear (Reset) input. This pin is an active low RESET to the device.
VPP						Programming voltage input.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19		TTL	
RA0				I/O		Digital I/O.
AN0				I		Analog input 0.
RA1/AN1	3	4	20		TTL	
RA1				I/O		Digital I/O.
AN1				I		Analog input 1.
RA2/AN2/VREF-/CVREF	4	5	21		TTL	
RA2				I/O		Digital I/O.
AN2				I		Analog input 2.
VREF-				I		A/D reference voltage (Low) input.
CVREF				O		Comparator VREF output.
RA3/AN3/VREF+	5	6	22		TTL	
RA3				I/O		Digital I/O.
AN3				I		Analog input 3.
VREF+				I		A/D reference voltage (High) input.
RA4/T0CKI/C1OUT	6	7	23		ST	
RA4				I/O		Digital I/O — Open drain when configured as output.
T0CKI				I		Timer0 external clock input.
C1OUT				O		Comparator 1 output.
RA5/SS/AN4/C2OUT	7	8	24		TTL	
RA5				I/O		Digital I/O.
SS				I		SPI slave select input.
AN4				I		Analog input 4.
C2OUT				O		Comparator 2 output.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8		TTL/ST(1)	
RB0				I/O		Digital I/O.
INT				I		External interrupt.
RB1	34	37	9	I/O	TTL	Digital I/O.
RB2	35	38	10	I/O	TTL	Digital I/O.
RB3/PGM	36	39	11		TTL	
RB3				I/O		Digital I/O.
PGM				I/O		Low voltage ICSP programming enable pin.
RB4	37	41	14	I/O	TTL	Digital I/O.
RB5	38	42	15	I/O	TTL	Digital I/O.
RB6/PGC	39	43	16		TTL/ST(2)	
RB6				I/O		Digital I/O.
PGC				I/O		In-Circuit Debugger and ICSP programming clock.
RB7/PGD	40	44	17		TTL/ST(2)	
RB7				I/O		Digital I/O.
PGD				I/O		In-Circuit Debugger and ICSP programming data.
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32		ST	
RC0				I/O		Digital I/O.
T1OSO				O		Timer1 oscillator output.
T1CKI				I		Timer1 external clock input.
RC1/T1OSI/CCP2	16	18	35		ST	
RC1				I/O		Digital I/O.
T1OSI				I		Timer1 oscillator input.
CCP2				I/O		Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1	17	19	36		ST	
RC2				I/O		Digital I/O.
CCP1				I/O		Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37		ST	
RC3				I/O		Digital I/O.
SCK				I/O		Synchronous serial clock input/output for SPI mode.
SCL				I/O		Synchronous serial clock input/output for I^2C mode.
RC4/SDI/SDA	23	25	42		ST	
RC4				I/O		Digital I/O.
SDI				I		SPI data in.
SDA				I/O		I^2C data I/O.
RC5/SDO	24	26	43		ST	
RC5				I/O		Digital I/O.
SDO				O		SPI data out.
RC6/TX/CK	25	27	44		ST	
RC6				I/O		Digital I/O.
TX				O		USART asynchronous transmit.
CK				I/O		USART 1 synchronous clock.
RC7/RX/DT	26	29	1		ST	
RC7				I/O		Digital I/O.
RX				I		USART asynchronous receive.
DT				I/O		USART synchronous data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

PIC16F87XA

TABLE 1-3: PIC16F874A/877A PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38		ST/TTL(3)	
RD0				I/O		Digital I/O.
PSP0				I/O		Parallel Slave Port data.
RD1/PSP1	20	22	39		ST/TTL(3)	
RD1				I/O		Digital I/O.
PSP1				I/O		Parallel Slave Port data.
RD2/PSP2	21	23	40		ST/TTL(3)	
RD2				I/O		Digital I/O.
PSP2				I/O		Parallel Slave Port data.
RD3/PSP3	22	24	41		ST/TTL(3)	
RD3				I/O		Digital I/O.
PSP3				I/O		Parallel Slave Port data.
RD4/PSP4	27	30	2		ST/TTL(3)	
RD4				I/O		Digital I/O.
PSP4				I/O		Parallel Slave Port data.
RD5/PSP5	28	31	3		ST/TTL(3)	
RD5				I/O		Digital I/O.
PSP5				I/O		Parallel Slave Port data.
RD6/PSP6	29	32	4		ST/TTL(3)	
RD6				I/O		Digital I/O.
PSP6				I/O		Parallel Slave Port data.
RD7/PSP7	30	33	5		ST/TTL(3)	
RD7				I/O		Digital I/O.
PSP7				I/O		Parallel Slave Port data.
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25		ST/TTL(3)	
RE0				I/O		Digital I/O.
$\overline{RD}$				I		Read control for parallel slave port.
AN5				I		Analog input 5.
RE1/$\overline{WR}$/AN6	9	10	26		ST/TTL(3)	
RE1				I/O		Digital I/O.
$\overline{WR}$				I		Write control for parallel slave port.
AN6				I		Analog input 6.
RE2/$\overline{CS}$/AN7	10	11	27		ST/TTL(3)	
RE2				I/O		Digital I/O.
$\overline{CS}$				I		Chip select control for parallel slave port.
AN7				I		Analog input 7.
VSS	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17, 28,40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

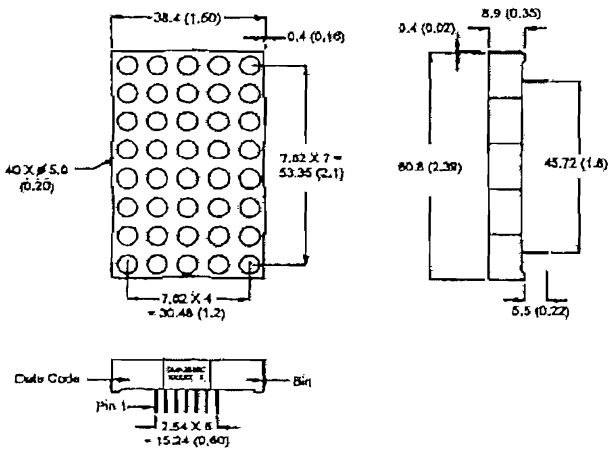
Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

FAIRCHILD
SEMICONDUCTOR™

2.3 INCH (58.4 mm) 5 X 8 DOT MATRIX STICK DISPLAY

AlGaAs Red GMA2285C
AlGaAs Red GMC2285C

PACKAGE DIMENSIONS



NOTE: Dimensions are in mm (inch).
Tolerances are ± 0.25 (0.1) unless otherwise noted.
All pins are 0.5 (.02).

DESCRIPTION

The GMX2285C 5 X 8, Single Hetero Junction AlGaAs Red dot matrix display. It has a grey face with neutral segment color.

FEATURES

2.5" (58.4mm) character height.
Low power requirement.
Wide 130° viewing angle.
High brightness and contrast
5 X 8 array with X-Y select.
X-Y stackable.
Easy mounting on P.C. board.

MODEL NUMBER

Part Number	Colour	Description
GMA2285C	AlGaAs Red	Common anode row.
GMC2285C	AlGaAs Red	Common Cathode row.

(For other color options, contact your local area Sales Office)

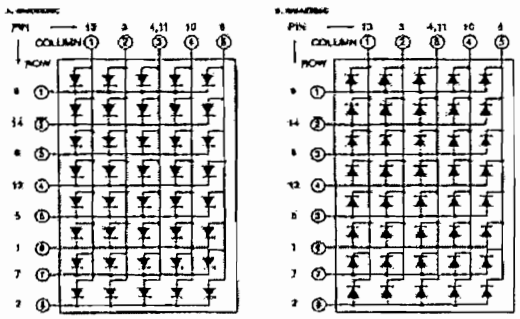
FAIRCHILD
SEMICONDUCTOR™

2.3 INCH (58.4 mm) 5 X 8 DOT MATRIX STICK DISPLAY

PIN CONNECTION:

GMA2285C		GMC2285C	
Pin Number	Function	Pin Number	Function
1	Anode Row 6	1	Cathode Row 6
2	Anode Row 8	2	Cathode Row 8
3	Cathode Column 2	3	Anode Column 2
4	Cathode Column 3	4	Anode Column 3
5	Anode Row 5	5	Cathode Row 5
6	Cathode Column 5	6	Anode Column 5
7	Anode Row 7	7	Cathode Row 7
8	Anode Row 3	8	Cathode Row 3
9	Anode Row 1	9	Cathode Row 1
10	Cathode Column 4	10	Anode Column 4
11	Cathode Column 3	11	Anode Column 3
12	Anode Row 4	12	Cathode Row 4
13	Cathode Column 1	13	Anode Column 1
14	Anode Row 2	14	Cathode Row 2

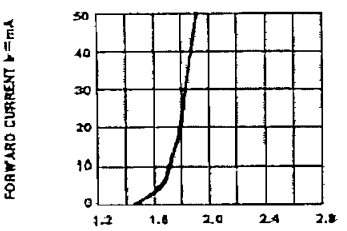
SCHEMATIC:



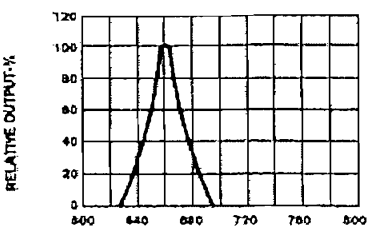
FAIRCHILD
SEMICONDUCTOR™

2.3 INCH (58.4 mm) 5 X
DOT MATRIX STICK DISPLA

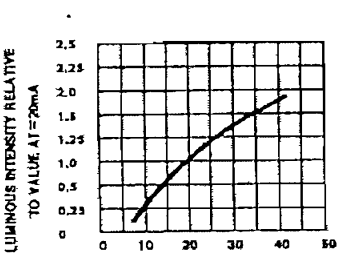
GRAPHICAL DETAIL: AlGaAs Red (T_A = 25°C unless otherwise specified)



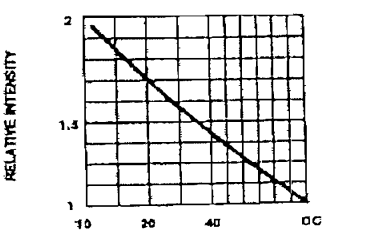
FORWARD VOLTAGE (V_F)-VOLTS
Fig.1 FORWARD CURRENT VS. FORWARD VOLTAGE.



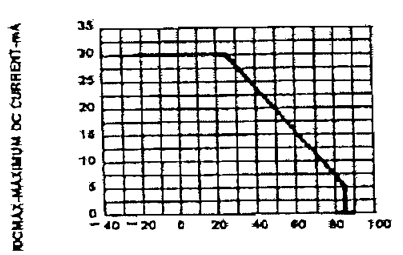
WAVELENGTH (λ)-nm
Fig.2 SPECTRAL RESPONSE



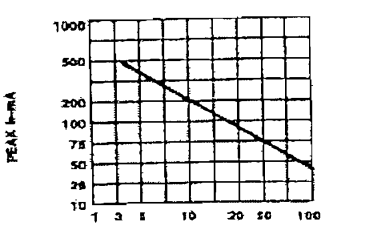
I_F-FORWARD CURRENT-mA
Fig.3 RELATIVE LUMINOUS INTENSITY VS. FORWARD CURRENT



DUTY CYCLE % PER SEGMENT
(AVERAGE I_F=10mA)
Fig.5 LUMINOUS INTENSITY VS. DUTY CYCLE



T_A AMBIENT TEMPERATURE °C
Fig.4 MAXIMUM ALLOWABLE DC CURRENT PER SEGMENT VS. A FUNCTION OF AMBIENT TEMPERATURE.



DUTY CYCLE %
Fig. 6 MAX PEAK CURRENT VS. DUTY CYCLE %
(REFRESH RATE f=1 KHz)

ANEXO 3.- MANUAL DE LOOPS MAGNETICOS

625X

Inductive Loop Detector

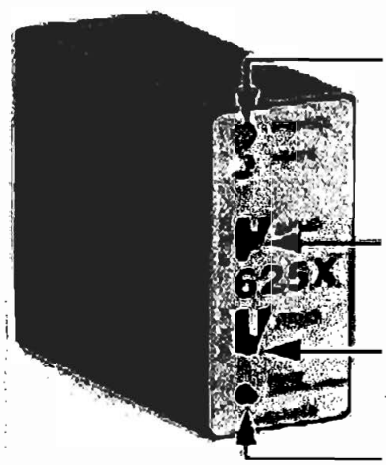
Installation Guide

PEEK TRAFFIC

Sarasota Series Detectors

Quick Setup

1. Verify the input voltage by looking at the label on the back of the detector (located just under the connector.)
2. Terminate the wiring harness to the proper connector on the terminal strip for each of the desired functions (power, desired outputs, loop, etc.)
3. Connect the harness to the detector.
4. Verify power is reaching the detector. (The Presence LED should come on and flash for two seconds, then go out.)
5. Set sensitivity switch (SENS) to Medium (M).
6. Adjust the frequency (FREQ) to Low (L) for a single loop. For multiple loop applications, set each detector to a different frequency setting.
7. Press and hold the RESET button for three seconds, until the Presence LED flashes. The detector is ready to operate.



Upon power-up, Presence LED will flash for 2 seconds, then go out.

Sensitivity: First set to Medium, then adjust if necessary.

Frequency: Set to Low or Medium for single loop installations.

To Reset Detector: Press and hold for 3 seconds, or until Presence LED blinks.

Setup Details

1. Power is supplied to the unit on pins 1 & 2 (black & white.)

2. Ground the unit by taking pin 4 (green) to the Operator ground.

3. The most common hook-up is for using the Presence relay, pins 5 & 6 (yellow & blue.) The Presence relay common, pin 5 (yellow), is taken to the common of the terminal strip on the Operator. The Presence relay N.O. Normally Open, pin 6 (blue) is taken to either Open, Close, Hold Open, etc. (i.e. whichever function is to be performed when presence is detected.)

4. For a second Presence or Pulse output, Relay 2 on pins 3 & 9 is used. Relay 2 Normally Open (N.O.) is on pin 3 (orange) and is taken to either Open, Close, Hold Open, etc. (whichever function is to be performed.) If a Normally Closed (N.C.) output is necessary, use pin 11 (white/red). The Relay 2 common, pin 9 (red) is taken to the common on the terminal strip of the Operator.

5. The loop leads come into the detector on pins 7 & 8 (gray & brown.) This is from the loop to the ground. These leads must be twisted all the way to the detector for proper operation.
(See diagram on page 7.)

Setup Details

1. Power is supplied to the unit on pins 1 & 2 (black & white.)

2. Ground the unit by taking pin 4 (green) to the Operator ground.

3. The most common hook-up is for using the Presence relay, pins 5 & 6 (yellow & blue.) The Presence relay common, pin 5 (yellow), is taken to the common of the terminal strip on the Operator. The Presence relay N.O. Normally Open, pin 6 (blue) is taken to either Open, Close, Hold Open, etc. (i.e. whichever function is to be performed when presence is detected.)

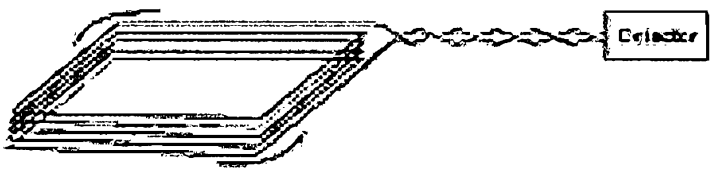
4. For a second Presence or Pulse output, Relay 2 on pins 3 & 9 is used. Relay 2 Normally Open (N.O.) is on pin 3 (orange) and is taken to either Open, Close, Hold Open, etc. (whichever function is to be performed.) If a Normally Closed (N.C.) output is necessary, use pin 11 (white/red). The Relay 2 common, pin 9 (red) is taken to the common on the terminal strip of the Operator.

5. The loop leads come into the detector on pins 7 & 8 (gray & brown.) This is from the loop to the ground. These leads must be twisted all the way to the detector for proper operation.
(See diagram on page 7.)

Testing the Loop

A good loop is critical for reliable operation from your detector. When installing your loop, take great care not to damage the insulation of the wire. Breaks in the insulation can cause the wire to act as a wick — pulling in moisture, corroding the wire itself, and causing erratic operation from the detector. Cross-link polyethylene is the most popular insulation and is strongly recommended (XHHW), in 16 or 18 gauge. If the lead-in is extremely long, increase the wire size. The insulation must be able to withstand wear and abrasion from the shifting of pavement, moisture, and attacks by solvents and oils, as well as able to withstand high-temperature sealants. Stranded wire is recommended over solid wire, because of its mechanical characteristics. Stranded wire is more likely to survive bending and stretching than solid wire.

Megging a loop and lead-in should have an insulation resistance to earth greater than 20MΩ, measured at 500 Volts. One end of the loop goes to one lead from the meggar, and the other lead from the meggar goes to a good ground. The loop should also display a series resistance of less than 1Ω with a standard Ohm meter. If a problem with a loop is suspected, try swapping the detector with a known good detector and see if the problem follows the detector or the loop.



Output Configuration Options

Presence means the relay will be energized the entire time a metal mass is within the field generated by the loop. The 625X has the ability to provide two presence outputs. Relay 2 can be set for presence or pulse.

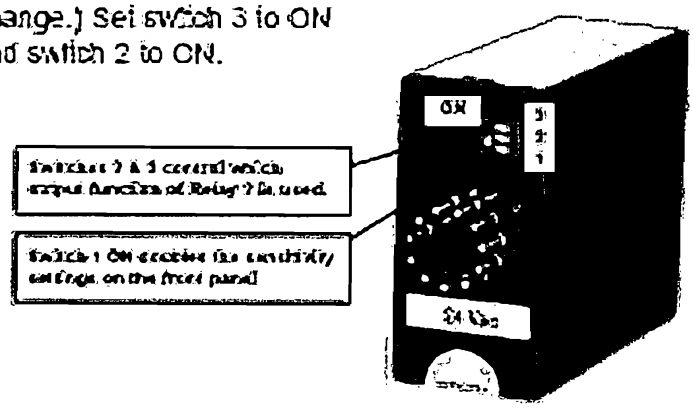
Relay 1 is always presence (pins 5 & 6, yellow and blue wires), however Relay 2 on the 625X can be assigned to one of several output configurations.

Relay 2 (pins 3 & 9, Orange and Red wires) can be set for presence using the switches on the back panel. Set switch 3 to OFF and switch 2 to ON

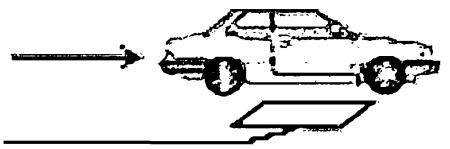
Pulse on Entry means that the relay 2 will be energized as soon as a metal mass enters the field generated by the loop. On the back panel, set switch 3 to OFF and switch 2 to OFF. This pulse will last 125mS.

Pulse on Exit means that relay 2 will be energized when a metal mass has left the loop. Set switch 3 to ON and switch 2 to OFF. The pulse will last 125mS.

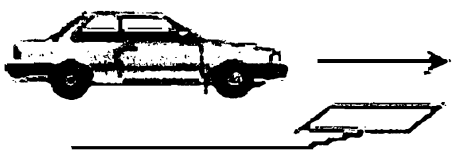
Loop fault output means the relay will be energized if there is a current fault (open loop, shorted loop or greater than a 25% inductance change.) Set switch 3 to ON and switch 2 to ON.



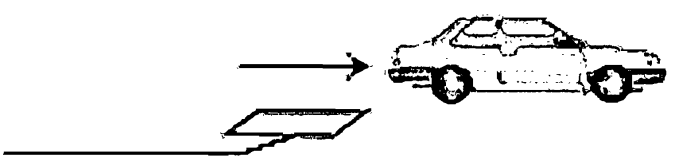
10



Presence—Constant presence (detect) while metal is over the loop



Pulse on Entry—Momentary pulse when metal enters the loop



Pulse on Exit—Momentary pulse when metal leaves the loop