UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIA Y TECNOLOGÍA

ESCUELA DE CIENCIAS INDUSTRIALES

DISEÑO Y CONSTRUCCIÓN DE UNA ALARMA CODIFICADA PARA VEHÍCULO.

Trabajo de Graduación previo a la obtención del Título de:
Tecnólogo Industrial en la especialidad de Electrónica.

Autor: Freddy A. Baculima Suárez
Director: Ing. Oswaldo López

CUENCA - ECUADOR

2006

Índice de Contenidos

Índice de anexos.

RESUMEN:

El presente trabajo tiene como objetivos los de diseñar y construir un sistema de alarma y antirrobo para vehículo, esto lo conseguimos con la recopilación de información necesaria, para así lograr el diseño de un circuito apropiado y versátil, que se ajuste a las necesidades de nuestro medio.

La base de este proyecto es una Unidad Microcontrolada con PIC 16F871 que es un pequeño computador diseñado específicamente para dar lectura a los diferentes periféricos de entrada y de acuerdo a su estado o manipulación hacer reaccionar a los periféricos de salida, esto se logro gracias a un programa (*software*) contenido en el PIC16F871.

ABSTRACT:

The aim of the present work is to design and to build a car alarm system, intended to avoid robberies. The description of a specific car alarm system is performed, aided with different information sources, as handbooks, specialized textbooks and internet web pages. The circuit design developed is suitable to most of the cars models available in our region.

This project is based in a microcontroller unit, with a PIC 16F871 system. This device is a computer which receives information from peripherals, such as: keyboard, main switch, on-off switch, etc. Based on the information received from the peripherals, the system activates the alarm outputs: siren, relays, leds. This information is processed by the PIC 16F871 device.

INTRODUCCIÓN.

Puesto que en la actualidad el campo de la electrónica cubre prácticamente todas las áreas en las que nos desenvolvemos, con la presente monografía se pretende abarcar una parte del campo de la electrónica en el automóvil específicamente en lo que hace referencia a la seguridad (alarmas y sistemas antirrobo), ya que la electrónica es ya una parte importante del conjunto de mecanismos que rigen el funcionamiento de un vehículo. Esta tecnología, que nos permitió el avance de los semiconductores se ha a desarrollado hasta un grado de altísima sofisticación, ha superado sus primitivas fronteras de aplicación al mundo de la radiodifusión y la TV para pasar a invadir, hoy por hoy, todos los terrenos de la industria, de modo que se encuentra presente en infinidad de variantes que van desde el amplísimo campo de los ordenadores, hasta la nueva tecnología de la robótica pasando, desde luego, por una enorme variedad de circuitos de control y de mando con las más vastas posibilidades de aplicación.

El automóvil ha venido ha favorecerse del imparable desarrollo de la electrónica y ahora tenemos muchas variantes de circuitos fundamentales que están regidos por sistemas construidos por procedimientos electrónicos, tal como ocurre en los circuitos de rectificación y regulación de corriente producida por el alternador, en el circuito de encendido, en los dispositivos de control de luces testigo sustituidas por paneles electrónicos compuestos por leds (componentes de estado sólido); en los tacómetros y velocímetros electrónicos; en los limpiaparabrisas intermitentes, también de control electrónico; en los sistemas de inyección de gasolina y en los ordenadores de a bordo, indicadores de giro, y por supuesto alarmas y antirrobos, y un largo número de aplicaciones.

Si todos estos mecanismos intervienen ya en muchos vehículos está claro que se hace necesario ampliar nuestros conocimientos en este campo, es por ello que el presente trabajo tiene por objeto abarcar una parte de la electrónica del automóvil hoy en día muy común en toda clase de vehículos, como son las alarmas y sistemas antirrobos.

Baculima Suárez Freddy Adriano
Trabajo de Graduación
Ing. Oswaldo López
Julio 2006

DISEÑO Y CONSTRUCCIÓN DE UNA ALARMA CODIFICADA PARA VEHÍCULO.

CAPITULO 1: HARDWARE Y PROGRAMACIÓN DE MICROCONTROLADORES.

Introducción:

En este capitulo se tratara la parte teórica que hace referencia a los microcontroladores PIC en especial de la serie 16F87X, describiendo sus ventajas, características, conexión de sus diferentes circuitos complementarios, así como conexión de periféricos principales a los puertos, análisis de la arquitectura interna y por ultimo en lo que hace referencia al hardware también se explicará el funcionamiento y características de los diferentes sensores. En lo que se refiere al software describiremos la teoría necesaria para la Programación de los Microcontroladores Microchip.

1.1.- Microcontroladores vs. Microprocesadores.

En **1971** aparece en el mercado el primer **microprocesador (uP)** que supuso un cambio decisivo en las técnicas de instrumentación y control. Un microprocesador es un chip programable, que integra pocos recursos de hardware; básicamente los relacionados con el procesamiento de información (CPU) y con el trabajo aritmético (ALU). Para completar el desempeño de los microprocesadores aparecieron un conjunto de chips periféricos, tales como puertos de entrada salida, memoria, temporizadores; entre otros. Tales periféricos formaron parte de una familia de chips discretos con los que el **uP** debía comunicarse empleando básicamente tres tipos de buses: bus de datos, bus de direcciones y bus de control, estos uP tenían un tipo de arquitectura interna denominada Von-Neumann o arquitectura CISC (Complex Instruction Set Computer) o sea manejaban un set complejo de instrucciones. **(figura 1**A).

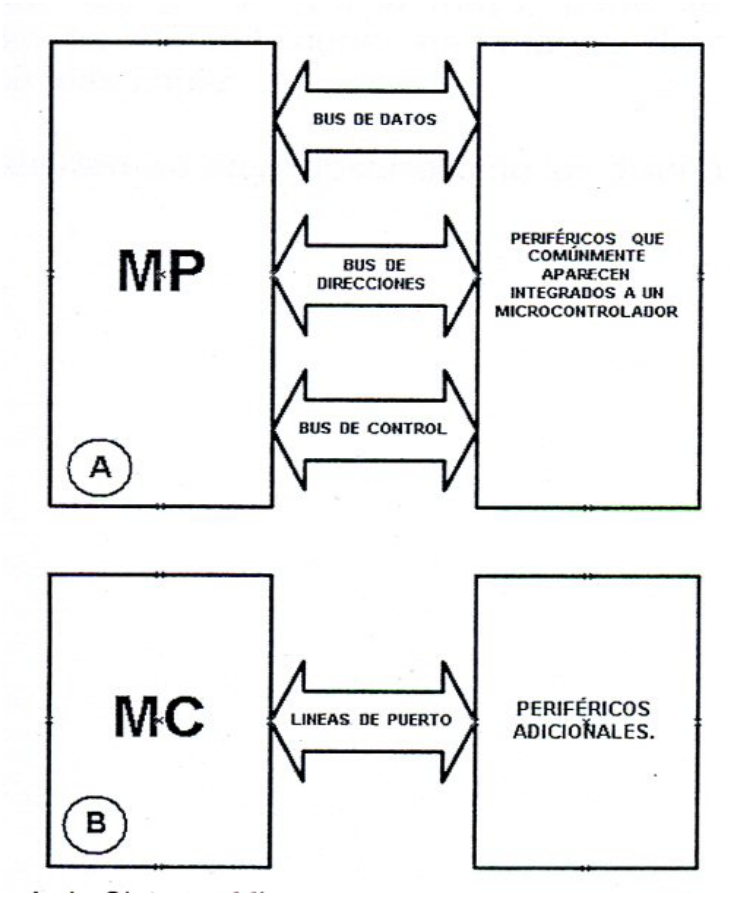


FIGURA 1. A: Sistema Microprocesador. B: Sistema Microcontrolador.

En el año **197**6, gracias al aumento de la capacidad de integración aparece el primer **microcontrolador (uC**), estos poseen un tipo de arquitectura interna denominada Harvard o arquitectura RISC (Reduced Instruction Set Computer) o sea manejan un set reducido de instrucciones. La diferencia fundamental de un **uC** con un **uP** es que el **Microcontrolador** integra la mayor cantidad de recursos en un solo chip y se comunica con el exterior solamente a través de líneas de entrada/salida o líneas de puerto **(figura 1**B). En la actualidad la solución de la mayoría de los proyectos electrónicos es pensada en primera instancia utilizando microcontroladores. Tal es el desarrollo alcanzado por las tecnologías de microcontroladores actuales que el papel de los u**P** ha sido relegado a la fabricación de PCs o a proyectos de gran escala. Algunas razones **que justifican la elección** de un **microcontrolador** son las siguientes:

- **Bajo costo**, puesto que integra muchos de los recursos que en u**P** aparecen de forma discreta y se miniaturizan los diseños, lo que supone abaratar costos de fabricación.
- **Fiabilidad.** Un uC integra la mayor parte de los recursos, por lo que se minimizan las interconexiones en la tarjeta de circuito impreso lográndose así un diseño más fiable.
- **Ahorro de tiempo** en el desarrollo de los diseños.

1.2.- Microcontroladores PIC.

En el mundo existen más de 50 fabricantes de Microcontroladores. Evidentemente, en estas condiciones es muy difícil elegir "el mejor". Realizando un análisis justo, "el mejor", en este tipo de tecnología no existe, porque cada aplicación generará necesidades específicas. Ahora bien Microchip, es la empresa que ha logrado reunir en sus MICROCONTROLADORES, la mayor cantidad de características ventajosas para satisfacer numerosas necesidades en diferentes tipos de aplicaciones. Algunas de las razones por las que los PICS han sido aceptados ampliamente, son las siguientes:

- Poseen una **relación costo-desempeño relativamente BA**JO, comparado con la de sus competidores.

- **Trabajan a velocidades grandes,** 40 Mhz, la última serie 18XXX y la que serie que utilizaremos en la elaboración de este trabajo (16F87X hasta 20 Mhz.). Hasta la gama media, poseen **Arquitectura RISC (Reduced Instruction Set Computer),** o sea manejan un **Set Reducido de Instruccione**s.

- **Amplia variedad de CHIPS,** Microchip ha optimizado la variedad de periféricos que incluye en sus MICROS, con el objetivo de que UD. encuentre el óptimo para su aplicación y no carezca o derroche recursos electrónicos en el CHIP. Así existen pequeños MC de 8 pines muy poderosos, existen series muy baratas y con gran desempeño, como la de los

PICS 16F870/871 y series ultra poderosas, que pueden inclusive emular el modo de trabajo de MICROPROCESADORES como la serie 18XXX.

- Otra gran ventaja de MICROCHIP, es la **migrabilidad del códi**go, entre diferentes tipos de CHIPS y familias, esto permite que un programa escrito para correr en un PIC12C508 de 8 pines sea fácilmente reelaborado para correr en un PIC16C57 que pertenece a la familia básica.

- **Tecnología fácil de estudiar.** Se puede estudiar Microchip de manera modular. ¿Qué significa esto?: para una mejor comprensión ilustremos con un ejemplo. Un PIC16F84, posee los módulos básicos: puerto A, puerto B, TMR0 y EEPROM de datos y es capaz de manejar 4 fuentes de interrupciones. Un PIC16F870 posee los mismos módulos y maneja también estas 4 fuentes de interrupción, pero adicionalmente posee dos timers, un módulo CCP, un puerto serie, y un conversor A/D de 10 BITS y maneja 7 fuentes de interrupciones adicionales. Si no se comprende aún la idea, esto quiere decir que si se asimiló el trabajo con los módulos incluidos en un F84, solo deberá estudiar en el 870, los módulos adicionales que aparecen... Ah! , dos detalles curiosos: ambos micros manejan el mismo SET de INSTRUCCIONES y un PIC16F870 es más barato que un F84.

- **Existe gran variedad de Herramientas de Desarrollo,** proporcionadas por Microchip o empresas alternativas. Gran cantidad de herramientas de SW para el desarrollo las proporciona Microchip de manera gratuita, además de una gran cantidad de Notas de Aplicación, que merecen una profunda consulta.

1.3.- Líneas de entrada/salida en los microcontroladores Microchip.

Como ya se había señalado, un microcontrolador se comunica con el entorno solamente a través de líneas de entrada / salida o puertos que vienen integrados al chip. En este sentido, para realizar la conexión de cualquier dispositivo periférico es imprescindible conocer las características relevantes de los puertos de la tecnología que se trabaje.

Las características ventajosas de las líneas de puerto en los PICS, también han determinado en gran medida la enorme tasa de acogida entre los diseñadores electrónicos. Dentro de las **características generales** más sobresalientes pudieran citarse las siguientes:

- **Pueden manejar hasta 25 mA** de corriente, tanto como fuente o sumidero, esto hace que sean capaces de manejar LEDs sin necesidad de Buffers.
- **Son configurables individualmente como salidas ó entrada**s, mediante registros denominados "TRIS". Existe un registro TRIS para cada puerto. Cada bit del cada registro está asociado al pin físico del puerto en cuestión. Al escribir un UNO en un bit de un TRIS queda programado el pin correspondiente como una entrada y al escribir un CERO queda programado como salida. Los pines vienen programados por defecto como entradas.
- **Muchas de ellas están multiplexadas para realizar una de varias funcione**s: por ejemplo: una misma línea pudiera ser configurada como entrada ó salida digital ó como una entrada analógica.
- Al igual que otros periféricos, **los puertos en los PICS** tratan de mantener las mismas características constructivas al migrar de un dispositivo a otro, esto facilita el aprendizaje de la tecnología.
- **Todas las líneas tienen protección contra ESD** (Electrostatic Discharge) y en el caso particular de las líneas de puerto tipo "B", existen resistencias de "PULL UP" internas, que pueden habilitarse o no por Software, limpiando el bit 7 del registro "OPTION".

1.4.- Descripción de los puertos integrados a la serie 16F87X.

En la **figura 2** se puede apreciar un "pin out" del microcontrolador que se trabajará en la elaboración del presente trabajo.

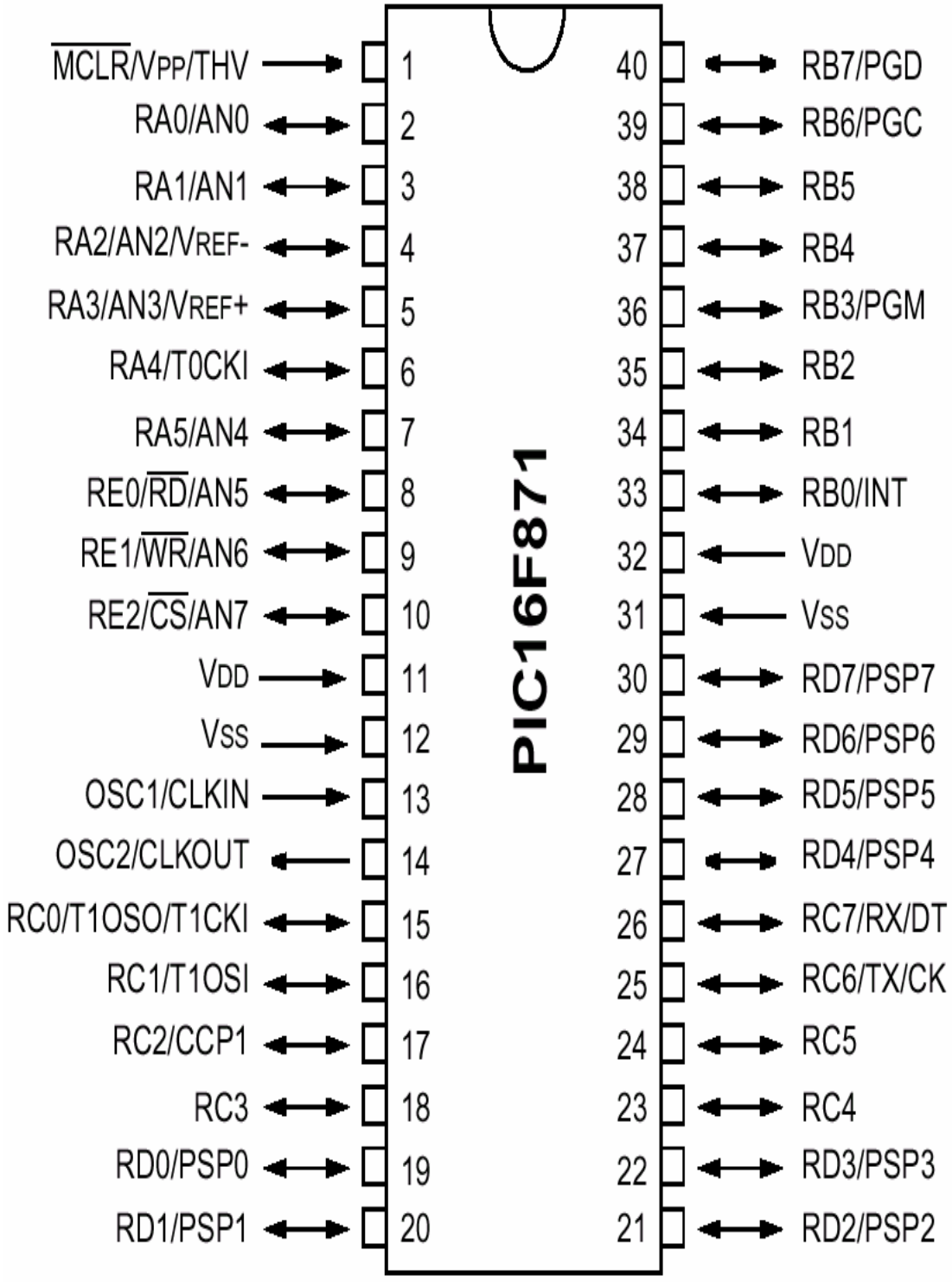


Figura2. Pin out característico de la serie 87x.

Todas las líneas del chip son líneas de puerto de propósito general excepto las que se resumen en la **tabla 1:**

Tabla 1. Pines con funciones especiales.

Nombre del pin	Número del pin	Función
(*)MCLR	1	Pin para generar un RESET externo[5].
OSC1	13	Pin para conexión del oscilador externo[6]
OSC2	14	Pin para conexión del oscilador externo[7]
VSS		Pin de referencia. Generalmente tierra.
VDD		Pin de Alimentación. Generalmente + 5V

Puerto A:

Es un puerto de entrada / salida de 6 pines (ra0... ra5) cuyas líneas pueden ser configuradas como entrada(s) / salida(s) digital(es) ó entrada(s) analógica(s); **excepto el pin ra**4, que es la entrada digital de conteo del timer0 8 cuando este es programado como contador de eventos. Los pines del puerto A se numeran en el chip desde el pin 1 hasta el pin 6.

Puerto B:

Es un puerto de entrada / salida **digital** de 8 bits, con pull up. El pull up se puede activar limpiando el bit 7 del registro OPTION_REG. El registro OPTION_REG es la localización 0x81 del banco1 de la Memoria RAM.

Puerto C:

Es un puerto de entrada / salida **digital** de 8 bits que multiplexa algunas funciones para sus líneas. Las líneas multiplexadas más interesantes son:

- **Rc2:** puede ser un pin de entrada / salida digital o la salida de una onda de PWM generada a partir de un recurso de HW denominado módulo CCP.
- **Rc6 y Rc7:** pueden ser pines de entrada / salida **digitales** o los pines de comunicación para el USART Tx y Rx respectivamente. Como los niveles de

salida en estos pines son CMOS (0 => 5V), es necesario conectarlos a un chip que convierta niveles TTL/CMOS a RS232

Puerto D:

Es un puerto de entrada / salida **digital** de 8 bits, que multiplexa funciones con el periférico denominado Puerto Paralelo Esclavo (PSP).

Puerto E:

Es un puerto de entrada / salida de 3 pines (re0, re1 y re2) cuyas líneas pueden ser configuradas como entrada(s) / salida(s) digital(es) ó entrada(s) analógica(s).

1.5.-Circuito oscilador.

Debe conectarse entre los pines OSC1 y OSC2 un cristal de la manera que se muestra en la **figura** 3. La conexión externa del cristal con dos condensadores conectados a tierra forma un oscilador con el inversor integrado al Pic y conectado entre los pines OSC1 y OSC2.

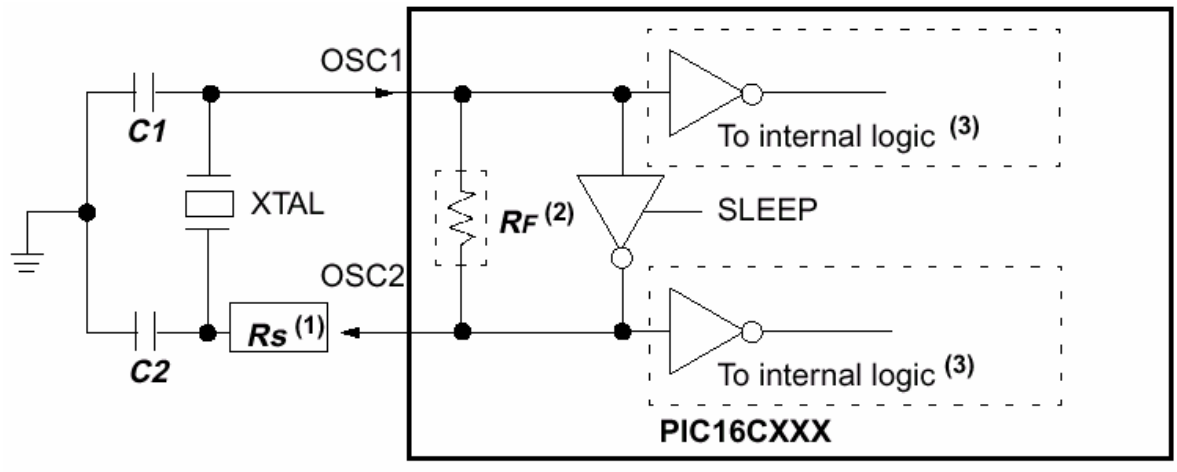


Figura 3. Configuración típica para el oscilador principal en los PICs de la familia media.

Los osciladores que puede utilizar esta familia se subdividen en las siguientes categorías:

- **H**S. Oscilador a cristal cuya frecuencia es típicamente mayor a 8 Mhz.
- **X**T. Oscilador a cristal cuya frecuencia es típicamente menor a 8 Mhz.

- **L**P. Oscilador basado en resonador cerámico o cuarzo de baja potencia, cuya frecuencia está en el orden de los KHz.
- **R**C. Este tipo de oscilador permite conectar, en lugar del tradicional cristal una resistencia y un condensador externo al pin **OSC**1, para formar junto a la circuitería integrada un oscilador de baja frecuencia muy económico pero también muy impreciso.

1.6.- Circuito de reset externo.

Generar un RESET al PIC significa cargar el Contador de Programas (PC) con el valor 0000h. Esto provoca que se ejecute la primera instrucción de cualquier aplicación, que es denominada "Vector de RESET". Para generar un RESET externo es necesario llevar a cero durante un determinado tiempo el pin 1 del chip (etiquetado como (*)MCLR). La **figura 4** muestra la circuitería necesaria para generar un pequeño pulso (activo en bajo) en el pin (*) MCLR.

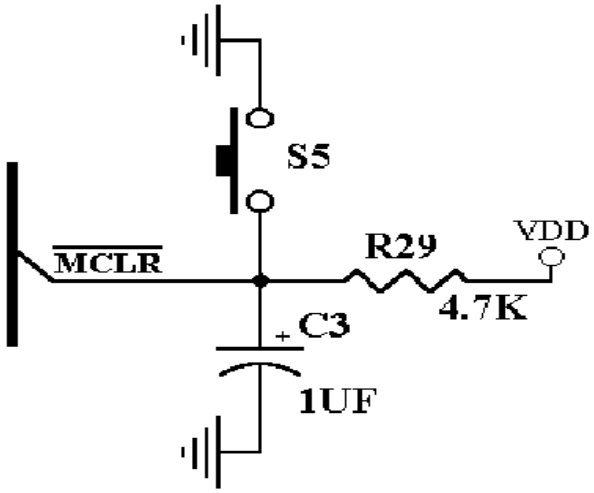


Figura 4. Circuito de RESET externo. El pin (*)MCLR se mantendrá en un nivel bajo durante el tiempo de carga del capacitor C3.

1.7.- Conexión de periféricos principales a los puertos.

1.7.1.- Conexión de Leds.

Los **leds** pueden conectarse directamente utilizando una resistencia limitadora para la corriente. Para activar un led es necesario que el pin al cual se conecta sea

programado como salida, escribiendo un cero en el bit del registro TRIS correspondiente. Existen 3 modalidades de conexión:

- El led se activa con 1 en el pin **(figura 5**a):

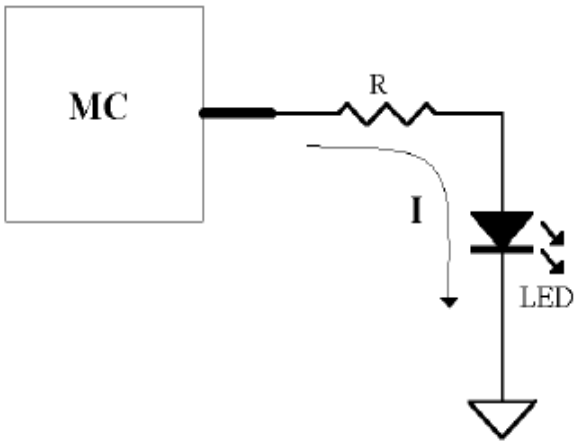


Figura 5a. Activación de un led con un "1" lógico.

- El led se activa con 0 en el pin **(figura 5**b):

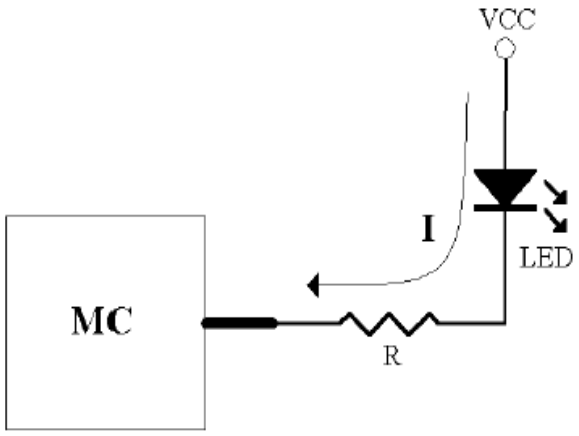


Figura 5b. Activación de un led con un "1" lógico.

- Conexión de **dos leds** a un mismo pin, uno activo con 0 y otro con 1 **(figura5**c):

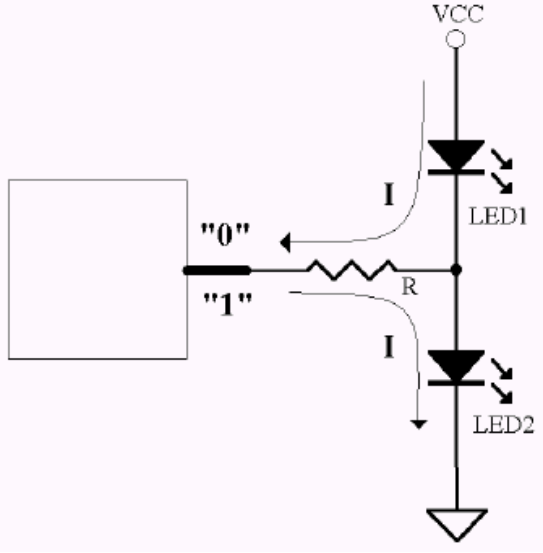


Figura 5c. Activación de dos leds con un mismo pin de puerto. El led1 se activa con "0" y el led2 se activa con "1".

En todos los casos anteriores la resistencia **R** puede calcularse suponiendo una corriente de activación para el led **(**I) de alrededor de 10 mA y un voltaje de activación para el led de unos **2**V. Bajo estas suposiciones:

R = (Vcc – 2V) / 10 mA.

Si se aproxima el Voltaje de "1" al valor de la fuente (Vcc).
Para **Vcc = 5V** se tiene que:

R = 300W W.

1.7.2.- Conexión de displays de 7 segmentos de Leds.

Un display de 7 segmentos es un arreglo de leds dispuesto como se muestra en la figura 6. Los segmentos se codifican en sentido horario empleando las letras **a, b** hasta la **g** y **PD (L y/o R)** si los displays contienen punto decimal. Cada display posee un terminal común **(CO**M) que sirve para activar o desactivar el display.

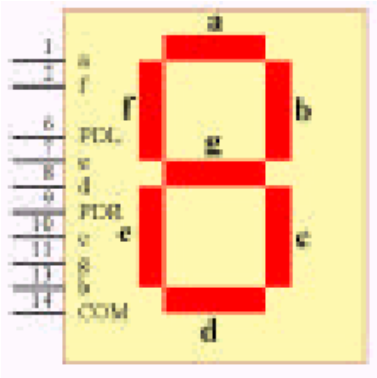


Figura 6. Esquema estándar para un Display de 7 segmentos.

Existen dos tecnologías de displays: **cátodo común y ánodo comú**n. En el primer caso todos los leds son activos con uno y el común es activo en cero. En el segundo los leds se activan con cero y el común con un uno. Dada la cantidad de corriente que circula por un común es necesario bufferear las líneas de puerto que activan los comunes empleando un transistor (PNP o NPN) dependiendo de la tecnología tal y como se muestra en la **figura** 7.

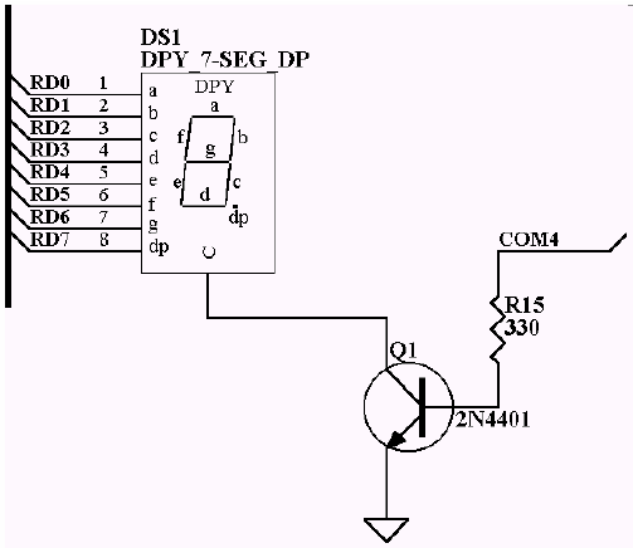


Figura 7. Esquema de activación para el común de un display de 7 segmentos de tecnología cátodo común.

Tradicionalmente los displays de 7 segmentos de leds eran tratados de forma independiente, utilizando 8 líneas de datos para encender cada led de cada uno. La conexión individual de displays resulta muy engorrosa dada la cantidad de líneas que

se involucran. El empleo de un **uC** permite implementar el manejo de displays de 7 segmentos multiplexado. El manejo multiplexado supone que se utiliza un único bus de datos (en nuestro caso el **puerto** D) para enviar el **dato en código 7 segmento** a los displays. Desde el punto de vista del hardware esto significa que los segmentos de los N displays involucrados se conectan en paralelo. Cada segmento (led) debe conectarse a través de una resistencia limitadora (de unos 220 ohm). Adicionalmente se necesitan líneas para manejar de manera individual los comunes de cada display.

El método multiplexado enciende en cada momento el display cuyo dato ha sido enviado al bus (puerto D). La secuencia de encendido se realiza a una frecuencia lo suficientemente alta como para provocar la sensación de persistencia de los datos. En la **figura 8** se muestra la filosofía del refrescamiento multiplexado de displays.

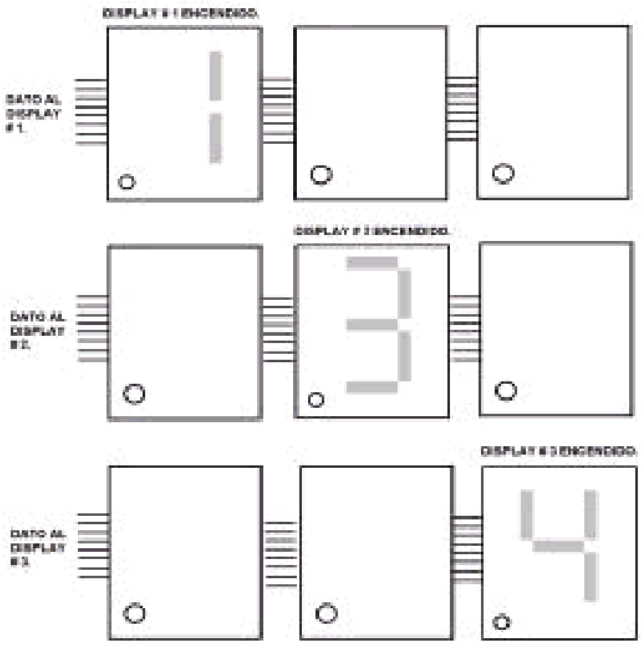


Figura8. Esquema que ilustra el tratamiento multiplexado de displays de 7 segmentos.

1.7.3.- Conexión de teclas “Push Button”.

Las líneas que se utilicen para conectar teclas deben programarse como entradas, puesto que las teclas son periféricos de entrada al MC.

Para conectar teclas resulta excelente el puerto B puesto que posee resistencias de pull ups internas como ya se había señalado. En la figura 9 se muestra la conexión

directa de pulsantes a través de una resistencia de 220 ohm. Las teclas entregan en este caso un nivel bajo cuando son pulsadas y un nivel alto cuando están en estado abierto.

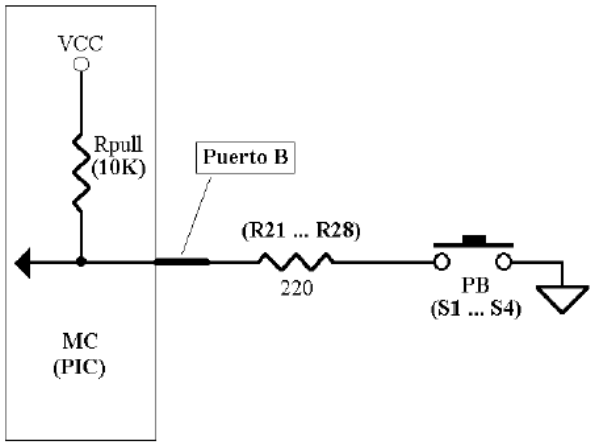


Figura 9. Esquema general de conexión de un pulsante tipo "push buttom".

La resistencia de 220 ohm se coloca para proteger la línea de puerto contra cortocircuitos a tierra ante una situación indeseada como la siguiente: Suponga que por error ud. programa los pines del **puerto B** como **salidas (registro TRISB = 00000000)** y además escribe un "1" en la línea a la cual conectada una tecla. Si la resistencia no estuviera presente, al pulsar la tecla se conectaría a tierra la línea de puerto que en ese momento está aun voltaje alto, por lo que pudiera circular una corriente por ella lo suficientemente alta como para producir daños. A pesar de la existencia de la resistencia limitadora de 220 ohm, el nivel de "0" no se afecta

1.7.4.- Conexión del teclado matricial.

La conexión de un teclado matricial de 4 columnas por 4 filas es como sigue: Las columnas se conectan a los pines rb3, rb2, rb1 y rb0. Tales pines deberán ser programados como entradas con "pull-up" activo. Por otra parte, las filas se conectan a los pines rb7, rb6, rb5 y rb4. Tales pines deberán configurarse como salidas.

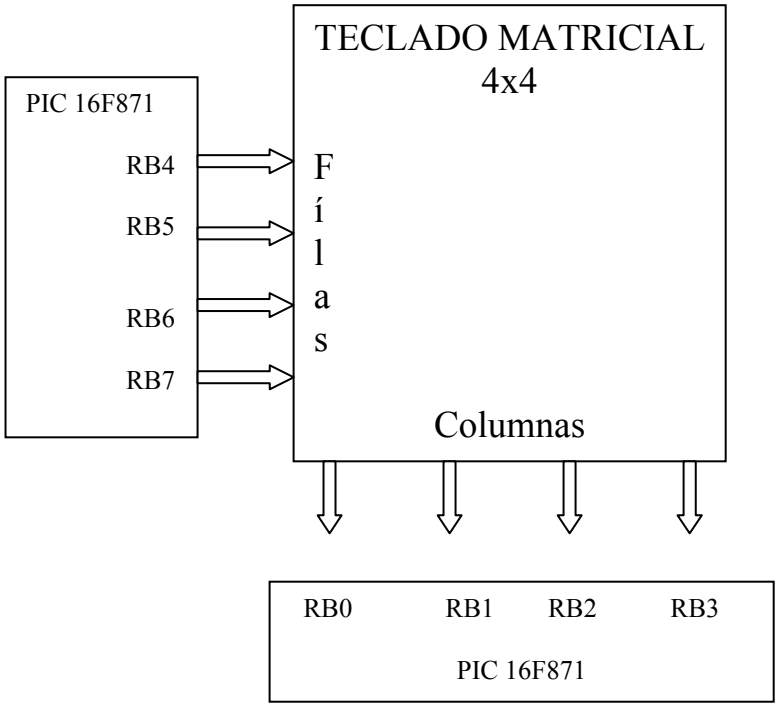


Figura 10. Esquema de conexión de un Teclado Matricial 4x4.

1.7.5.- Conexión de relés

Usualmente la bobina de la mayoría de los relevadores posee baja impedancia y trabaja a un nivel de voltaje superior al de alimentación del MC. Estas dos razones hacen necesaria la conexión de relés utilizando transistores, que sirven de buffers y acopladores de nivel. En la **figura 11** se muestra el acople de un relé a una **salida digital** empleando un transistor NPN. La activación del relé se produce en este caso con un "1" lógico.

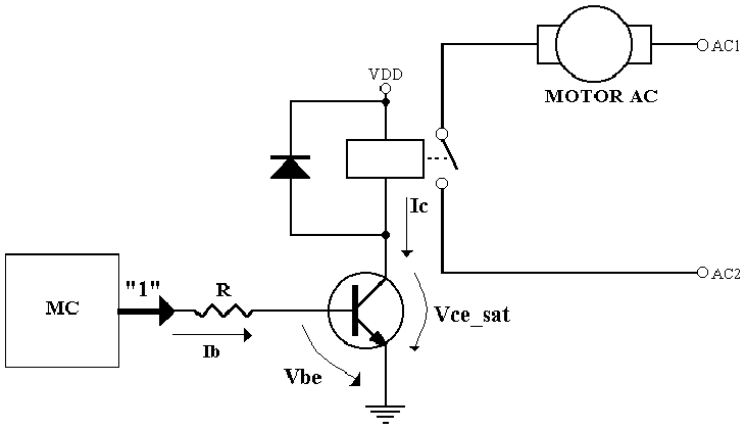


Figura 11. Esquema de conexión para relé.

1.7.6.- Acoplamiento de cargas opto acopladas.

Se utiliza optoacoplamiento para aislar cargas que pudieran introducir interferencias en el funcionamiento del MC. Tal es el caso de los motores, debido al ruido inductivo, o de conmutación en el caso del motor de escobillas que generan. En otros casos se implementa optoaislamiento cuando no se desea compartir la misma referencia para la tarjeta de control y algún elemento externo por razones de seguridad eléctrica. Tal es el caso de muchos equipos destinados a uso médico o a zonas con alto riesgo de explosión. Seguidamente se ofrecen algunas alternativas de conexión para diferentes tipos de carga:

- **Acoplamiento de cargas en régimen de corriente continúa.**

En las **figuras 12 a y b** respectivamente se muestra un esquema alternativo para la conexión de una salida y una entrada digital.

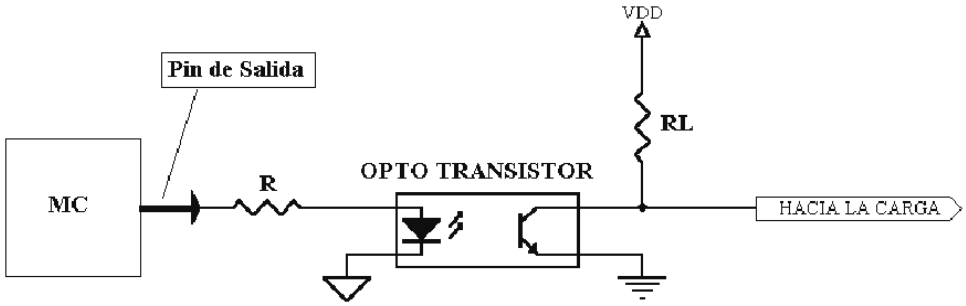


Figura 12 a. Conexión aislada de un pin de salida que maneja una carga que trabaja en régimen de DC.

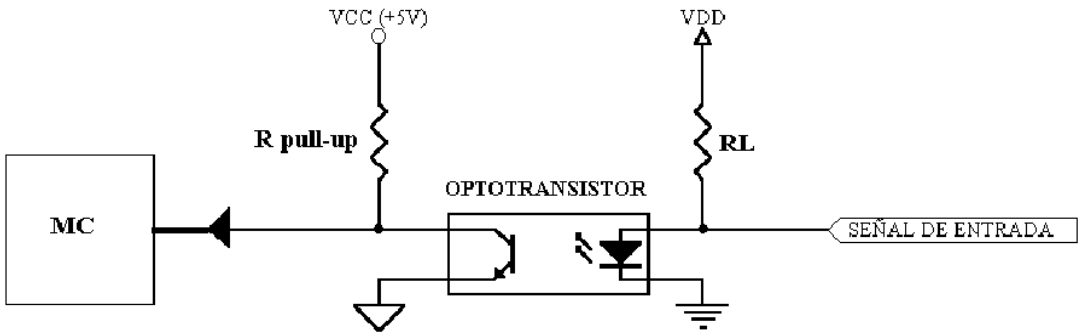


Figura 12 b. Conexión aislada de una señal a un pin de entrada digital.

En el caso de la **salida digital** el fototransistor es excitado por la señal de luz que se genera en el led conectado al MC. Observe que el "lado del led" comparte la referencia del MC y el "lado del fototransistor" comparte la referencia de la carga, por lo tanto este es un esquema de conexión aislada. Por otra parte la **entrada digital** es conmutada por el fototransistor, cada vez que es activado por los pulsos de luz que se generan en el led a partir de la señal de entrada. Observe que en este caso el "lado del led" comparte la referencia de la señal de entrada y el "lado del fototransistor" comparte la referencia del MC, por lo tanto también este es un esquema de conexión aislada.

- **Acoplamiento de cargas en régimen de C.A.**

En la **figura 13** se representa un esquema de acople elemental de **una salida digital** con una carga que trabaja en régimen de AC. Es probable que para motores de gran potencia, se necesite un interfaz adicional entre el optotriac y el motor.

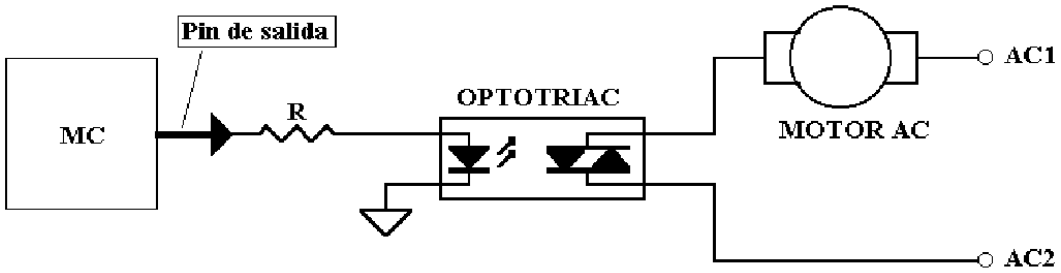


Figura 13. Acople con una carga que trabaja en régimen de AC.

1.8.- Arquitectura Interna de los Microcontroladores Microchip.

La **Arquitectura Interna** de los procesadores tipo **RIS**C, como los PICs se caracteriza (entre otros) por los siguientes aspectos:

- Utilizan Arquitectura tipo HARVARD.
- Instrucciones largas que se codifican en una sola palabra.
- Arquitectura "registro-fichero" para la Memoria de Datos.
- Instrucciones simétricas.

Arquitectura HARVARD.

Este tipo de arquitectura supone la separación de los Buses que comunican la *Memoria de Programas* y la *Memoria de Dat*os, a diferencia de la tradicional arquitectura tipo Von Neuman **(figura. 14).**

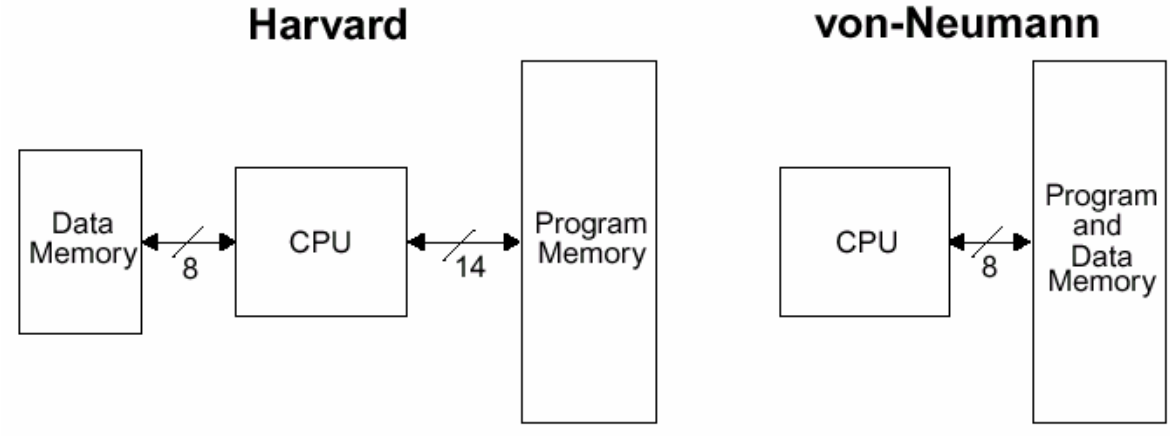


Figura 14. Arquitectura tipo Harvad vs. Arquitectura tradicional "Von Neuman".

La **separación de buses** posee las siguientes ventajas:

- Acceso a la Memoria de Datos (MD) y la Memoria de Programas (MP) al mismo tiempo, esto permite realizar multitarea, esto es: a la misma vez que se busca el código de una instrucción es posible ejecutar la que ya se decodificó.
- Como los buses son separados, es viable construir un bus para la Memoria de Programas más extenso, esto hace posible que el código de las instrucciones pueda ser grabado en la Memoria de Programas en una sola palabra de longitud mayor a 8 bits 2 (longitud tradicional). La codificación de las instrucciones en una sola palabra es ventajosa porque de esta manera es posible decodificar toda la instrucción en un solo acceso a la MP, pues el código de la instrucción se lee de una vez.

Simetría (Ortogonalidad de la Instrucciones).

Un Set de instrucciones es simétrico, cuando con una misma instrucción es posible direccionar y actuar, ya sea sobre una localización de RAM o un registro de

propósito específico. Esta característica es indispensable para lograr un Set de Instrucciones tipo **RISC.** La simetría es posible gracias a que la Memoria de Datos soporta un tipo de arquitectura denominada "registro-fichero", que permite direccionar de igual manera un **Registro de Propósito Específico** y uno **de Propósito Genera**l. Los procesadores PICS, incluyen dentro del mismo espacio de Memoria de Datos a las localizaciones de propósito general (o del usuario) y a las dedicadas a propósitos específicos (registros especiales). Los **Registros de Propósito General (G.P.R)** sirven para almacenar datos que garantizan el funcionamiento de determinada aplicación. Pueden ser utilizados "libremente". Por otra parte los **Registros de Propósito Específico (S.F.R)** almacenan información relacionada con el funcionamiento del Microcontrolador.

Ciclo de Instrucción en Microcontroladores PICS.

Ciclo de Instrucción.- La ejecución de lo que se denomina **"Ciclo de Instrucción (CI**)" comienza en el estado 1 del **Ciclo de Máquina 4** y *es el tiempo que demora la ejecución de una instrucción.* Un CI puede requerir de uno o más **Ciclos de Máquina** (CM) en función de las características de cada instrucción.
En los PICS la ejecución de la mayoría de las instrucciones dura **un** ciclo de máquina, por lo tanto en la mayoría de los casos 5 el CM = CI.
Un **CM en PICs** es igual a 4 veces el período del oscilador externo. Por ejemplo: si la frecuencia del oscilador externo es igual a 16 Mhz, se tiene un CM = 0,25 mS.

1.9.- Arquitectura de la Memoria de Programas en los PICs.

En la **Memoria de Programas** (MP) quedan almacenados los códigos de las instrucciones de determinada aplicación (programa). Es una memoria no volátil, por lo que la información grabada en ella permanece invariante al suprimir la alimentación al chip. En los MC de la Familia Media de PICs la MP es una memoria tipo "Flash" o "EEPROM", por lo que puede borrarse y escribirse de manera rápida utilizando electricidad.

La MP en los PICs de la familia media está organizada en páginas de "2K palabras cada una". Decimos "2K palabras cada una", porque los códigos de las localizaciones de la **MP** en esta familia tienen una longitud de **14 bit**s. En el primer epígrafe se discutían las ventajas asociadas con esta característica. Los PICS de esta familia poseen como máximo **4 páginas de "2K palabra**s", lo que hace un total de 8192 posibles instrucciones para cualquier programa. Para direccionar esta cantidad de memoria se necesitan **13 bits (2 13 = 8192).**

Para direccionar la **MP** existe un contador – puntero **denominado Contador de Programas (PC**). El PC almacena en cada momento **la dirección** de la instrucción que se está ejecutando. Debe aclararse que el PC sólo almacena la dirección de la instrucción, no la instrucción. Como la cantidad máxima posibles de instrucciones de un programa es 8192, el PC es un contador de 13 bits de longitud. En la **figura 15** se presenta un esquema general de la MP. Debe considerarse que no todos los procesadores de la familia poseen las 8K palabras de memoria. Por ejemplo el PIC16F871 posee sólo 2K palabras y el PIC16F877 posee 8K palabras.

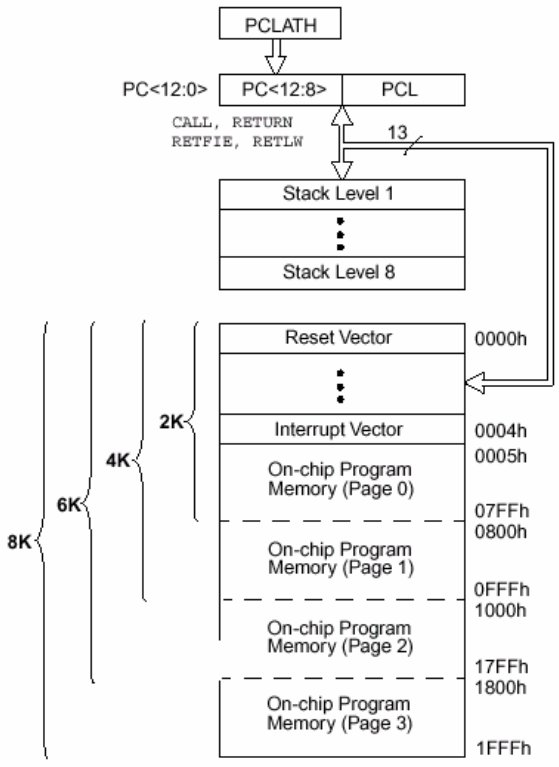


Figura 15. Esquema de la Memoria de Programas.

El salto entre páginas de la MP se logra escribiendo los dos bits más significativos (bits 11 y 12) de PC. El PC no puede escribirse de una vez porque es de 13 bits, por lo que el fabricante implementó dos registros en la Memoria RAM: el PCLATH y el PCL. El PCLATH es un registro de 5 bits que almacena la parte alta del PC y el PCL es un registro que almacena los 8 bits menos significativos.

Aspectos relevantes de la MP.

Vector de Reset y Vector de Interrupción.

El **Vector de Reset** es la primera instrucción que deberá ejecutar el PIC después de un RESET Se graba en la localización 0000h de la MP. El **Vector de Interrupción** es siempre una instrucción de salto incondicional (goto XXXX) a la dirección (XXXX) de la **subrutina 6** donde se atiende(n) la(s) interrupción(es). **El Vector de Interrupción** se coloca siempre en la localización cuya dirección es **0004h.** El Vector de Interrupción estará presente si se trabaja con interrupciones en la aplicación que se está desarrollando.

Direccionado de la Memoria de Programas.

"Direccionar" la MP significa apuntar (en el sentido directo "señalar") a la instrucción que en cada momento debe ejecutarse. Durante la ejecución de un programa el direccionado puede ocurrir secuencialmente de manera automática a medida que el PC se va incrementando o se pueden producir "saltos". Un salto o ruptura del proceso secuencial de la ejecución de un programa se produce cuando se ejecutan instrucciones de tipo **call, goto, return, retfie, retlw** ó en determinados casos en instrucciones del tipo **btfss f,b** y **btfsc f,b.**

La instrucción **"call"** permite realizar "llamadas" a subrutinas. Una subrutina es un programa que se ejecuta cada vez que sea llamado. Cualquier subrutina que sea llamada con un **call** debe terminar en una instrucción **retur**n. La instrucción call modifica el contenido del PC y por lo tanto provoca un salto dentro de la MP.

La instrucción **"goto"** se utiliza para "desviar" la ejecución de un programa hacia otro conjunto de instrucciones. Se diferencia del **call** en que no retorna a la instrucción que le sucede a menos que se ejecute un nuevo **goto** hacia la instrucción deseada al final del proceso que se invocó. La instrucción **goto** modifica el contenido del PC y por lo tanto provoca un salto dentro de la MP Un **goto** no permite recuperar el valor del PC después de ejecutado un proceso.

Por último las instrucciones **retfie** y **retlw** son casos de "retornos especiales". El primero permite retornar al **programa principal** desde una subrutina de atención a interrupción y el segundo retorna de una subrutina con un número grabado en el registro W ("working register").

1.10.- Arquitectura de la Memoria de Datos en PICs.

La **Memoria de Datos (MD)** es una memoria de tipo volátil 7 formada por registros de 8 bits, que comparte espacios para el almacenamiento de los datos necesarios para el funcionamiento de cualquier aplicación **(Registros de Propósito Genera**l) y para el funcionamiento del Microcontrolador **(Registros de Propósito Específic**o). La **MD** es una memoria también paginada. Cada página en este caso se refiere como banco. Los procesadores de la serie 87X, poseen hasta **4 bancos** de 128 **bytes** cada uno; lo hace un total de 512 localizaciones. A diferencia de la Memoria de Programas la MD es volátil de tipo RAM (Random Access Memory) y sus localizaciones son de 8 bits.

Como ya se había señalado la MD soporta Arquitectura Registro – Fichero, lo que significa que los SFRs y GPRs comparten el mismo espacio (en este caso banco).
En cada banco los SFRs ocupan siempre las primeras localizaciones y los GPRs las localizaciones superiores. Algunos SFRs que se utilizan con mucha frecuencia aparecen replicados en varios o todos los bancos. En el Anexo1 se muestra un **"Mapa" de la Memoria de Dato**s.

Direccionado de la Memoria de Datos.

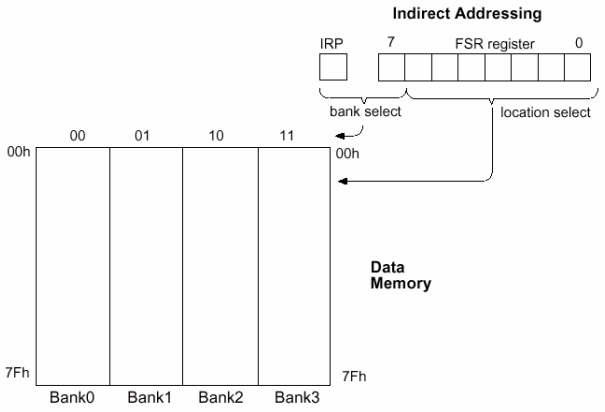
Para direccionar la MD es necesario establecer la dirección dentro del banco (cada banco posee 128 localizaciones) y en cuál de los **4 bancos** posibles se está trabajando. Para seleccionar el **banco** a direccionar se necesitan dos bits.

Modos de direccionado de la Memoria de Datos.

Las direcciones en la **MD** podrán tener como máximo 9 bits pues existe un total de 512 localizaciones (2 9 = 512). Basado en esto, existen dos modos para formar la dirección completa de una localización de la MD: directo e indirecto.

Modo directo.

En el **Modo Directo** (figura 16 abajo) 7 bits de la dirección vienen en el código de la instrucción. Estos 7 bits son la dirección de un registro (SFR ó GPR) dentro de uno de los **4 bancos** posibles (cada banco posee como máximo 128 (7F) bytes). Para formar la dirección completa faltarían **dos** bits; estos dos bits son los **bits 6 y 5** del registro STATUS (denominados RP1 y RP0 respectivamente). El registro STATUS se encuentra en la dirección 03h de los 4 bancos, o sea es la cuarta localización de c/u de los bancos.



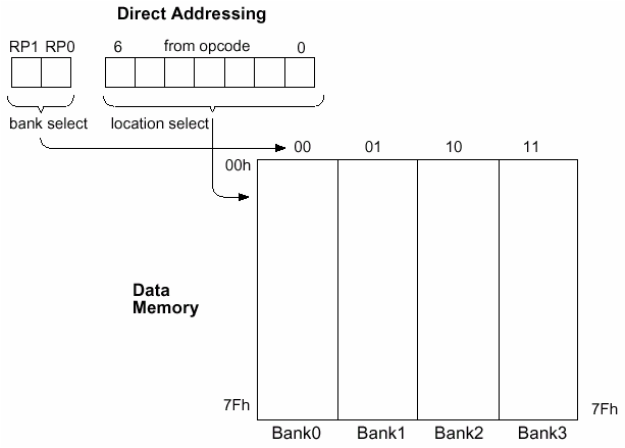


Figura 16. Modos de Direccionado de la MD en los MC PICS. Indirecto (izquierda). Directo (derecha).

Modo indirecto.

En el **Modo Indirecto** (figura.16 arriba), la localización de memoria se direcciona mediante un puntero: el registro FSR y se modifica mediante un registro auxiliar, el INDF. El proceso de direccionado ocurre de la manera siguiente:

En el registro FSR **(**puntero de la MD) se cargan los 8 BITS de la dirección base. Como la dirección cargada es de 8 BITS, esta puede referirse a una localización que se encuentre ubicada entre las direcciones 00h y la FFh ó entre la dirección 100 y 1FF, lo que significa que es necesario también definir a que bloque nos referiremos, si al **banco 0 y 1** ó al **banco 2 y** 3. Para formar entonces la dirección completa, es necesario entonces definir el estado del bit "IRP", que es el BIT # 7 del registro STATUS. Una vez que la dirección ha sido formada es necesario "actuar" sobre el registro de la RAM que está siendo apuntado. Para operar sobre el registro se utiliza un registro auxiliar denominado INDF. El INDF ocupa la dirección 00h de todos los bancos.

1.11.- Programación de los Microcontroladores Microchip.

Como ya habíamos señalado, los PIC´s son microcontroladores construidos de tal manera que poseen un set de instrucciones reducido (arquitectura RISC).

Recordemos que un procesador RISC **(Reduced Instruction Set Compute**r) posee las siguientes ventajas:

Debido a que realizan la canalización (pipelining), son mucho más veloces, porque realizan procesamiento paralelo. Para los programadores esto es una ventaja substancial pues solo deben aprender un número pequeño de instrucciones (en esta familia, solo un número de 35). A pesar de ser pequeño, este juego de instrucciones tiene casi las mismas prestaciones, en términos de **capacidad de código** que el juego de los microcontroladores convencionales, que a priori parecen mucho más completos. Otras características ventajosas de las instrucciones que manejan los MC microchip, están relacionadas con la filosofía Harvard de su arquitectura:

· **Instrucciones simétrica**s. Las instrucciones de este tipo hacen posible llevar a cabo cualquier operación sobre cualquier registro usando cualquier modo de direccionado. UD. se refiere a una localización de RAM con la misma instrucción que a un FSR o a un BIT de una localización o de un puerto.
· **Palabras de instrucciones largas.** Como el bus de instrucciones es separado, este se puede hacer de modo que se cargue de una vez el código de la instrucción.

· **Instrucciones de una sola palabra.** Relacionado con lo anterior, la instrucción se codifica en una sola palabra, en este caso de 14 bits.

1.12.- Tipos o categorías de instrucciones microchip.

Las instrucciones se agrupan en 3 categorías:

- Instrucciones orientadas a Byte.
- Instrucciones orientadas a Bits.
- Instrucciones para operaciones con literales y de control.

Orientadas a Byte.

En estas instrucciones un operando “f” representa un registro determinado y “d” representa el destino que tendrá el resultado de la operación que se realiza, f” es un número de 7 bits que puede representar un Registro de Propósito Específico (SFR) o

un Registro de Propósito General (GPR) dentro de la RAM. "d", solo puede tener dos valores [0,1], si "d = 0" significa que el resultado de la operación realizada será depositado en el registro acumulador (W) y si es = 1 significa que el resultado quedará grabado en el propio registro "f" especificado en la instrucción.

Ejemplos:

ADDWF	**suma W y "f"**
Sintaxis	ADDWF f,d
operandos	$0 \leq f \leq 127$ $d \in [0,1]$
operación	(W) + (f) → **destino**
banderas que afecta	C, DC, Z
descripción	**Suma el contenido del registro "f" con W. Si d = 0, el resultado se coloca en W, si es = 1 el resultado se coloca en el propio registro "f"**

Figura 17. Ejemplos de Instrucciones Orientadas a Byte.

DECF f,d	**decrementa el reg "f"**
Sintaxis	DECF f,d
operandos	$0 \leq f \leq 127$ $d \in [0,1]$
operación	(f) - 1 → **destino**
banderas que afecta	Z
descripción	**Dec. el contenido del reg "f". Si d = 0 el resultado se coloca en el acumulador y si d = 1 el resultado se coloca en el registro "f".**

Orientadas a Bits.

Esta categoría está especialmente dedicada al **manejo independiente de Bit**s, siguiendo el mismo principio de simetría, o sea con estas instrucciones se pueden

manejar por separado bits de cualquier registro, ya sea un Registro de Propósito Específico o un Registro de Propósito General. En estas instrucciones un operando "b" representa el número del BIT dentro del registro "f", que será afectado por la operación, por lo tanto, como todos los registros son solo de 8 bits, "b" es un número decimal que va desde 0 hasta 7. "f" es un operando similar al empleado en la categoría anterior, o sea un número entre 0 y 127 (decimal).

Ejemplos:

BCF	**lleva a cero el bit "b" del registro "f"**
Sintáxis	[*label*] BCF f,b
operandos	$0 \le f \le 127$ $0 \le b \le 7$
operación	$0 \rightarrow$ (f<b>)
banderas que afecta	**no afecta ninguna bandera**
descripción	Bit 'b' in register 'f' is cleared.

Figura 18. Ejemplos de Instrucciones Orientadas a Bits.

BTFSC	**Realiza el "test" del bit "b", de cualquier registro "f" y si es igual a 0, no ejecuta la instrucción que sigue.**
Sintáxis	BTFSC f,b
operandos	$0 \le f \le 127$ $0 \le b \le 7$
operación	skip if (f<b>) = 0
banderas que afecta	**ninguna**
descripción	**Si el bit "b" en el registro "f" es 1 se ejecuta la instrucción que sigue. En este caso el "btfsc" demora un ciclo. Si el bit "b" es igual a cero, entonces no se ejecuta la proxima instrucción. En este caso la demora es de dos ciclos de instrucción.**

Instrucciones para operaciones con literales y de control.

Estas instrucciones, están diseñadas para realizar operaciones aritméticas y lógicas sobre literales (constantes) y para cambiar el contenido del contador de programas de manera incondicional. Emplean el operador "k" que es un valor de 8 ó 9 bits según sea el caso.

Ejemplos:

ADDLW	Add Literal and W
Sintáxis	ADDLW k
operandos	$0 \leq k \leq 255$ Note que en este caso "k" es un número de 8 bits, porque es una operación aritmética.
operación	$(W) + k \rightarrow (W)$
banderas que afecta	C, DC, Z
descripción	Suma el contenido del acumulador con el número de 8 bits "K" y coloca el resultado siempre en el acumulador.

Figura 19. Ejemplos de Operaciones con literales (arriba) y de control (siguiente pg).

CALL	Llamada a Subrutina.
Sintáxis	CALL k
operandos	$0 \leq k \leq 2047$ En este caso "k" es un número entre cero y 3ff
operación	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
banderas que afecta	Ninguna.
descripción	Primero, guarda la dirección PC+1 en la pila. Los 11 primeros bits de la dirección del salto se cargan en el PC (bits 10 : 0). Los bits superiores (11 y 12) se cargan desde el registro PCLATH, con esto se forma al fin la dirección de 13 bits. El CALL es una instrucción que siempre dura dos ciclos.

Formato General de Codificación de las tres categorías que existen.

Las instrucciones Microchip se codifican en una sola palabra, en el caso de la familia media la longitud de esta palabra es de 14 bits. Dentro de la palabra de 14 bits, según la categoría o tipo de instrucción, existen grupos de bits que codifican cada parte de la instrucción.

En la figura se muestra el formato para cada una de las tres categorías, analizadas en el epígrafe anterior:

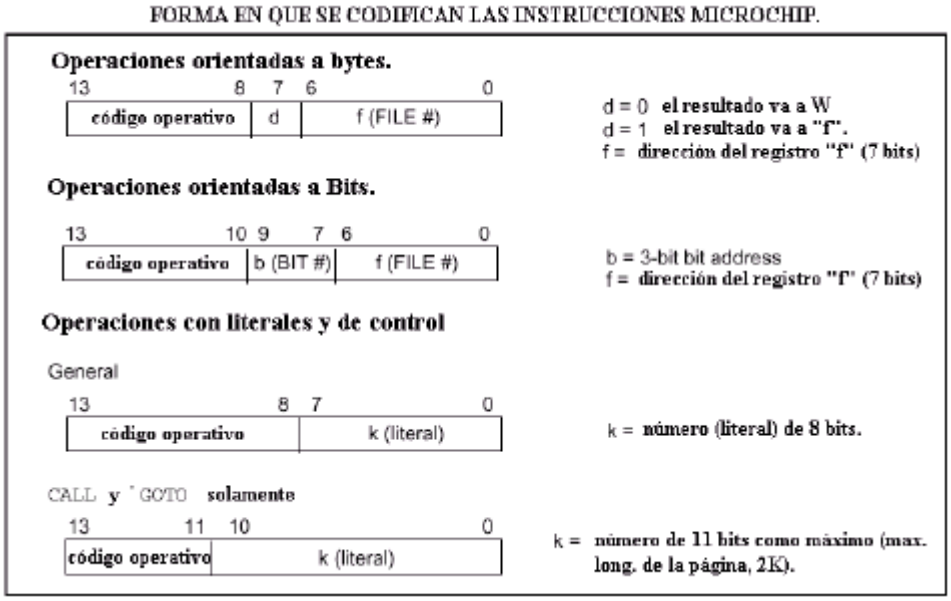


Figura 20: Formato general de las 3 categorías de Instrucciones Microchip.

Duración de las instrucciones.

La mayoría de las instrucciones Microchip se ejecutan en un solo ciclo de instrucción, que es igual a 4 veces el período del reloj externo, aunque existen las siguientes excepciones:

Se ejecutarán siempre en dos ciclos de instrucción:

- **CALL k** Llamada a una subrutina.
- **GOTO k** Salto incondicional a la dirección “k”.
- **RETFIE** Retorno de una subrutina de atención a interrupción.
- **RETURN** Retorno de un CALL.
- **RETLW k** Retorno de un CALL pero grabando el número “k” en el Acum.

Se ejecutarán en uno o dos ciclos de instrucción:

- **DECFSZ f, d** Decrementa el registro “f” y salta la instrucción siguiente si “f” llegó a cero (en este caso dura dos ciclos).

- **INCFSZ f, d** Similar, pero incrementando “f”, o sea se salta la instrucción siguiente cuando “f” sufre un “overflow”. En este caso la instrucción demora dos ciclos.

- **BTFSC f, b** Hace el test del BIT “b” dentro del registro “f” y salta la próxima instrucción cuando este está en cero. En este caso la instrucción dura dos ciclos.

- **BTFSS f, b** Hace el test del BIT “b” dentro del registro “f” y salta la próxima instrucción cuando este está en uno. En este caso la instrucción dura dos ciclos.

1.13.- Sensores.

A los sensores actualmente se les denomina transductores, ya que transforman un fenómeno físico a una señal de tipo eléctrico de una forma inmediata. Para instalar los sensores hay que tener en cuenta que los normalmente cerrados (N.C) se conectan en serie entre si y los normalmente abiertos (N.A) en paralelo. En el caso de nuestra alarma utilizaremos dos tipos de sensores: Para las puertas utilizaremos simples interruptores N.C o N.A que generalmente vienen ya instalados en el vehículo de fabrica, ya que estos se utiliza para encender la luz de salón cundo se abre una de las

puertas, pero en algunos casos puede ser necesario utilizar en lugar de los interruptores de las puertas, sensores magnéticos; estos constan de un imán permanente que se fija a la puerta y el denominado Reed switch que se fija al parante de la misma y no hace otra cosa que la de abrirse ante la no presencia cercana del imán permanente, es decir cuando la puerta esta cerrada las dos partes se juntan, cerrándose de esta manera el Reed switch ante la presencia del imán y al abrirse la puerta ocurre lo contrario ya que las dos partes se separan.

Por último utilizaremos también un sensor de golpe o vibración, el mismo que detecta como su nombre lo indica cualquier golpe o vibración en el caso de que se intente forzar las seguridades de las puertas o sustraer algún accesorio del vehículo.

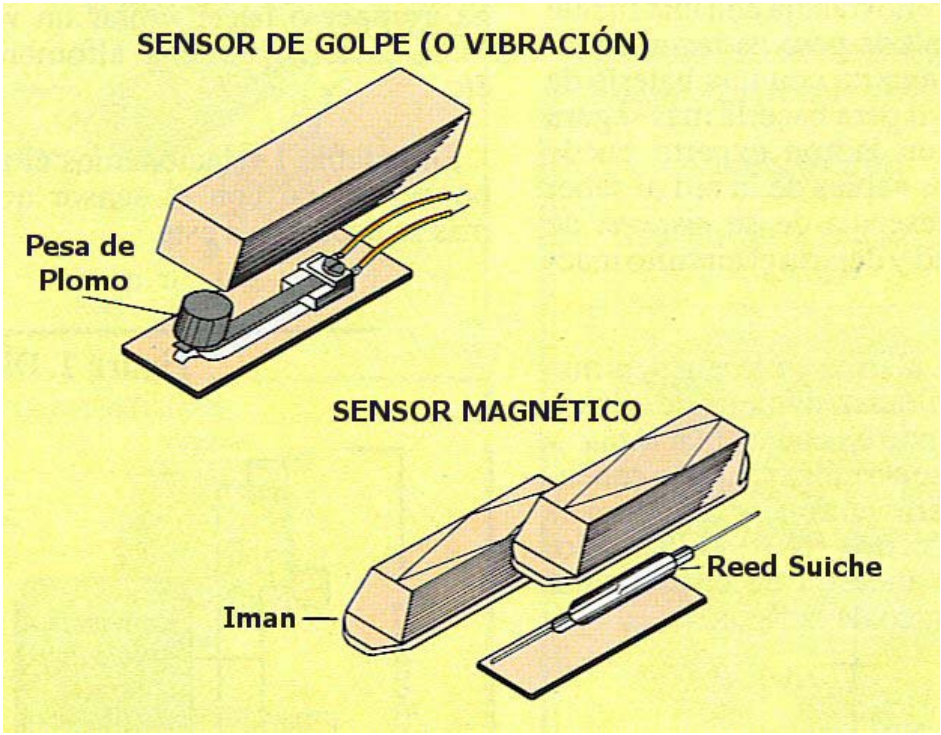


Figura 21. Tipos de sensores utilizados.

Conclusiones:

A lo largo de este capitulo se ha desarrollado toda la teoría necesaria para tener una idea clara y precisa acerca del manejo tanto de software como de hardware de los microcontroladores de la serie PIC16F87X, con estos conocimientos adquiridos podremos desarrollar en el posterior capitulo un conjunto hardware-software acorde a las necesidades del medio con partes, accesorios, dispositivos o componentes existentes en el mercado local y que se ajusten a los objetivos propuestos en el presente trabajo de grado.

CAPITULO 2: DISEÑO Y CONSTRUCCIÓN DE UN MODULO DE ALARMA CODIFICADA PARA VEHÍCULO.

Introducción:

En el presente capítulo abordaremos la parte práctica del trabajo de tesis, aplicando los conocimientos teóricos adquiridos en lo que se refiere a hardware como a software en el capítulo previo para así lograr un diseño de una alarma codificada para vehículo apropiada, versátil, conveniente y aplicable a las condiciones de los diferentes vehículos de nuestro medio, con ello alcanzaremos que el mencionado dispositivo cumpla con los objetivos, condiciones, secuencias y proyecciones planteadas en el presente trabajo.

2.1.- Descripción general del módulo de Alarma.

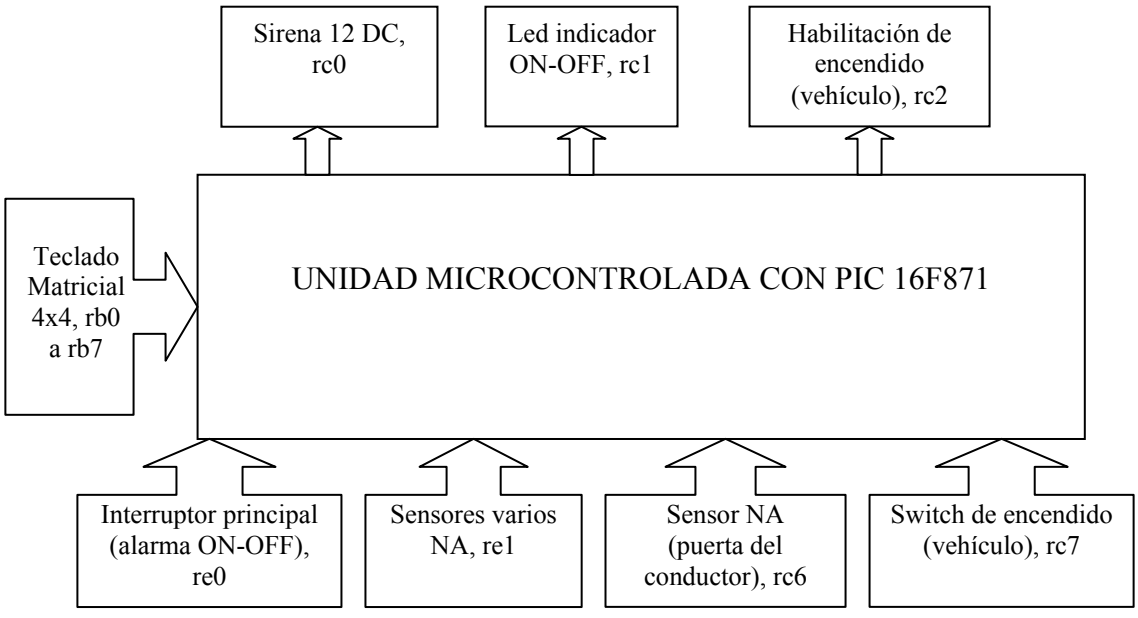


Figura 22. Diagrama de bloques del Módulo de Alarma codificada para vehículo

En la figura 22 se puede observar un diagrama de bloques del circuito que nos ocupa en este capítulo, en el mismo tenemos La Unidad Microcontrolada con PIC 16F871

que es la base del módulo de alarma codificada para vehículo, la mencionada unidad microcontrolada se puede decir que hace la función de cerebro ya que a la final es un pequeño computador diseñado específicamente para dar lectura a los periféricos de entrada como:

- Teclado Matricial 4x4 es decir 4 filas x 4 columnas que da un total de 16 teclas, rb0 a rb7.
- Interruptor principal On-Off, re0.
- Sensores varios NA es decir normalmente abiertos, re1.
- Sensor NA colocado en la puerta del conducto, rc6.
- Switch de encendido con el que viene equipado todo vehículo, rc7.

Y de acuerdo a su estado o manipulación hacer reaccionar a los periféricos de salida como:

- Sirena 12 DC, rc0
- Led indicador ON-OFF, rc1
- Habilitación de encendido (vehículo), rc2

En otras palabras la Unidad microcontrolada hará reaccionar a los periféricos de salida activándolos o desactivándolos de acuerdo a los cambios que sufran los periféricos de entrada cuando estos sean manipulados, esto se lograra gracias a un programa (software) contenido en el PIC16F871.

2.2.- Descripción de sensores utilizados en el módulo de alarma.

En este punto describiremos la función que cumplirá cada sensor de acuerdo a los pines del PIC16F871 (Fig.22) al que estén conectados, cave recalcar que aunque en los pines programados como salidas no están precisamente conectados sensores, estos también serán descritos en este mismo punto para evitar confusiones posteriores.

- Rb0 a Rb7.- Están programados como entradas: rb0, rb1, rb2 y rb3 y como salidas: rb4, rb5, rb6 y rb7, teniendo como función la de “scaniar” el Teclado

Matricial para determinar cual de las dieciséis teclas del mismo es pulsada (Ver Figura 10). Este teclado Matricial estará ubicado en el interior del vehículo y a través de este ingresaremos una clave de uno o varios dígitos (hemos escogido 5 en nuestro caso) con el objeto de desbloquear el sistema de encendido del motor con ello el motor de arranque dará movimiento a la máquina del vehículo más esta nunca encenderá si no se ha marcado la clave correcta previamente.

- Rc0.- Esta programado como salida y esta encargado de dar la señal de activación a un transistor que a su ves alimenta a la bobina de un relé (Ver Figura 11), activando este último la sirena de 12 DC a través de sus contactos normalmente abiertos, indicando que la alarma fue disparada a través de uno de sus sensores; también se puede dar una señal visual cuando la alarma es disparada, interconectando el cable de alimentación del rele de las luces intermitentes o también llamadas luces de parqueo del vehículo a uno de los contactos del rele de activación de la sirena, haciendo esto la sirena y luces intermitentes funcionaran en conjunto dándonos de esta manera una señal visual como sonora cuando las seguridades del vehículo han intentado ser o han sido violentadas.

- Rc1.- Esta programado como salida y tiene la misión de indicar el estado de la alarma, por medio de un led, si este esta encendido, la alarma esta activa y si esta apagado la alarma esta desactivada (Ver Figura 5A).

- Rc2.- Esta programado como salida y esta encargado de dar la señal de activación a un transistor que a su ves alimenta a la bobina de un rele (Ver Figura 11), el mismo que por medio de su contacto normalmente cerrado tiene la misión de cortocircuitar los platinos o a su ves el sistema infrarrojo en los vehículos más modernos con sistema de encendido electrónico, impidiendo de esta forma que la alta tensión contenida en la bobina de encendido del vehículo llegue a las bujías del motor del mismo, bloqueando de esta manera el sistema de encendido del vehículo cuando rc2 esta en nivel lógico 0, pues al no haber chispa de encendido en las bujías no abra combustión de la mezcla aire-gasolina en la cámara de compresión del motor.

- Rc6.- Esta programado como entrada y a esta ingresa la señal de un pulsante NA (normalmente abierto), colocado en la puerta del conductor y funciona de dos maneras: 1) Cuando la alarma esta activada y la puerta antes mencionada es abierta arbitrariamente el pulsante NA se cierra (Ver Figura 9) dando la señal a la unidad microcontrolada para que active la sirena a través de la salida rc0. 2) Cuando la alarma esta desactivada, el vehículo encendido y a su ves alguien abre la puerta del conductor sin previamente apagar el vehículo, el pulsante NA se abre, lo cual es detectado por la unidad microcontrolada, la misma que envía a nivel lógico 0 la salida rc3 (bloqueo de encendido) con lo cual el vehículo se apaga; es decir hace la función de sistema antiatraco.

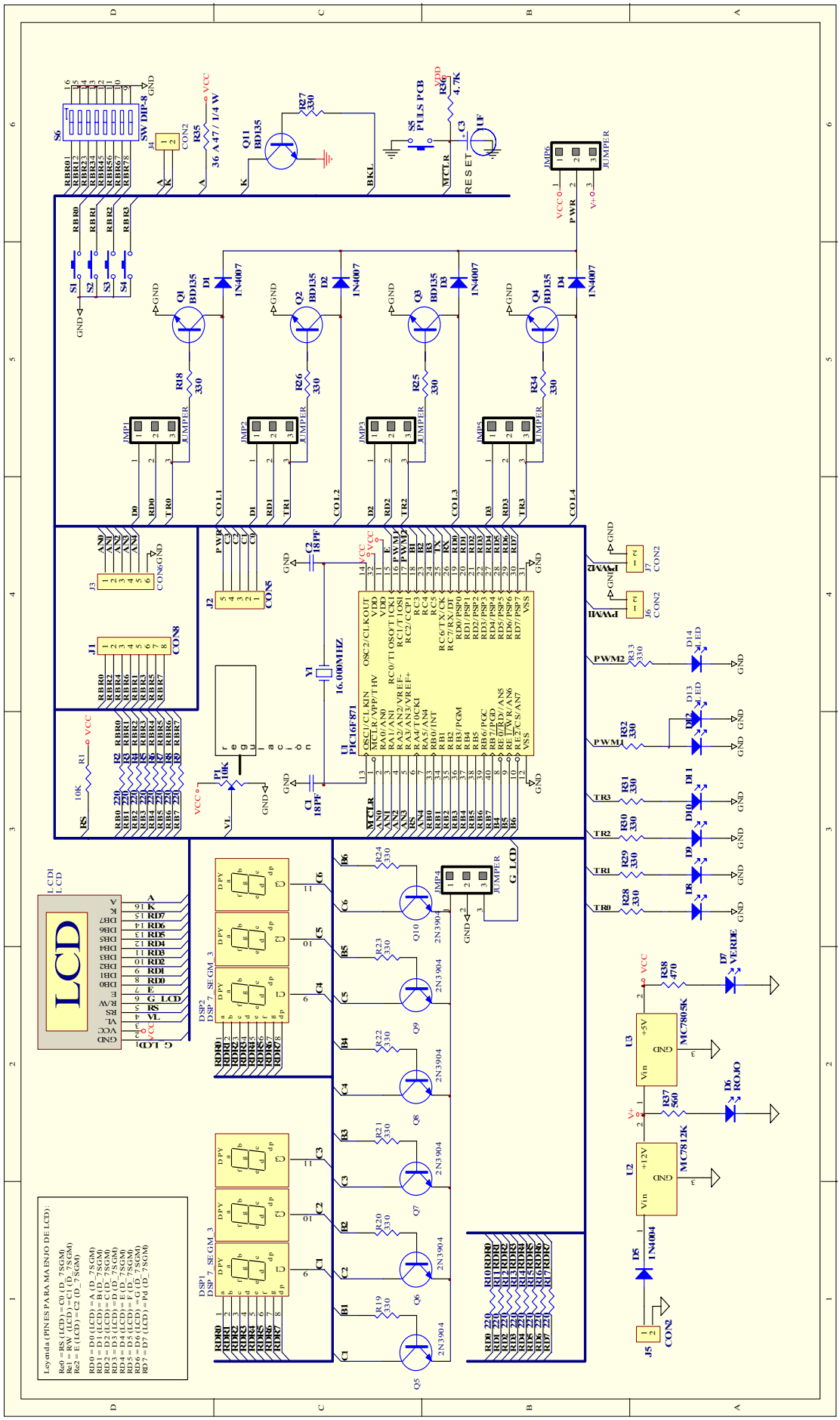
- Rc7.- Esta programado como entrada y esta conectado al switch de encendido del vehículo (Ver Figura 9), captando el instante en que dicho switch es activado (es decir el vehículo encendido), para mediante la unidad microcontrolada poner en "vigilia" el sistema antiatraco y apagar el motor si la puerta del conductor es abierta una ves el motor del vehículo este en marcha.

- Re0.- Esta programado como entrada y a esta ingresa la señal de un switch con clave mecánica NA (normalmente abierto), ubicado en un lugar accesible en la parte exterior del vehículo y sirve para activar o desactivar la alarma mediante una llave de seguridad única y no duplicable en el medio (Ver Figura 9).

- Re1.- Esta programado como entrada y a esta pueden conectarse en paralelo varios pulsantes NA (normalmente abiertos), colocados en las puertas restantes que no sea la del conductor así como en el baúl de equipaje, capot del motor o en un sin numero de accesorios del vehículo que deseemos proteger, también en esta entrada conectamos en paralelo el contacto NA del sensor de vibración o en su defecto del sensor de movimiento o los dos (Ver Figura 9); esta entrada se encarga de captar cuando uno o varios de los sensores son activados para mediante la unidad microcontrolada activar la sirena, indicando de manera sonora que la alarma fue disparada, por supuesto esto se da solamente cuando la alarma fue activada con anterioridad.

2.3.- Diseño del hardware del modulo de Alarma digital.

En la siguiente página tenemos el esquema de conexiones del hardware genérico que fue diseñado para ser utilizado durante el curso de graduación y que a su vez nos sirvió para la elaboración del presente proyecto, cabe recalcar que la conexión del LCD y de los dos displays de siete segmentos extra es opcional y que en nuestro kit de entrenamiento se decidió no incluirlos:


U1
P16F877
16.000MHZ
LCD
G_LCD
DSP1
DSP 7_SEGM 3
DSP2
DSP 7_SEGM 3
SW DIP-8
PULS PCB
RESET
BD135
1N4007
MC7805K
MC7812K
1N4004
JUMPER
Leyenda (PINES PARA MANEJO DE LCD):
Rs0 = RS (LCD) = G0 (D_7SGM)
Rs1 = RW (LCD) = G1 (D_7SGM)
Rs2 = E (LCD) = G2 (D_7SGM)
RD0 = D0 (LCD) = A (D_7SGM)
RD1 = D1 (LCD) = B (D_7SGM)
RD2 = D2 (LCD) = C (D_7SGM)
RD3 = D3 (LCD) = D (D_7SGM)
RD4 = D4 (LCD) = E (D_7SGM)
RD5 = D5 (LCD) = F (D_7SGM)
RD6 = D6 (LCD) = G (D_7SGM)
RD7 = D7 (LCD) = Pd (D_7SGM)

2.4.- Diseño del software para el control del hardware.

Antes de Comenzar el Diseño del Software primero debemos considerar ciertos puntos como: objetivos del circuito, secuencia que debe cumplir dicho programa, así como el principio de funcionamiento del software para control de teclado matricial, con ello trataremos de diseñar un programa compacto y a su vez lo más funcional posible, evitando de esta manera que los usuarios se confundan en la operación o manejo de la alarma codificada para su vehículo.

Objetivos del circuito:

Los dos principales objetivos de este circuito son:

- Primero el proteger el vehículo, dando señales audiovisuales en el momento de detectar algún problema.
- Segundo impedir que el vehículo pueda ser encendido sin previamente haber ingresado una clave de seguridad o hacer que este se apague en caso de atraco.

Secuencia de Operación:

La secuencia de operación es la siguiente:

Al conectar el módulo a la batería del vehículo, el programa grabado en el pic 16F871 inicia limpiando el bit 3 del puerto C (rc2) que es la salida se encarga de manejar el sistema de bloqueo del encendido, con esto si alguien desconecta la batería pretendiendo desactivar la alarma, no va a conseguirlo ya que al volver a conectar la batería la unidad microcontrolada volverá a bloquear el encendido y activar la sirena.

Luego para desactivar el bloqueo y sirena, se lo puede hacer marcando la clave a través del teclado matricial o con la llave de seguridad que únicamente desactivará la sirena mas no el bloqueo del encendido, la única forma de desbloquear el encendido es marcando la clave; una vez hecho esto la alarma esta desarmada; para armarla:

una vez cerradas todas las puertas con la llave de seguridad se gira el switch colocado en el exterior del vehículo hacia la derecha y se retorna hacia la izquierda a la posición original, de esta forma se enciende el led indicador de alarma armada, en este punto la alarma bloquea el encendido y permanece en alerta para; cuando es activado cualquiera de los sensores disparar la sirena, para desarmar la alarma y así poder ingresar al vehículo o a su vez apagar la sirena si esta fué disparada, se gira el switch (parte exterior del vehículo) hacia la derecha y se lo retorna a su posición original; ya en el interior del vehículo si se intenta encender el motor este no encenderá ya que el bloqueo de encendió permanece activo, para encender el motor primero debemos marcar la clave de desbloqueo del encendido a través del teclado matricial para posteriormente poder dar arranque al motor.

Una vez el motor en marcha se pueden accionar cualquiera de los sensores incluídos los de las puertas, con tal que no sea la del conductor y no afectará la unidad microcontrolada, pero si el sensor de la puerta del conductor es activado, se activa el bloqueo del encendido (sistema antiatraco) con lo cual el motor se para, para desactivar el bloqueo del encendido si se ha activado por error se marca la clave de seguridad en el teclado matricial y se enciende el motor nuevamente. Si por error se marca incorrectamente uno o varios números de la clave de desbloqueo, pulsamos la tecla asterisco (*) para volver a marcar nuevamente la clave correcta desde el inicio.

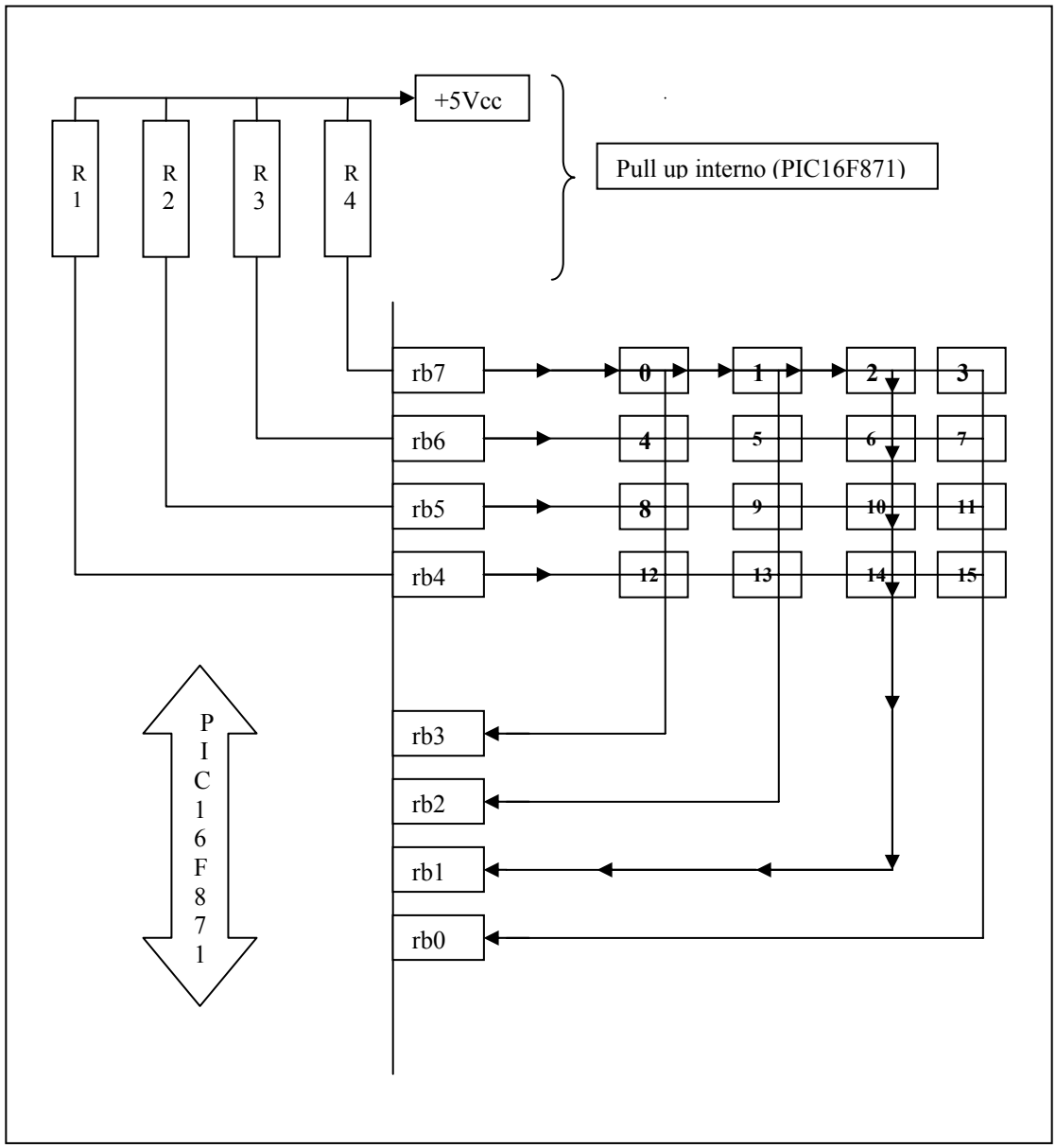
Principio de funcionamiento del Software para control de Teclado Matricial:

Como se observa en la figura 23, el telado matricial esta compuesto por 4 filas que están conectadas a las salidas (rb7, rb6, rb5 y rb4) y a través de las cuales ira saliendo un nivel lógico 0 de manera secuencial, como se muestra en la tabla adjunta, y por supuesto consta de 4 columnas conectadas a las entradas (rb3, rb2, rb1 y rb0) las mismas que serán “interrogadas” secuencialmente para determinar a cual de las columnas llega el nivel lógico cero, que sale por determinada fila.

Por ejemplo, si se pulsa la tecla de la posición 2, en primer lugar sacamos un cero por rb7 y se “interroga” a rb3 para saber si llego ahí el mencionado cero, como no es así se incrementa un registro que previamente debe ser limpiado (puesto en cero todos sus bits) y al que llamaremos “X “ como ejemplo, entonces ahora el registro X

es igual a 1 e “interrogamos” esta ves a rb2 para saber si llego ahí el cero, al no ser así se incrementa nuevamente el registro X que ahora es igual a 2 y se pasa a “interrogar” a rb1 al que si llega el cero que se estamos buscando como muestran las flechas en la figura 23, ya que la tecla de la posición 2 esta pulsada, al ser positiva la interrogación se consulta el numero contenido en el registro X que es el que determina la posición de la tecla pulsada y que para nuestro ejemplo es 2 y así de idéntica forma será la mecánica del scaneo para cada posición de tecla pulsada.

Como aclaración, debemos mencionar que el número contenido en el registro X determina únicamente la posición de la tecla pulsada y es independiente del valor o símbolo impreso por el fabricante sobre la tecla a la que corresponde esa posición.



rb7	rb6	rb5	rb4
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

Figura 23. Esquema de Teclado Matricial.

SOFTWARE PARA CONTROL DEL MODULO DE ALARMA CODIFICADA PARA VEHICULO:

```
;SOFTWARE PARA CONTROL DEL MODULO DE ALARMA CODIFICADA PARA VEHICULO.

list p=16f871
#include "P16f871.inc"

    ;Declaracion de variables
    wtemp equ    0x21
    status_temp equ 0x22
    cont1    equ   0x23  ;contadores para demora
    cont2    equ   0x24  ;contadores para demora
    cont3    equ   0x25  ;contadores para demora
    N     equ   0x26  ;factor de demora
    M     equ   0x27  ;valor al cont3
    flag  equ   0x28  ;registra si hay o no tecla
    tecla equ   0x29  ;Contiene el valor de la tecla pulsada
    org 0
    goto inicio

inicio
    ;Bloque de config de Registros
    ;definir E/s
    bsf STATUS,5 ;banco 1
    bcf OPTION_REG,7 ;Habilitación de pull-up del puerto B.
    ;Configura rb0, rb1, rb2 y rb3 como entradas y rb4, rb5, rb6 y rb7 como salidas.
    movlw b'00001111'
    movwf TRISB
    ;Configura rc0, rc1, rc2, rc3, rc4 y rc5 como salidas y; rc6 y rc7 como entradas.
    movlw b'11000000'
    movwf TRISC
    ;Configura re0 y re1 como entradas.
    bsf TRISE,0
    bsf TRISE,1
    ;lineas digitales
    movlw 7
    movwf ADCON1

    ;-------------------------------------------------------------
    ;bloque principal
    ;-------------------------------------------------------------

    bcf STATUS,5 ;cambio a banco 0
    goto alarm ;salta a alarm
on  btfsc PORTE,0
    goto on
on1 movlw b'00000011' ;activa rc0 (bocina) y rc1 (led ON-OFF) momentaneamente
    movwf PORTC
    call demoral ;llama a subrutina demoral
    movlw b'00000010' ;desactiva rc0 (bocina) y activa rc1 (led ON-OFF)
    movwf PORTC
    call demoral   ;llama a subrutina demoral
sen btfss PORTE,1  ;interroga si re1 (sensores varios N.A.) es activado
    goto alarm  ;salta a larm
    btfss PORTC,6  ;interroga si rc6 (sensor N.A. de puerta del conductor) esta activado
    goto alarm  ;salta a alarm
    btfsc PORTE,0  ;interroga si re0 (llave mecanica ON-OFF) es activada
    goto sen ;salta a sen
    movlw b'00000001' ;activa rc0 (bocina) momentaneamente
```

```
      movwf PORTC
      call demoral   ;salta a demoral
off   movlw b'00000000' ;desactiva rc0 (bocina) y rc1 (led ON-OFF)
      movwf PORTC
      call demoral ;llama a sbrutina demoral
scan1 btfss PORTE,0  ;interroga si re0 (llave mecanica ON-OFF) es activada
      goto on1 ;salta a on1
tecin call teclado   ;llama a subrutina teclado
      btfss flag,0   ;interroga si hubo tecla
      goto scan1  ;salta a scan1
tec1  movf tecla,w   ;mueveel valor contenido en tecla a W
      sublw d'0'  ;resta 0d a W (tecla 1)
      btfss STATUS,2 ;interroga si el bit 3 del registro estatus es "1" (w = 0)
      goto alert  ;salta a alert
      call demora0 ;llama a subrutina demora0
scan2 call teclado   ;llama a subrutina teclado
      btfss flag,0   ;interroga si hubo tecla
      goto scan2  ;salta a scan2
      movf tecla,w   ;mueveel valor contenido en tecla a W
      sublw d'13' ;resta 13d a W (tecla 0)
      btfss STATUS,2 ;interroga si el bit 3 del registro estatus es "1" (w = 0)
      goto alert  ;salta a alert
      call demora0   ;llama a subrutina demora0
scan3 call teclado   ;llama a subrutina teclado
      btfss flag,0   ;interroga si hubo tecla
      goto scan3  ;salta a scan3
      movf tecla,w   ;mueveel valor contenido en tecla a W
      sublw d'8'  ;resta 8d a W (tecla 7)
      btfss STATUS,2 ;interroga si el bit 3 del registro estatus es "1" (w = 0)

      goto alert  ;salta a alert
      call demora0   ;llama a subrutina demora0
scan4 call teclado   ;llama a subrutina teclado
      btfss flag,0   ;interroga si hubo tecla
      goto scan4  ;salta a scan4
      movf tecla,w   ;mueveel valor contenido en tecla a W
      sublw d'5'  ;resta 5d a W (tecla 5)
      btfss STATUS,2 ;interroga si el bit 3 del registro estatus es "1" (w = 0)
      goto alert  ;salta a alert
      call demora0   ;llama a subrutina demora0

scan5 call teclado   ;llama a subrutina teclado
      btfss flag,0   ;interroga si hubo tecla
      goto scan5  ;salta a scan5
      movf tecla,w   ;mueveel valor contenido en tecla a W
      sublw d'14' ;resta 14d a W (tecla #)
      btfss STATUS,2 ;interroga si el bit 3 del registro estatus es "1" (w = 0)
      goto alert  ;salta a alert
      call demora0   ;llama a subrutina demora0
      ;desactiva rc0 (bocina), rc1 (led ON-OFF) y activa rc2 (desbloqueo de encendido)
      movlw b'00000100'
      movwf PORTC
scan6 btfss PORTE,0  ;interroga si re0 (llave mecanica ON-OFF) es activada
      goto on1 ;salta a on1
      btfss PORTC,7  ;interroga si rc7 (switch de encendido) esta activado
      goto scan6  ;salta a scan6
      btfsc PORTC,6  ;interroga si rc6 (sensor N.A. de puerta del conductor) esta activado
      goto scan6  ;salta a scan6
      movlw b'00000000' ;desactiva rc2 (desbloqueo de encendido)
```

```
      goto tecin  ;salta a tecin

alarm movlw b'00000011' ;activa rc0 (bocina) y rc1 (led ON-OFF)
      movwf PORTC
notec btfss PORTE,0  ;interroga si rc0 es activado (Llave mecánica)
      goto off ;salta a off
      call teclado   ;llama a subrutina teclado
      btfss flag,0   ;interroga si hubo tecla
      goto notec  ;salta a notec
      goto tec1   ;salta a tec1

alert btfss PORTE,0  ;interroga si rc0 es activado (Llave mecánica)
      goto on1 ;salta a on1
      call teclado   ;llama a subrutina teclado
      btfss flag,0   ;interroga si hubo tecla
      goto alert  ;salta alert
      movf tecla,w   ;mueveel valor contenido en tecla a W
      sublw d'12' ;resta 12d a W (tecla *)
      btfss STATUS,2 ;interroga si el bit 3 del registro estatus es "1" (w = 0)
      goto alert  ;salta a alert
      goto scan1  ;salta a scan1

      ;-------------------------------------------------------------------
      ;Bloque de Subrutinas
      ;-------------------------------------------------------------------

      ;Subrutinas de demoras
      ;###################################################################



demora0
      movlw d'128' ;carga 128d en la variable N
      movwf N
      movlw d'10' ;carga 10d en la variable M
      movwf M

      movf N,w ;mueve el contenido de la variable N a W
      movwf cont1 ;mueve el contenido de W (N) a la variable cont1
      movwf cont2 ;mueve el contenido de W (N) a la variable cont2
      movf M,w ;mueve el contenido de la variable M a W
      movwf cont3 ;mueve el contenido de W (M) a la variable cont3
      ;se decrementa la variable cont1 y si el resultado es cero salta a la siguiente instrucción
loop0    decfsz cont1
      goto loop0  ;salta a loop0
      movf N,w ;mueve el contenido de la variable N a W
      movwf cont1 ;mueve el contenido de W (N) a la variable cont1
      ;se decrementa la variable cont2 y si el resultado es cero salta a la siguiente instrucción
      decfsz cont2
      goto loop0  ;salta a loop0
      movf N,w ;mueve el contenido de la variable N a W
      movwf cont2 ;mueve el contenido de W (N) a la variable cont2
      ;se decrementa la variable cont3 y si el resultado es cero salta a la siguiente instrucción
      decfsz cont3
      goto loop0  ;salta a loop0
      ;fin del proceso, inicia contadores
      return


demora1  movlw d'128'   ;carga 128d en la variable N
```

```
      movwf N
      movlw d'60' ;carga 60d en la variable M(variacion de demora)
      movwf M

      movf N,w ;mueve el contenido de la variable N a W
      movwf cont1 ;mueve el contenido de W (N) a la variable cont1
      movwf cont2 ;mueve el contenido de W (N) a la variable cont2
      movf M,w ;mueve el contenido de la variable M a W
      movwf cont3 ;mueve el contenido de W (M) a la variable cont3
      ;se decrementa la variable cont1 y si el resultado es cero salta a la siguiente instrucción
loop1    decfsz cont1
      goto loop1  ;salta a loop1
      movf N,w ;mueve el contenido de la variable N a W
      movwf cont1 ;mueve el contenido de W (N) a la variable cont1
      ;se decrementa la variable cont2 y si el resultado es cero salta a la siguiente instrucción
      decfsz cont2
      goto loop1  ;salta a loop1
      movf N,w ;mueve el contenido de la variable N a W
      movwf cont2 ;mueve el contenido de W (N) a la variable cont2
      ;se decrementa la variable cont3 y si el resultado es cero salta a la siguiente instrucción
      decfsz cont3
      goto loop1  ;salta a loop1
      ;fin del proceso, inicia contadores
      return

      ;subrutina de atención a teclado matricial**
      ;*****************************************************************
teclado     clrf tecla ;limpia la variable tecla
      bcf flag,0     ;indica "no tecla"

      movlw b'01111111' ;saca un "0" por rb7
      movwf  6
nextcol     btfss 6,3   ;interroga si llego el "0" a rb3
      goto sitecla   ;salta a sitecla
      incf tecla  ;incrementa la variable o registro tecla
      btfss 6,2   ;interroga si llego el "0" a rb2
      goto sitecla   ;salta a sitecla
      incf tecla  ;incrementa la variable o registro tecla
      btfss 6,1   ;interroga si llego el "0" a rb1
      goto sitecla   ;salta a sitecla
      incf tecla  ;incrementa la variable o registro tecla
      btfss 6,0   ;interroga si llego el "0" a rb0
      goto sitecla   ;salta a sitecla
      incf tecla  ;incrementa la variable o registro tecla
      movf tecla,w   ;mueve el contenido de tecla a W (registro de trabajo)
      sublw 0x10  ;  ;resta 16d a W
      btfss 3,2   ;interroga si el bit 3 del registro estatus es "1" (w = 0)
      goto menor  ;salta a menor

      ;igual a 17d   (no hubo tecla), se indica con el bit 0 de flag en 1
      bcf flag,0  ;indica "no tecla"
      return

menor    ;seguir escaneando, todavía no es la última tecla

      bsf 3,0     ;carry = 1, preparo para rotar
      rrf 6,1     ;rota puerto b a la derecha
      goto nextcol


sitecla     bsf flag,0     ;hubo tecla, y su valor esta en "tecla"
      return


        end
```

2.5.- Pruebas y comprobaciones de hardware y software.

Una vez ensamblado completamente, el sistema en lo que se refiere a hardware y haber detectado y corregido posibles errores de montaje se debe proceder a probarlo asegurándose que las conexiones de la tarjeta principal con los componentes externos sean correctas; para ello previamente procedemos a compilar el programa diseñado valiéndonos del Programa MPLAB IDE que provee de forma gratuita la compañía Microchip fabricante del PIC utilizado en este proyecto, con ello obtenemos un archivo de nuestro programa en formato hexadecimal necesario para ser grabado en la memoria del PIC16F871 mediante un grabador de microcontroladores que puede ser el denominado PICSTAR. Corrido el software en nuestro kit procedemos una vez más a detectar, corregir y depurar posibles fallas pero esta vez del conjunto completo, es decir tanto de software como hardware, hasta conseguir un conjunto "software-Hardware" que funcione de manera optima.

A continuación se muestra una tabla de los niveles lógicos obtenidos durante las pruebas y comprobaciones de la operación de nuestro conjunto final "Software-Hardware":

ACCESORIOS							**TECLADO MATRICIAL**							
ENTRADAS				SALIDAS			ENTRADAS				SALIDAS			
re0	re1	rc6	rc7	rc0	rc1	rc2	rb0	rb1	rb2	rb3	rb4	rb5	rb6	rb7
■					■									
	■			■	■									
		■		■	■									
■	■	■												
										■				1
									■		**0**			
										■		7		
									■				**5**	
						■		■			#			
	■		■			■								
		■	■											

■	Nivel lógico "1" u activo
	Nivel lógico "0" o inactivo

re0: Switch ON - OFF con clave de seguridad mecánica
re1: Sensores N.A. varios, incluídos de puertas (Menos de puerta del conductor)
rc6: Sensor N.A. de puerta del conductor
rc7: Switch de encendido del vehículo
rc0: Bocina de alarma
rc1: Led indicador On - OFF
rc2: Bloqueo de Encendido
rb0 a rb7: Teclado Matricial 4x4
1, 0, 7, 5, # : Números impresos de fábrica sobre las teclas; es decir la clave de desbloqueo

Conclusiones:

El conjunto Hardware-Software que hemos obtenido al final cumple con todas las secuencias de operación planteadas al inicio para así brindar seguridad al vehículo, además de un sistema anti-atraco para mayor tranquilidad del usuario. Su operación es acorde a otros sistemas estándar de alarma existentes en el mercado, por supuesto con ciertas mejoras o cambios como la inclusión de un teclado matricial que a la postre brindan mayor seguridad que otros sistemas disponibles en nuestro medio.

CONCLUSIONES.

- El PIC16F871 es una de las mejores opciones cuando se pretende diseñar un sistema microcontrolado ya que no requiere un excesivo número de componentes externos y tiene un precio muy competitivo, además mediante software el sistema se puede variar o extender sin problemas consiguiendo con ello un interesante, eficaz y versátil sistema electrónico de Alarma para Vehículo.

- Al momento de instalar en el vehículo el módulo o tarjeta principal del sistema, se debe tratar de ubicar la misma en un sitio fuera del alcance de otras personas y donde el agua no pueda dañar el equipo.

- En cuanto al cable que se lleva hasta los sensores u otros componentes externos, este se debe pasar lo más oculto posible tanto por seguridad como por estética.

- La bocina es un dispositivo medular que se recomienda ubicar en un sitio muy seguro fuera de la vista o alcance de personas ajenas que pretendan manipular la misma con el fin de anularla o silenciarla.

- Si se utiliza algún sensor infrarrojo u otros que tienen calibración de sensibilidad, deben ajustarse al valor adecuado según como recomiende el fabricante, estos ajustes y recomendaciones vienen junto con cada dispositivo en una pequeño manual de montaje adjunto.

- Deben fijarse todos y cada uno de los sensores firmemente para evitar falsas alarmas que causen molestias al usuario.

- Se debe aplicar soldadura a todas las uniones y usar cinta aislante para evitar posibles cortos.

- Las conexiones a la batería hay que hacerlas en forma impecable y en lo posible utilizar terminales ya que una mala masa o un corte de alimentación pueden hacer que el equipo falle.

- Este dispositivo es aplicable prácticamente a todo vehículo con motor de combustión interna, ya sea éste con sistema de alimentación de combustible a carburador o a inyección y por supuesto que disponga de carrocería, es decir no es aplicable a motocicletas u otros vehículos no carrozados.

- El dispositivo expuesto a lo largo del presente trabajo además de brindar seguridad al vehículo se lo puede proyectar a futuro para controlar los diferentes componentes eléctricos o mecánicos del vehículo.

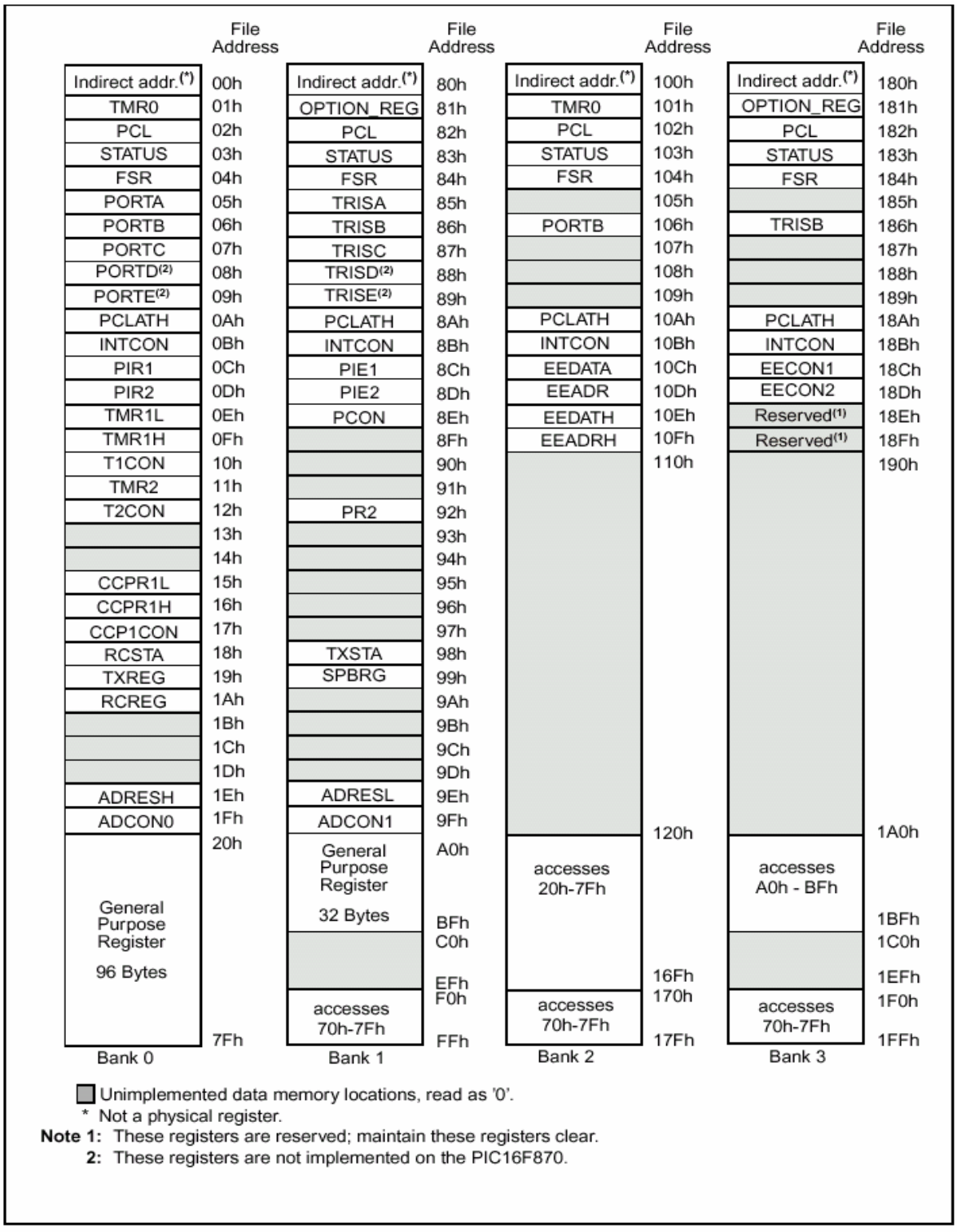
Bibliografía:

- ANGULO José M. Enciclopedia de Electrónica Moderna, Tomo 6. Madrid-España. Editorial PARANINFO, S.A. 1990. 272 páginas. Segunda edición.
- BELOVE Charles. Enciclopedia de la Electrónica, Ingeniería Técnica. Barcelona-España. Grupo Editorial Océano. [199-].
- CEKIT. Curso práctico de Electrónica Digital y circuitos integrados.
- CEKIT. Electrónica & Computadores, Ejemplar #23. Cartago-Colombia. Editora Géminis.Ltda. 2001. 80 páginas. Primera edición.
- COUGHLIN Robert F, DRISCOLL Frederick F. Amplificadores Operacionales y Circuitos Integrados Lineales. México. Prentice-Hall. Hispanoamericana. S.A. 1996. 538 páginas.
- DE CASTRO VICENTE Miguel. La Electrónica en el Automóvil. Barcelona-España. Ediciones CEAC, S.A. 1988. 364 páginas. Segunda Edición.
- GENERAL MOTORS. Manual del Conductor (Chevrolet Corsa). Santafé de Bogotá D.C.-Colombia. 2003. 120 páginas.
- PÉREZ Leonel Msc. CD del Curso de Graduación a distancia para Tecnólogos. Cuenca-Ecuador. 2004.
- SANZ GONZALEZ Ángel. Tecnología de la Automoción 2.1, Mecánica y Electricidad del Automóvil. Barcelona-España. Editorial Bruño. [s.a.] 283 páginas. Primera edición.
- MICROCHIP. PICmicro Mid-Range Mcu Family. USA. 1997. 688 páginas

ANEXOS:

ANEXO 1:

Mapa de la Memoria de Datos



ANEXO 2:

Set de Instrucciones PIC16F871

TABLE 12-2: PIC16CXXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

Note: Additional information on the mid-range instruction set is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).