ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

"ZIGBEE PARA LA IMPLEMENTACIÓN DE UNA SALA DE CONFERENCIAS INTELIGENTE EN LA EMPRESA ELÉCTRICA RIOBAMBA S.A."

TESIS DE GRADO

PREVIA OBTENCIÓN DEL TÍTULO DE

INGENIERO EN ELECTRÓNICA Y COMPUTACIÓN.

PRESENTADO POR:

MERCEDES CRISTINA NARANJO ORDÓÑEZ

DIEGO MARCELO CHILIQUINGA CHILIQUINGA

RIOBAMBA - ECUADOR

2010

Agradezco a Dios por otorgarme el preciado don de la vida y permitirme día a día luchar por conseguir mis objetivos, a mi madre y a mis hermanas, por su esfuerzo, apoyo, amor y paciencia, pero sobre todo les agradezco por luchar junto a mi en todo momento.

CRISTINA

Agradezco a Dios por haberme guiado por el camino de la felicidad hasta ahora, a mis padres y a mi hermana por siempre haberme dado su fuerza y apoyo incondicional que me han ayudado y llevado hasta donde estoy ahora.

DIEGO

Un agradecimiento al Ingeniero Hugo Moreno por el apoyo presentado en el desarrollo de este Proyecto. Y un especial agradecimiento a la Empresa Eléctrica Riobamba S.A. en personas del Ing. Joe Ruales Gerente de la EERSA Y Al Ing. Luis Almeida, por su apoyo desinteresado.

CRISTINA Y DIEGO

Dedico este proyecto de tesis a Dios, a mi madre y a mis hermanas, pilares fundamentales en mi vida. Sin ellas, jamás hubiese podido conseguir mis metas y sueños. A mi madre por su tenacidad y lucha insaciable ha hecho de ella el gran ejemplo a seguir y destacar, no solo para mí, sino también para mis hermanitas.

CRISTINA

Dedico este proyecto de tesis a Dios, a mis padres y a mi hermana. A Dios porque ha estado conmigo a cada paso que doy, cuidándome y dándome fortaleza para continuar, a mis padres y hermana, quienes a lo largo de mi vida han velado por mi bienestar y educación siendo mi apoyo en todo momento. Depositando su entera confianza en cada reto que se me presentaba sin dudar ni un solo momento en mi inteligencia y capacidad. Es por ellos que soy lo que soy ahora. Los amo con mi vida.

DIEGO

NOMBRE	FIRMA	FECHA
Dr. Ms.c. Romeo Rodríguez **DECANO DE LA FACULTAD** **DE INFORMÁTICA Y** **ELECTRÓNICA**	**……**	**……**
Ing. Paúl Romero **DIRECTOR DE LA** **ESCUELA DE INGENIERÍA** **ELECTRÓNICA.**	**……**	**……**
Ing. Hugo Moreno **DIRECTOR DE TESIS**	**……**	**……**
Ing. William Calvopiña **MIEMBRO DEL TRIBUNAL**	**……**	**……**
Tlgo. Carlos Rodríguez **DIRECTOR DPTO.** **DOCUMENTACION**	**……**	**……**
NOTA DE LA TESIS	**……**	

“Nosotros, **MERCEDES CRISTINA NARANJO ORDÓÑEZ** y **DIEGO MARCELO CHILIQUINGA CHILIQUINGA**, somos los responsables de las ideas, doctrinas y resultados expuestos en esta Tesis de Grado, y el patrimonio intelectual de la misma pertenecen a la Escuela Superior Politécnica de Chimborazo”

Mercedes Cristina Naranjo Ordóñez	Diego Marcelo Chiliquinga Chiliquinga

ÍNDICE DE ABREVIATURAS

AIB	APS Information Base.
ALU	Unidad Aritmético Lógica.
APS	Sub capa de aplicación.
CAD	Conversor Analógico a digital.
CAN	Controlador de red de Área.
CDA	Conversor Diagital a analógico.
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DOMUS	Casa en Latín.
DSSS	Direct Sequence Spread Spectrum.
EEPROM	Memoria de solo lectura programable y borrable eléctricamente.
EPROM	Memoria programable y borrable de sólo lectura.
FFD	Dispositivos de función completa.
FLASH	Memoria no volátil de bajo consumo.
FSK	Modulación de frecuencia.
GOF	General Operations Frame.
GLCD	Pantalla de cristal Liquida Grafica.
GPR	Registros de propósito General.
HS	Cristal de alta velocidad.
HSEROUT	Salida serial de hardware asíncrono.
IEEE	Instituto de ingenieros eléctricos y electrónicos.
LP	Cristal para baja potencia y bajo consumo de corriente.
LR-WPAN	Redes Inalámbricas de Área Personal de Baja Velocidad.
MAC	Chequeo de Autenticidad de Mensaje
MCLR	Master Clear (Reset).
NWK	Capa de red.
NWKID	Identificador de red.
OSC1/CLKIN	Entrada del oscilador (cristal). Entrada de oscilador externo.
PAN	Red de área personal.
PDU	Unidad de protocolo de datos.
PHY	Capa Física.
PIC	Peripherical Interface Controller.
POP	Power-on Reset.
PWM	Modulación por ancho de pulso.
RAM	Memoria de acceso aleatorio.

RISC	Computadores de Juego de Instrucciones Reducido.
RF	Radio Frecuencia.
RFD	Dispositivos de función reducida.
SI	Sistema Inteligente.
SRAM	RAM estática.
TICA	Gestión de Medios Informáticos.
UART	Adaptador de comunicación serie asíncrona.
UCP	Unidad Central de Proceso.
USART	Transmisor/Receptor Asíncrono Universal.
USB	Bus serial Universal.
WDT	Perro Guardián instrucción.
WPAN	Redes De Área Personal.
XBEE	Módulos de tecnología ZigBee.

ÍNDICE GENERAL

CAPÍTULO I: GENERALIDADES

CAPÍTULO II: FUNDAMENTO TEÓRICO

CAPÍTULO III: ZIGBEE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

INTRODUCCIÓN

Gracias a la incesante evolución de la tecnología, se puede contar hoy en día con espacios de uso cotidiano que nos puedan ofrecer mejores condiciones de vida, tanto para edificios nuevos como para construcciones existentes. Este desarrollo se debe esencialmente al progreso en cuanto a la electrónica y las redes internas de comunicación, así como la integración de servicios inalámbricos en beneficio de la automatización.

Respecto a la automatización, ésta comenzó a darse hace veinte años en industrias innovadoras, tales como la aeronáutica y la automotriz, donde comenzaron a utilizar diversas automatizaciones con cierto grado de integración. Después siguieron los edificios comerciales y administrativos para finalmente ser las construcciones educacionales y viviendas los sitios en donde se han utilizado criterios de integración técnico-espacial.

Una vez que iniciaron los primeros ensayos con aparatos electrodomésticos y dispositivos automáticos para los hogares; los franceses definieron como domótica a esta nueva corriente arquitectónica, la cual define una vivienda capaz de integrar todos los sistemas automáticos de gestión de la energía, protección patrimonial, confort, comunicaciones y demás.

Respecto a esto, la domótica abre nuevas posibilidades en cuanto a la integración del hogar, pues se convierte en una herramienta mediante la cual, los habitantes de un hogar pueden controlar y administrar de manera eficiente su espacio común.

Por ejemplo, en un día normal una persona que vive en una oficina inteligente no deberá preocuparse por en media conferencia levantarse para adecuar el lugar. Además, el sistema se encargará automáticamente de abrir y cerrar las persianas, las ventanas, regular la luz, o a su vez encenderla o apagarla.

Debido al impacto de la Domótica dentro de la integración y automatización de construcciones, se crearon tecnologías de comunicaciones tanto cableadas como inalámbricas con el fin de facilitar la interacción de dispositivos dentro de una vivienda. Para relacionarse remotamente con todos estos dispositivos, necesitamos trabajar con un solo estándar que permita tenerlos a todos bajo una misma red.

Uno de los protocolos inalámbricos más prometedores y de mayor desarrollo en los últimos años es el estándar IEEE 802.15.4 o también denominado “Zigbee”, el cual ha evolucionado precisamente como una aplicación de la automatización de hogares.

ZigBee es un sistema ideal para redes domóticas, específicamente diseñado para reemplazar la proliferación de sensores/actuadores individuales. Esta tecnología fue creada para cubrir la necesidad del mercado de un sistema de bajo costo a través de un estándar para redes Wireless de pequeños paquetes de información, bajo consumo, seguro y fiable.

Para llevar a cabo este sistema, un grupo de trabajo formado por varias industrias está desarrollando el mismo. Esta alianza de empresas está trabajando conjuntamente con el IEEE para asegurar una integración, completa y operativa. La alianza ZigBee también sirve para probar los dispositivos que se crean con esta tecnología.

CAPÍTULO I

GENERALIDADES

1.1. ANTECEDENTES

La domótica se puede definir como la integración de la tecnología en el diseño inteligente de un recinto. Esto es, un conjunto de sistemas capaces de automatizar una vivienda, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo control goza de cierta ubicuidad, desde dentro y fuera del hogar. En el ámbito del nivel de confort se pueden considerar actividades como: la automatización del encendido, apagado y regulación de las luces de la vivienda y de otros sistemas o equipos.

ZigBee es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radios digitales de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal wireless personal área network, WPAN. Su objetivo son las aplicaciones que requieren

comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías.

IEEE 802.15.4 es un estándar que define el nivel físico y el control de acceso al medio de redes inalámbricas de área personal con tasas bajas de transmisión de datos low-rate wireless personal area network, LR-WPAN. La actual revisión del estándar se aprobó en 2006.

Los protocolos ZigBee están definidos para su uso en aplicaciones embebidas con requerimientos muy bajos de transmisión de datos y consumo energético. Se pretende su uso en aplicaciones de propósito general con características auto organizativas y bajo costo redes en malla. Puede utilizarse para realizar control industrial, albergar sensores empotrados, recolectar datos médicos, ejercer labores de detección de humo o intrusos o domótica.

Estos sistemas generalmente están integrados por medio de redes de comunicación de diferentes topologías, permite el control de 255 nodos activos simultáneamente, entre las topologías más utilizadas la topología en estrella la cual alcanza un total máximo de 65535 nodos por medio de dispositivos ruteadores, tanto en el interior como en el exterior del lugar donde esta implementada inalámbricamente, controladas igualmente tanto desde dentro como fuera de ésta, además uno de las características principales de los módulos Xbee es su largo alcance que varia desde los 70m hasta los 1.6Km de acuerdo a la versión y el modelo del módulo, donde los módulos de aplicación básica rodean los 80m.

Para interactuar remotamente con todos estos dispositivos, se necesita trabajar con un solo estándar para poder tenerlos todos bajo una misma plataforma. Uno de los

protocolos más prometedores en el área de la domótica es ZigBee, basado en el estándar IEEE 802.15.4.

En la Tabla I.I., se puede observar la comparación entre las principales Tecnologías Wireless, destacando las ventajas de Zigbee para la aplicación en domótica.

Tabla I.I. Comparación entre las principales tecnologías Wireless

Estándar	**Ancho de Banda**	**Consumo de Corriente**	**Ventajas**	**Aplicaciones**
Wi-Fi	Hasta 54Mbps	400ma transmitiendo, 20ma en reposo	Gran ancho de banda	Navegar por Internet, redes de ordenadores, transferencia de ficheros
Bluetooth	1 Mbps	40ma transmitiendo, 0.2ma en reposo	Interoperabilidad, sustituto del cable	Wireless USB, móviles, informática casera
ZigBee	250 Kbps	30ma transmitiendo, 10ua en reposo	Batería de larga duración, bajo costo	Control remoto, productos dependientes de la batería, sensores, juguetería

Los sistemas de control domótico permiten dedicar ese bien tan preciado, el tiempo, para el ocio y disfrute. La Domótica promete mayor seguridad, confort y vanguardismo en nuestras viviendas, acompañado con la nueva tecnología inalámbrica como lo es Zigbee.

1.2. JUSTIFICACIÓN

Sabiendo que actualmente en nuestro país no se utiliza esta tecnología en control domótico por ser aún desconocida, pero en países más desarrollados ya está siendo utilizada y brindando grandes beneficios como los mencionamos anteriormente.

Es por esto que debido al crecimiento de las aplicaciones con diferentes tecnologías en la actualidad, tales como wifi, infrarrojo, bluetooth, etc. Este proyecto se desarrolló con la finalidad de prestar comodidad a las personas dentro de su entorno de trabajo, creando una interfaz capaz de controlar diferentes dispositivos mediante un control remoto que pueda acceder desde cualquier lugar dentro de un rango aproximado de 70 metros cumpliendo con los estándares de módulos Xbee para este estudio el 802.15.4 de la IEEE.

Es por lo que este proyecto se basó en la automatización de:

- Alumbrado Eléctrico dentro de la sala de conferencias.
    - Dos Variadores de luminosidad.
    - Un fijo.
- Control de las dos persianas de la sala.
- Control de la abertura y cerradura de las ventanas.

Este control se llevó ha cabo con la utilización de un dispositivo maestro el que será un control remoto el cual posee una pantalla táctil para acceder a las distintas funciones.

El proceso de automatización en los diferentes dispositivos controlados se los realizó reemplazando su entorno manual o mecánico por un entorno electrónico e inteligente, es así que los interruptores mecánicos fueron reemplazados por módulos receptores Xbee que realicen una misma función de encendido y apagado pero desde cualquier lugar que se encuentre dentro del rango de cobertura.

1.3. OBJETIVOS

1.3.1 Objetivos Generales

- Implementar un proyecto de Zigbee para la implementación de una sala de conferencias inteligente en la empresa eléctrica Riobamba S.A.

1.3.2 Objetivos Específicos

- Estudiar la tecnología ZigBee para la implementación del proyecto de control de la sala inteligente.
- Diseñar el proyecto de control de la sala inteligente con la utilización de dispositivos que mas se adapten a nuestros requerimientos y tecnología estudiada.
- Implementar el proyecto con tecnología ZigBee en la sala de conferencias de gerencia.
- Realizar pruebas de funcionamiento y puesta en marcha de nuestro sistema.

CAPÍTULO II

FUNDAMENTO TEÓRICO.

2.1. DOMÓTICA

La robótica es una ciencia o rama de la tecnología, que estudia el diseño y construcción de máquinas capaces de desempeñar tareas realizadas por el ser humano o que requieren del uso de inteligencia. Básicamente, la robótica se ocupa de todo lo concerniente a los robots, lo cual incluye el control de motores, mecanismos automáticos neumáticos, sensores, sistemas de cómputos, etc. Las ciencias y tecnologías de las que deriva podrían ser: el álgebra, los autómatas programables, las máquinas de estados, la mecánica o la informática.

Antes de realizar una definición sobre lo que es domótica, conviene resaltar el concepto de lo que es un edificio o vivienda digital: “El Hogar Digital es una vivienda que a través de equipos, sistemas, y la integración tecnológica entre ellos, ofrece a sus habitantes funciones y servicios que facilitan la gestión y el mantenimiento del hogar, aumentan la seguridad; incrementan el confort; mejoran las telecomunicaciones; ahorran energía, costes y tiempo, y ofrecen nuevas formas de entretenimiento, ocio y otros servicios dentro de la misma y su entorno .

El término Domótica proviene de la unión de las palabras domus “que significa casa en latín” y del sufijo tica, el cual se refiere a la “gestión de medios informáticos”.

Se define a la Domótica como “el conjunto de sistemas capaces de automatizar una vivienda, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo control goza de cierta ubicuidad, desde dentro y fuera del hogar. Se podría definir como la integración de la tecnología en el diseño inteligente de un recinto”, es decir la domótica al equipamiento de cualquier vivienda o edificio con una sencilla tecnología que permita gestionar de forma energéticamente eficiente, segura y confortable para el usuario, los distintos aparatos e instalaciones domésticas tradicionales que conforman un hogar.

La domótica busca la integración de nuevas tecnologías de comunicación, automatización y control al espacio arquitectónico, de tal forma que se consiga formar un todo coherente y medible, donde cada tarea asignada al sistema, mejore calidad de vida de las personas.

Al ser un campo tecnológico tan amplio y variado, los beneficios que ofrece la domótica se incrementan a diario, por lo que resulta conveniente agruparlos según su campo de acción, para lo cual se pueden citar algunos de ellos:

- El control completo del consumo de energía; en este caso se refiere a un “consumo inteligente” de los recursos eléctricos, ya que el usuario puede disponer únicamente de los aparatos que necesite, sin desperdiciar corriente en aquellos dispositivos que se encuentren en reposo. Este aspecto supone un ahorro económico al momento de cancelar la planilla eléctrica.

- La potenciación y enriquecimiento de la propia red de comunicaciones, pues ésta puede seguir creciendo de forma permanente en todo sentido, ya que al ser una tecnología escalable, es posible instalar o incluir aparatos y dispositivos del hogar de acuerdo a las características propias de la red.
- Al tener control sobre los accesos hacia el hogar e implementar alarmas eficaces en cuanto a penetración no permitida de personas, es factible obtener la más completa y eficiente seguridad personal y patrimonial, en donde cada habitante encuentre protegido por el sistema.
- Al hablar de implementar una red externa de comunicaciones propia, se puede transmitir y recibir información con cualquier central de ayuda que haya sido creada para asistir a los dueños del hogar inteligente.
- Se consigue un nivel de confort muy superior, con lo cual la calidad de vida aumenta considerablemente.

2.1.1. Sistema Domótico

Un Sistema Domótico es un Sistema Inteligente (S.I.) que consta de: una red de comunicación configurada de tal manera que admita la interconexión de una serie de equipos que permitan obtener información acerca del entorno arquitectónico (es decir el edificio) con el fin de, compilar y procesar dicha información, realizar tareas sobre dicho entorno. Simplificando este concepto, podemos decir que un Sistema Domótico se encarga de interconectar todos los sistemas automáticos y tomar las decisiones respectivas.

Este tipo de sistemas buscan la integración completa de aquellos servicios del hogar que forman parte del mismo en un solo conjunto; permitiendo de esta manera el acceso desde diversos dispositivos, tales como:

- Una Computadora Personal
- Un teclado alfanumérico
- Una pantalla Touch Screen
- Un teléfono celular
- Internet

2.1.2. Características Generales de un Sistema Domótico

- **Integración:**

  Todo el sistema funciona bajo el control de un Ordenador Personal, de tal forma que los usuarios no deban estar pendientes de los diversos equipos, pues éstos son completamente autónomos, con su propia programación; además se incorporan indicadores situados en diferentes lugares que notifiquen dificultades de interconexión entre equipos de distintos fabricantes, fallas en la alimentación de algún equipo, pérdida de información en cualquier punto de la red, etc.

- **Interrelación:**

  Una de las principales características que debe ofrecer un sistema domótico es la capacidad para relacionar diferentes elementos y obtener una gran versatilidad y variedad en la toma de decisiones. Así, por ejemplo, es sencillo relacionar el funcionamiento del Aire Acondicionado con el de otros electrodomésticos, o con la apertura de ventanas, etc.

- **Facilidad de uso:**

  Con una sola mirada a la pantalla del control remoto, el usuario estará completamente informado del estado de su vivienda. Y si desea modificar algo, solo necesitará pulsar una pantalla.

- **Control remoto:**

  Las mismas posibilidades de supervisión y control disponibles localmente.

2.2. MICROCONTROLADOR

Desde la invención del circuito integrado, el desarrollo constante de la electrónica digital ha dado lugar a dispositivos cada vez más complejos. Entre ellos los microprocesadores y los microcontroladores.

Los microcontroladores están presentes en nuestro trabajo, en nuestra casa y en nuestra vida, en general. Se pueden encontrar controlando el funcionamiento de los ratones y teclados de los computadores, en los teléfonos, en los hornos microondas y los televisores de nuestro hogar.

Una de las principales ventajas de los microcontroladores y microprocesadores al ser sistemas programables es su flexibilidad, lo que permite actualizar el funcionamiento de un sistema tan sólo mediante el cambio del programa sin tener que volver a diseñar el hardware. Esta flexibilidad es muy importante, al permitir que los productos se actualicen con facilidad y economía.

Recibe el nombre de controlador el dispositivo que se emplea para el gobierno de uno o varios procesos. Aunque el concepto de controlador ha permanecido invariable a través del tiempo, su implementación física ha variado frecuentemente.
En la actualidad, todos los elementos del controlador se han podido incluir en un chip, el cual recibe el nombre de microcontrolador. Realmente consiste en un sencillo pero completo computador contenido en el corazón (chip) de un circuito integrado.

Un microcontrolador es un circuito integrado de alta escala de integración que incorpora la mayor parte de los elementos que configuran un controlador.

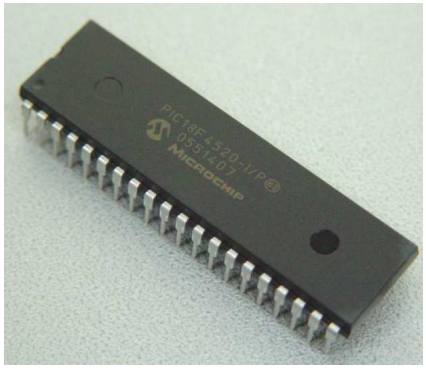


Figura II.1. Apariencia física de un Microcontrolador.

2.2.1. Diferencia entre microprocesador y microcontrolador.

El microprocesador es un circuito integrado que contiene la Unidad Central de Proceso (UCP), también llamada procesador, de un computador. La UCP está formada por la Unidad de Control, que interpreta las instrucciones, y el Camino de Datos, que las ejecuta.

Las patitas de un microprocesador sacan al exterior las líneas de sus buses de direcciones, datos y control, para permitir conectarle con la Memoria y los Módulos de E/S y configurar un computador implementado por varios circuitos integrados. Se dice que un microprocesador es un sistema abierto porque su configuración es variable de acuerdo con la aplicación a la que se destine (ver Figura II.1.).

Un microprocesador es un sistema abierto con el que puede construirse un computador con las características que se desee, acoplándole los módulos necesarios.

Un microcontrolador es un sistema cerrado que contiene un computador completo y de prestaciones limitadas que no se pueden modificar. La disponibilidad de los buses en el exterior permite que se configure a la medida de la aplicación.

Todas las partes del computador están contenidas en su interior y sólo salen al exterior las líneas que gobiernan los periféricos.

Si sólo se dispusiese de un modelo de microcontrolador, éste debería tener muy potenciados todos sus recursos para poderse adaptar a las exigencias de las diferentes aplicaciones. Esta potenciación supondría en muchos casos un despilfarro.

En la práctica cada fabricante de microcontroladores oferta un elevado número de modelos diferentes, desde los más sencillos hasta los más poderosos.

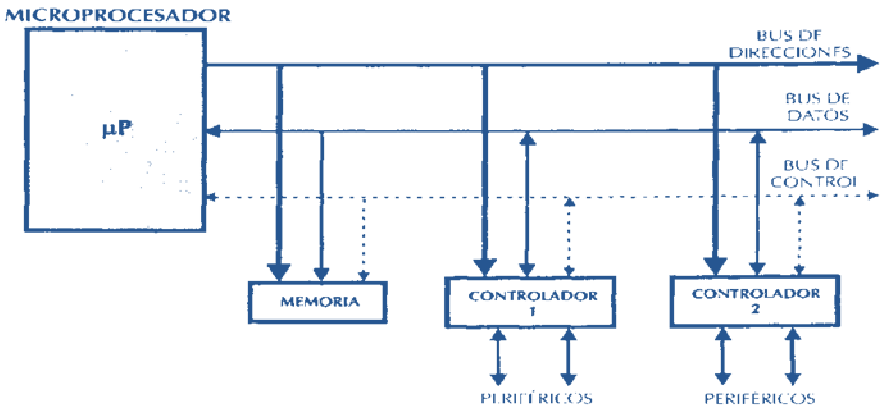


Figura II.2. Estructura de un sistema abierto basado en un microprocesador

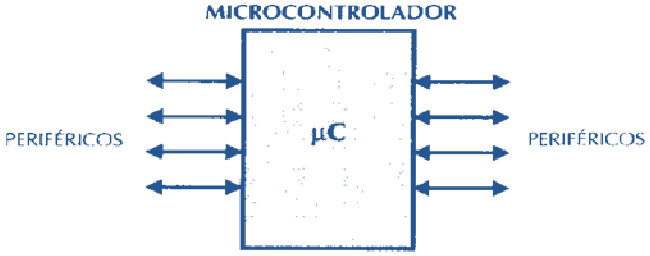


Figura II.3. El Microcontrolador en un sistema cerrado

Es posible seleccionar la capacidad de las memorias, el número de líneas de E/S, la cantidad y potencia de los elementos auxiliares, la velocidad de funcionamiento, etc. Por todo ello, un aspecto muy destacado del diseño es la selección del microcontrolador a utilizar como podemos ver en la figura II.2. y figura II.3.

2.2.2. Arquitectura de los microcontroladores

- **Arquitectura Von Neuman**

  Los microcontroladores fueron concebidos bajo una arquitectura clásica como se puede ver en la figura II.4, la cual se caracteriza por poseer una sola memoria principal en la que se almacenan indistintamente los datos e instrucciones del programa.

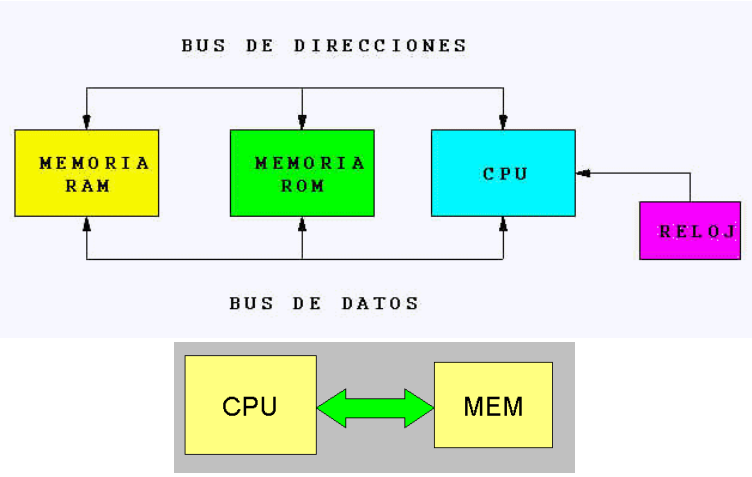


Figura II.4. Arquitectura Modelo Van Neumann

- **Arquitectura Hardvard**

  Actualmente los microcontroladores son diseñados con una arquitectura HARVARD como se puede observar en la figura II.5., la cual se fundamenta en el uso de dos memorias independientes, la primera que se conoce como

FLASH y es en la que se almacenan las instrucciones de programa; y la otra que se denomina RAM, en la cual se almacenan los datos.

Cada una de estas memorias posee su estructura de sistema de buses de acceso, lo que posibilita realizar operaciones de manera simultánea en cada memoria.

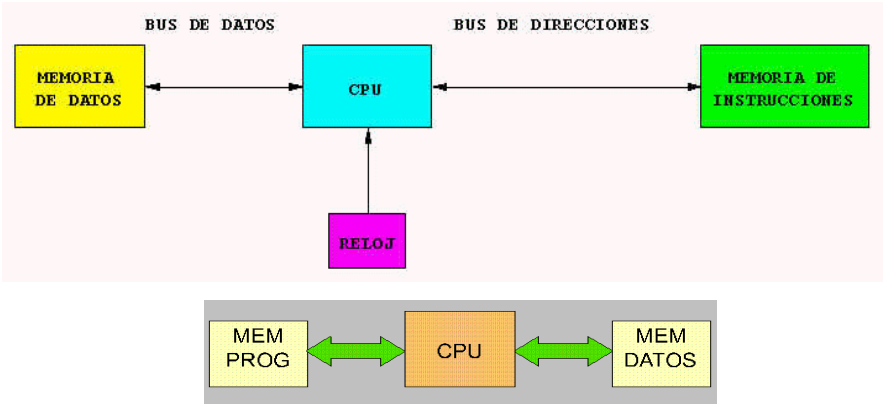


Figura II.5. Arquitectura Modelo Harvard

- **Arquitectura Risc**

  RISC, abreviatura de Reduced Instruction Set Computer es una tecnología de diseño de microprocesadores / microcontroladores en la que prima un juego de instrucciones muy pequeño y de tamaño fijo, con el que se fomenta la modularidad, segmentación y portabilidad del código.

2.2.3. Arquitectura interna de un microcontrolador

Un microcontrolador posee todos los componentes de un computador, pero con unas características fijas que no pueden alterarse.

Las partes principales de un microcontrolador son:

1. Procesador

2. Memoria no volátil para contener el programa
3. Memoria de lectura y escritura para guardar los datos
4. Líneas de EIS para los controladores de periféricos:
    a) Comunicación paralelo
    b) Comunicación serie
    c) Diversas puertas de comunicación (bus l2ºC, USB, etc.)
5. Recursos auxiliares:
    a) Circuito de reloj
    b) Temporizadores
    c) Perro Guardián (watchdog)
    d) Conversores A/D y D/A
    e) Comparadores analógicos
    f) Protección ante fallos de la alimentación
    g) Estado de reposo o de bajo consumo.

2.2.4. Partes Principales de un Microcontrolador.

- **Procesador:**

  También conocido como CPU es el encargado de coordinar la operación de todo el computador, es decir ejecución de los programas, almacenamiento de información temporal y la comunicación con los dispositivos de entrada y salida.

- **Memoria ROM:**

  Se graba el chip durante su fabricación, implica costos altos y de recomienda solo cuando se produce en serie.

- **Memoria EPROM:**

  Se graba con un dispositivo que es gobernado mediante un computador personal, que recibe el nombre de grabador, y es reprogramable, se borra con luz ultra violeta.

- **Memoria OTP:**

  Se graba por el usuario igual que la memoria EPROM, pero con la diferencia que la OTP se puede grabar una sola vez.

- **Memoria EEPROM:**

  La grabación es similar a la de las memorias EPROM y OTP, la diferencia radica en que el borrado se efectúa de la misma forma que el grabado, es decir eléctricamente.

- **Memoria FLASH:**

  Posee las mismas características que la EEPROM, pero esta tiene menor consumo de energía y mayor capacidad de almacenamiento.

- **Memoria de lectura y escritura para guardar datos:**

  Algunos micro controladores manejan la memoria RAM estática (SRAM), otros disponen de una memoria de datos tipo EEPROM, de esta forma, un corte en el suministro de alimentación no ocasiona a perdida de la información y por ende, está disponible al reiniciarse el programa.

- **Líneas de E/S para controladores de periféricos:**
  - Comunicación por puerto Paralelo.
  - Comunicación por puerto Serial.
  - Diversas puertas de comunicación.

2.2.5. **Recursos Auxiliares**.

- Temporizadores o "Timers".

- Perro guardián o "Watchdog".
- Protección ante fallo de alimentación o "Brownout".
- Estado de reposo o de bajo consumo.
- Conversor A/D.
- Conversor D/A.
- Comparador analógico.
- Modulador de anchura de impulsos o PWM.
- Puertas de E/S digitales.
- Puertas de comunicación.
- Interrupciones

Temporizadores (Timers).

Se emplean para controlar periodos de tiempo y para llevar la cuenta de acontecimientos que suceden en el exterior (contadores).

Para la medida de tiempos se carga un registro con el valor adecuado y a continuación dicho valor se va incrementando o decrementando al ritmo de los impulsos de reloj o algún múltiplo hasta que se desborde y llegue a 0, momento en el que se produce un aviso.

Perro guardián (Watchdog).

El Perro guardián consiste en un temporizador que, cuando se desborda y pasa por 0, provoca un reset automáticamente en el sistema.

Se debe diseñar el programa de trabajo que controla la tarea de forma que refresque o inicialice al Perro guardián antes de que provoque el reset. Si falla el programa o se bloquea, no se refrescará al Perro guardián y, al completar su temporización, provocara el reset.

Protección ante fallo de alimentación (Brownout).

Se trata de un circuito que resetea al microcontrolador cuando el voltaje de alimentación (VDD) es inferior a un voltaje mínimo (brownout). Mientras el voltaje de alimentación sea inferior al de brownout el dispositivo se mantiene reseteado, comenzando a funcionar normalmente cuando sobrepasa dicho valor.

Estado de reposo ó de bajo consumo.

Son abundantes las situaciones reales de trabajo en que el microcontrolador debe esperar, sin hacer nada, a que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. Para ahorrar energía, (factor clave en los aparatos portátiles), los micro controladores disponen de una instrucción especial (SLEEP en los PIC), que les pasa al estado de reposo o de bajo consumo, en el cual los requerimientos de potencia son mínimos. En dicho estado se detiene el reloj principal y se "congelan" sus circuitos asociados, quedando sumido en un estado de reposo. Al activarse una interrupción ocasionada por el acontecimiento esperado, el microcontrolador se despierta y reanuda su trabajo.

Conversor A/D (CAD).

Los microcontroladores que incorporan un Conversor A/D (Analógico/Digital) pueden procesar señales analógicas, tan abundantes en sus aplicaciones. Suelen disponer de un multiplexor que permite aplicar a la entrada del CAD diversas señales analógicas desde los pines del circuito integrado.

Conversor D/A (CDA).

Transforma los datos digitales obtenidos del procesamiento del computador en su correspondiente señal analógica que saca al exterior por uno de los pines del integrado.

Comparador Analógico.

Algunos modelos de microcontroladores disponen internamente de un Amplificador Operacional que actúa como comparador entre una señal fija de referencia y otra variable que se aplica por uno de los pines del integrado. La salida del comparador proporciona un nivel lógico 1 ó 0 según una señal sea mayor o menor que la otra.

Modulador de anchura de impulsos (PWM). Son circuitos que proporcionan en su salida impulsos de anchura variable, que se ofrecen al exterior a través de los pines del integrado.

Puertos de E/S digitales.

Todos los microcontroladores destinan algunos de sus pines a soportar líneas de E/S digitales. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertos.
Las líneas digitales de los Puertos pueden configurarse como Entrada o como Salida cargando un 1 ó un 0 en el bit correspondiente de un registro destinado a su configuración.

Puertos de comunicación.

Con objeto de dotar al microcontrolador de la posibilidad de comunicarse con otros dispositivos externos, otros buses de microprocesadores, buses de sistemas, buses de redes y poder adaptarlos con otros elementos bajo otras normas y protocolos. Algunos modelos disponen de recursos que permiten directamente esta tarea, entre los que destacan:

- UART, adaptador de comunicación serie asíncrona.
- USART, adaptador de comunicación serie síncrona y asíncrona

- USB (Universal Serial Bus), que es un moderno bus serie para los PC.
- CAN (Controller Area Network), para permitir la adaptación con redes de conexionado multiplexado.

Interrupciones:

En el microcontrolador se puede realizar interrupciones ya sea de hardware o internas, es un evento que interrumpe el programa que se esta ejecutando para ejecutar otro trozo de programa que en ese momento tiene más prioridad.

Una vez ejecutado este trozo se continúa con el programa pendiente.

- Externas:
    - Cambio de estado de un pin.
    - Recepción puerto serie.
- Internas:
    - Desbordamiento de timer
    - Fin de conversión a/d
    - Interrupción software

2.2.6. Métricas para la elección del microcontrolador.

Entre las características que ofrecen los microcontroladores se debe tomar en cuenta para su elección:

- **Sencillez de programación, manejo y grabación:** El encapsulado se recomienda que sea de tipo DIP para facilitar el modo de grabación y realizar las pruebas en un Bread Board.
- **Alta velocidad**
- **Capacidad de Memoria:** El microprocesador debe tener la memoria de

programa (Memoria Flash) en un rango entre 16 Kb y 32 Kb.

- **Bajo precio:** Esta métrica tiene gran importancia para que permita introducir al mercado el producto con mayor facilidad.
- **Disponibilidad:** Se refiere a la facilidad de encontrarlo en nuestro país.
- **Tamaño**: El factor más importante por el tamaño del microrobot.

2.2.7. Microcontrolador PIC16F877A

Bajo el nombre de esta subfamilia de microcontroladores, actualmente encontramos cuatro modelos: EL PIC 16F873/4/6 y 7. Estos microcontroladores disponen de una memoria e programa FLASH de 4 a 8 KBytes de 14 bits, considerablemente superior frente al PIC 16F84 en el que solo disponíamos de 1 Kbyte de 14 bits.

De los microcontroladores indicados, el 16F873 y el 16F876 son de 28 pines, mientras que 16F874 y el 16F877 tienen 40 pines como se puede observar en la figura II.6., lo que les permite disponer de hasta 33 líneas de E/S. En su arquitectura además incorporan:

- Varios Timer
- USART
- Bus I2C

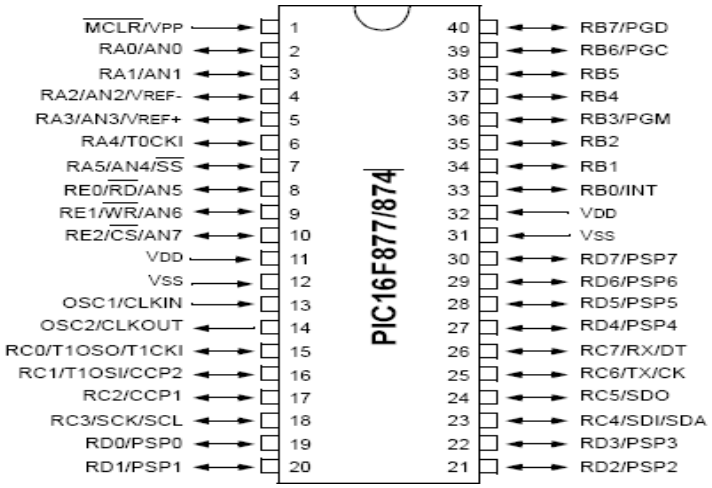


Figura II.6. Diagrama de pines PIC16F877A

Principales características

- Procesador de arquitectura RISC avanzada.
- Juego de solo 35 instrucciones con 14 bits de longitud. Todas ellas se ejecutan en un ciclo de instrucción, menos las de salto que tardan dos.
- Hasta 8K palabras de 14 bits para la Memoria de Programa, tipo FLASH en los modelos 16F876 y 16F877 y 4KB de memoria.
- Hasta 368 Bytes de memoria de Datos RAM.
- Hasta 256 Bytes de memoria de Datos EEPROM.
- Pines de salida compatibles para el PIC 16C73/74/76/77.
- Hasta 14 fuentes de interrupción internas y externas.
- Pila de 8 niveles.
- Modos de direccionamiento directo e indirecto.
- Power-on Reset (POP).
- Temporizador Power-on (POP) y Oscilador Temporizador Start-Up.
- Perro Guardián (WDT).
- Código de protección programable.
- Modo SLEEP de bajo consumo.
- Programación serie en circuito con dos pines, solo necesita 5V para programarlo en este modo.
- Voltaje de alimentación comprendido entre 2 y 5,5 V.
- Bajo consumo: < 2 mA valor para 5 V y 4 Mhz 20μA para 3V y 32 M <1 μ A en standby.

Organización de la Memoria

Hay tres bloques de memoria en cada uno de estos PlCmicro. La Memoria de Programa y la Memoria de Datos que tienen los buses separados para poder permitir

el acceso simultáneo a estos dos bloques. El tercer bloque que la Memoria de datos EEPROM.

La información adicional sobre la memoria del dispositivo puede encontrarse en el manual de referencia de los PICmicros de gama media (DS33023).

Organización de la Memoria de datos

La memoria de los datos se divide en los múltiples bancos que contiene los Registros del Propósito Generales y Los Registros de la Funciones especiales Los bit RP1 STATUS <6> y RP0 el ESTADO <5> seleccionan cada uno de estos bancos, de acuerdo a la siguiente tabla:

Tabla II.I. Selección de los bancos de memoria RAM con RP0 y RP1

RP1	**RP0**	**Banco**
0	0	Banco 0
0	1	Banco 1
1	0	Banco 2
1	1	Banco 3

En cada banco hay 7Fh posiciones de memoria (128 bytes). Las posiciones más bajas están, reservadas para los Registros de Funciones Especiales. Por encima de los Registros de Funciones Especiales se encuentran los Registros de Propósito General, que se utilizan como posiciones de memoria RAM estática. Todos están estructurados en bancos. Algunos Registros de Funciones Especiales están reflejados en varios bancos para reducir el código y tener un acceso más rápido.

Universal synchronous asynchronous receiver Transmitter (USART)

El módulo transmisor (USART) es otro de los dos puertos serie de los que dispone esta familia de microcontroladores. Los módulos de I/O. (USART también se conocen como Serial Communications o Interface o SCI). El USART puede configurarse como asíncrono full dúplex que puede comunique con los dispositivos periféricos como el terminales de CRT y los ordenadores personales, o puede configurarse como un sistema síncrono half dúplex que puede comunicar con otros microcontroladores, con dispositivos periféricos como A/D o D/A circuitos integrados, EEPROMs serie etc.

El USART puede configurarse en los modos siguientes:

- Asíncrono(Full duplex)
- Síncrono - Master (half duplex)
- Síncrono - Slave (half duplex)

Poniendo a "1" bit SPEN RCSTA <7>, y los bits TRISC <7:6>, se configura RC6/TX/CK y RC7 /RX/DT como USART.

En modo síncrono se utilizan formatos estándar: un bit de Start y ocho bits o nueve bits de Stop, siendo el más común el de ocho bits de datos.

Las Interrupciones

Esta familia de microcontroladores dispone de 14 fuentes de interrupción, de las cuales algunas de ellas se habilitan por medio del registro de control INTCON, otras, sin embargo, como la del convertidor A/D se encuentra en el registro PIE1, en este mismo registro está el bit del TIMER1 (TMRIE ,PIE1<0>) entre otros. En el registro INTCON sólo se encuentran tres controles de interrupciones. Interrupciones ocasionadas en la patilla RBO/INT, las ocasionadas por cambios de estado en las

patillas RB7:RB4 y las que tienen lugar por el desbordamiento del temporizador TMR0, el resto de los controles de periféricos y funciones especiales se encuentran en los registros PIR1, PIR2.

El bit GIE (INTCON<7> habilita las interrupciones no enmascaradas, por el contrario, cuando está a cero, deshabilita todas las interrupciones con independencia de cómo se encuentren los bits individuales de interrupciones. Cuando tiene lugar una interrupción, el valor del Contador de Programa envía al Stack y se carga con 0004 hex el vector de interrupción. El microcontrolador comienza a ejecutar las instrucciones de la subrutina de interrupciones y cuando se encuentre una instrucción RETFILE dará por finalizada la subrutina volviendo a la siguiente dirección de la última instrucción ejecutada antes de producirse la interrupción.

2.2.8. Microcontrolador PIC18F4550

Microcontroladores de 8 bits, con gran variedad de número de pines como se puede ver en la figura II.7 y prestaciones medias/altas.

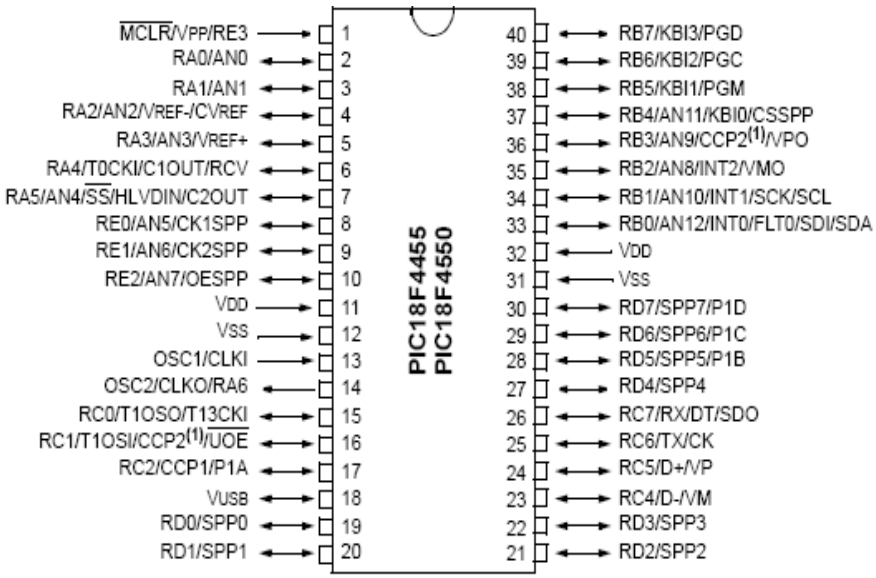


Figura II.7. Diagrama de pines PIC18F4550

Características fundamentales:

- Arquitectura RISC avanzada Harvard: 16 - bit con 8 - bit de datos.
- 77 instrucciones.
- Desde 18 a 80 pines.
- Hasta 64K bytes de programa (hasta 2 Mbytes en ROMless).
- Multiplicador Hardware 8x8.
- Hasta 3968 bytes de RAM y 1KBytes de EEPROM.
- Frecuencia máxima de reloj 40Mhz. Hasta 10 MIPS.
- Pila de 32 niveles.
- Múltiples fuentes de interrupción.
- Periféricos de comunicación avanzados (CAN y USB).

Organización de memoria:

El microcontrolador PIC18F4550 dispone de las siguientes memorias:

- Memoria de programa: memoria flash interna de 32.768 bytes
  - ✓ Almacena instrucciones y constantes/datos.
  - ✓ Puede ser escrita/leída mediante un programador externo o durante la ejecución programa mediante unos punteros.
- Memoria RAM de datos: memoria SRAM interna de 2048 bytes en la que están incluidos los registros de función especial.
  - ✓ Almacena datos de forma temporal durante la ejecución del programa.
  - ✓ Puede ser escrita/leída en tiempo de ejecución mediante diversas instrucciones.
- Memoria EEPROM de datos: memoria no volátil de 256 bytes.
  - ✓ Almacena datos que se deben conservar aun en ausencia de tensión de alimentación.

- ✓ Puede ser escrita/leída en tiempo de ejecución a través de registros.

- Pila: bloque de 31 palabras de 21 bits.
  - ✓ Almacena la dirección de la instrucción que debe ser ejecutada después de una interrupción o subrutina.
- Memoria de configuración es la memoria en la que se incluyen los bits de configuración 12 bytes de memoria flash y los registros de identificación 2 bytes de memoria de solo lectura.

Memoria de configuración:

Se trata de un bloque de memoria situado a partir de la posición 30000H de memoria de programa más allá de la zona de memoria de programa de usuario.

En esta memoria de configuración se incluyen:

- Bits de configuración: contenidos en 12 bytes de memoria flash permiten la configuración de algunas opciones del uC como:
  - ✓ Opciones del oscilador
  - ✓ Opciones de reset
  - ✓ Opciones del watchdog
  - ✓ Opciones de la circuiteria de depuración y programación
  - ✓ Opciones de protección contra escritura de memoria de programa y memoria EEPROM de datos

  Estos bits se configuran generalmente durante la programación del microcontrolador, aunque también pueden ser leídos y modificados durante la ejecución del programa.
- Registros de identificación: se trata de dos registros situados en las direcciones 3FFFFEH y 3FFFFFH que contienen información del modelo y revisión del

dispositivo. Son registros de solo lectura y no pueden ser modificados por el usuario.

Puerto A:

Dispone de 7 líneas de E/S. Las funciones alternativas son:

- RA0: entrada analógica (AN0)/ entrada de comparación (C1IN-)
- RA1: entrada analógica (AN1)/ entrada de comparación (C2IN-)
- RA2: entrada analógica (AN2)/ entrada de comparación (C2IN+)
- RA3: entrada analógica (AN3)/ entrada de comparación (C1IN+)
- RA4: entrada de reloj del Temporizador 0 (T0CKI)/salida de comparación (C1OUT)
- RA5: entrada analógica (AN4)/ salida de comparación (C2OUT)/HLVDIN entrada de detección de tensión alta/baja
- RA6: entrada del oscilador principal (OSC2)/salida de señal de reloj (CLK0).

En el reset las líneas RA0, RA1, RA2, RA3 y RA5 se configuran como líneas de entrada analógicas. Para poder utilizarlas como líneas digitales de E/S hay que desactivar la función analógica.

Puerto B:

Dispone de 8 líneas de E/S. Las funciones alternativas son:

- RB0: entrada analógica (AN12)/ interrupción externa 0 (INT0)/entrada de fallo del ECCP (FLT0)/entrada de datos del SPI (SDI)/línea de datos del I2C (SDA)
- RB1: entrada analógica (AN10)/ interrupción externa 1 (INT1)/línea de reloj del SPI (SDI)/línea de reloj del I2C (SDA)
- RB2: entrada analógica (AN8)/ interrupción externa 2 (INT2)/salida de datos del USB (VCMO)

- RB3: entrada analógica (AN9)/ línea de E/S del CCP2 (CCP2)/salida de datos del USB (VPO)
- RB4: entrada analógica (AN11)/ interrupción por cambio en pin (KBI0)/ salida de CS del SSP (CSSP)
- RB5: interrupción por cambio en pin (KBI1)/ línea de programación (PGM)
- RB6: interrupción por cambio en pin (KBI2)/ línea de programación (PGC)
- RB7: interrupción por cambio en pin (KBI3)/ línea de programación (PGD)

Puerto C:

Dispone de 5 líneas de E/S (RC0, RC1, RC2, RC6 y RC7) y 2 líneas de solo entrada (RC4 y RC5). Las funciones alternativas son:

- RC0: salida del oscilador del Temp. 1 (T1OSO)/ entrada de contador de los Temp. 1 y 3 (T13CKI)
- RC1: entrada del oscilador del Temp. 1 (T1OSI)/ línea de E/S del CCP2 (CCP2)/salida OE del transceiver del USB (UOE)
- RC2: línea de E/S del CCP1 (CCP1)/ salida PWM del ECCP1 (P1A)
- RC4: línea menos del bus USB (D-)/ línea de entrada del USB (VM)
- RC5: línea más del bus USB (D-)/ línea de entrada del USB (VP)
- RC6: salida de transmisión del EUSART (TX)/ línea de reloj del EUSART (CK)
- RC7: entrada de recepción del EUSART (RX)/ línea de datos síncrona del EUSART (DT)/ salida de datos del SPI (SDO)

Puerto D:

Dispone de 8 líneas de E/S. Las funciones alternativas son:

- RD0: línea de datos del SPP (SPP0)
- RD1: línea de datos del SPP (SPP1)

- RD2: línea de datos del SPP (SPP2)
- RD3: línea de datos del SPP (SPP3)
- RD4: línea de datos del SPP (SPP4)
- RD5: línea de datos del SPP (SPP5) / salida PWM del ECCP1 (P1B)
- RD6: línea de datos del SPP (SPP6) / salida PWM del ECCP1 (P1C)
- RD7: línea de datos del SPP (SPP7) / salida PWM del ECCP1 (P1D)

Puerto E:

Dispone de 3 líneas de E/S (RE0, RE1 y RE2) y 1 línea de solo entrada (RE3). Las funciones alternativas son:

- RE0: entrada analógica (AN5)/ salida de reloj 1 del SPP (CK1SPP)
- RE1: entrada analógica (AN6)/ salida de reloj 2 del SPP (CK2SPP)
- RE2: entrada analógica (AN7)/ salida de habilitación del SPP (OESPP)
- RE3: Línea de reset externo (MCLR) / línea de programación (VPP)

En el reset todas las líneas RE2..RE0 se configuran como entradas analógicas.

Canal de comunicación serie EUSART:

Características fundamentales:

- Modos de trabajo:
  - ✓ Modo asíncrono de 8 bits
  - ✓ Modo asíncrono de 9 bits
  - ✓ Modo síncrono Maestro
  - ✓ Modo síncrono Esclavo
- Auto-activación por detección de dato recibido
- Detección automática de velocidad de comunicación (baudrate)
- Transmisión y detección de carácter de BREAK (bus LIN)

En este tema solo se tratará el modo asíncrono básico (8 y 9 bits).

Recepción en el modo asíncrono de la USART:

El bloque de recepción de la USART incorpora un registro de desplazamiento serie (RSR). Los datos entran en serie por el pin RC7/RX y son muestreados por le bloque de lectura de bits (que trabaja a una frecuencia de 16 veces el baudrate). Si el bit de habilitación de recepción CREN está a '1', el bloque de lectura de bits identifica los bits recibidos (Start Bit, Bits de datos, 9º bit y Stop bit) y los va pasando en serie al RSR. Una vez se han recibido todos los bits correspondientes a un byte el valor recibido se pasa en paralelo de RSR al registro RCREG. De esta forma el registro RSR queda listo para recibir un nuevo dato.

2.3. ELEMENTOS DE LA INSTALACIÓN

Los elementos que componen los distintos sistemas domóticos de automatización de viviendas y edificios son muy diversos, éstos pueden ir desde una central de gestión para sistemas centralizados hasta un mando automático a distancia. Dentro de esta multiplicidad de elementos, a continuación se mencionan dos de ellos muy característicos: los sensores y los actuadores.

2.3.1. Sensor

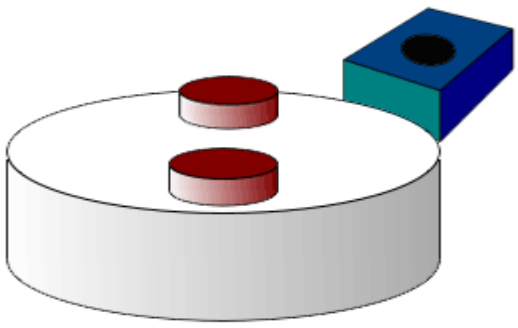


Figura II.8. Sensor de toque.

Un sensor es un aparato capaz de transformar magnitudes físicas o químicas, llamadas variables de instrumentación, en magnitudes eléctricas. Las variables de instrumentación dependen del tipo de sensor y pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, etc. Una magnitud eléctrica obtenida puede ser una resistencia eléctrica, una capacidad eléctrica como en un sensor de humedad, una tensión eléctrica como en un termopar, una corriente eléctrica como un fototransistor, etc.

Un sensor se diferencia de un transductor en que el sensor está siempre en contacto con la variable a medir o a controlar. Hay sensores que no solo sirven para medir la variable, sino también para convertirla mediante circuitos electrónicos en una señal estándar (4 a 20 mA, o 1 a 5VDC) para tener una relación lineal con los cambios de la variable censada dentro de un rango, para fines de control de dicha variable en un proceso (ver Figura II.8).

Puede decirse también que es un dispositivo que aprovecha una de sus propiedades con el fin de adaptar la señal que mide para que la pueda interpretar otro dispositivo. Como por ejemplo el termómetro de mercurio que aprovecha la propiedad que posee el mercurio de dilatarse o contraerse por la acción de la temperatura. Un sensor también puede decirse que es un dispositivo que convierte una forma de energía en otra. Áreas de aplicación de los sensores: Industria automotriz, Industria aeroespacial, Medicina, Industria de manufactura, Robótica, etc.

Los sensores pueden estar conectados a un computador para obtener ventajas como son el acceso a una base de datos, la toma de valores desde el sensor, etc.

2.3.2. Actuador

Se denominan actuadores a aquellos elementos que pueden provocar un efecto sobre un proceso automatizado, estos elementos que necesita un sistema domótico para modificar el estado de ciertos equipos e instalaciones.

Los actuadores son dispositivos capaces de generar una fuerza a partir de líquidos, de energía eléctrica y gaseosa. El actuador recibe la orden de un regulador o controlador y da una salida necesaria para activar a un elemento final de.

Existen tres tipos de actuadores:

- Hidráulicos
- Neumáticos
- Eléctricos

Los actuadores hidráulicos, neumáticos y eléctricos son usados para manejar aparatos mecatrónicos. Por lo general, los actuadores hidráulicos se emplean cuando lo que se necesita es potencia, y los neumáticos son simples posicionamientos. Sin embargo, los hidráulicos requieren mucho equipo para suministro de energía, así como de mantenimiento periódico. Por otro lado, las aplicaciones de los modelos neumáticos también son limitadas desde el punto de vista de precisión y mantenimiento.

Los actuadores eléctricos también son muy utilizados en los aparatos mecatrónicos, como por ejemplo, en los robots.

2.3.3. Pantalla Glcd

Las pantallas táctiles de última generación consisten en un cristal transparente donde se sitúa una lámina que permite al usuario interactuar directamente sobre esta superficie, utilizando un proyector para lanzar la imagen sobre la pantalla de cristal. Se

sale de lo que hasta hoy día se entendía por pantalla táctil que era básicamente un monitor táctil.

2.3.4. Panel Táctil

Un panel táctil es un fino panel autoadhesivo colocado sobre la pantalla de un LCD gráfico. Es muy sensible a la presión de manera que un suave toque provoca algunos cambios en la señal de salida. Hay diferentes tipos de paneles táctiles. El más sencillo de ellos es el panel táctil resistivo como se puede observar en la figura II.9.

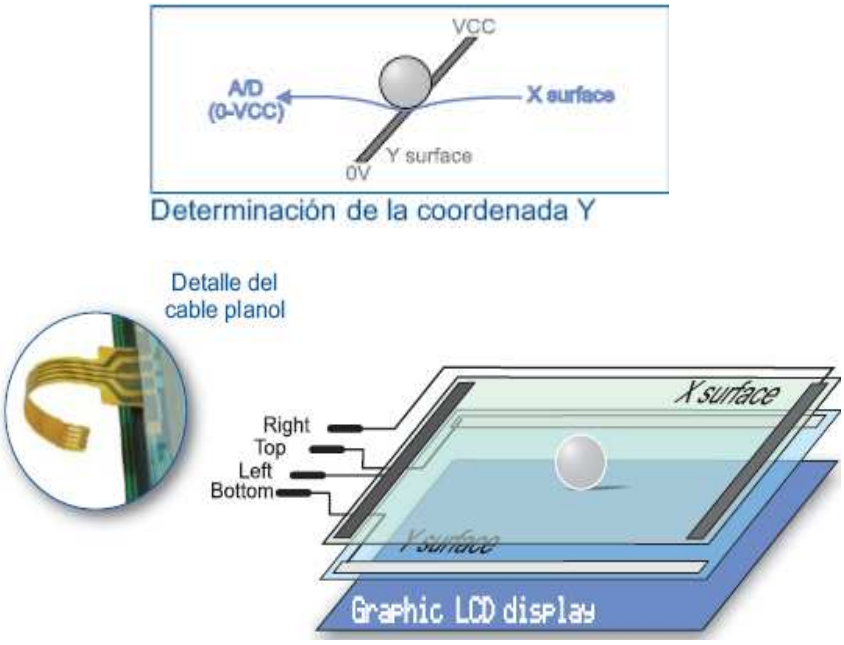


Figura II.9. Estructura interna de la pantalla táctil.

2.3.4.1. Principio de Funcionamiento.

Un panel táctil resistivo está compuesto por dos láminas rígidas transparentes, formando una estructura sándwich, que tienen una capa resistiva en sus caras internas. La resistencia de estas capas no excede normalmente de 1Kohm.

Los lados opuestos de las láminas disponen de contactos para acceder a un cable plano. El procedimiento para determinar las coordenadas de la posición del panel que ha sido presionada puede dividirse en dos pasos.

El primero es la determinación de la coordenada X y el segundo el de la coordenada Y del punto. Para determinar la coordenada X, es preciso conectar el contacto izquierdo de la superficie X a masa y el contacto derecho a la fuente de alimentación. Esto permite obtener un divisor de tensión presionando el panel táctil. El valor de la tensión obtenida en el divisor se puede leer en el contacto inferior de la superficie Y. La tensión variará en el rango de 0 V a la tensión suministrada por la fuente de alimentación y depende de la coordenada X. Si el punto está próximo al contacto izquierdo de la superficie X, la tensión estará próxima a 0 V. Para la determinación de la coordenada Y, es preciso conectar el contacto inferior de la superficie Y a masa, mientras que el contacto superior se conectará a la fuente de alimentación. En este caso, la lectora de la tensión se hará en el contacto izquierdo de la superficie X.

2.3.4.2. Conexión a un Microcontrolador

Para conectar un panel táctil al micro controlador, es preciso crear un circuito para el control del panel táctil. Por medio de este circuito, el microcontrolador conecta los contactos adecuados del panel táctil a masa y a la tensión de alimentación. El contacto inferior de la superficie Y y el contacto izquierdo de la superficie X están conectados al conversor A/D del microcontrolador. Las coordenadas X e Y se determinan midiendo la tensión en los respectivos contactos.

En software consiste en mostrar un menú en una pantalla LCD gráfica, conmutar de encendido a apagado del panel táctil control del panel táctil y leer los valores del conversor A/D que representan realmente las coordenadas X e Y de la posición. Una

vez determinadas las coordenadas, es posible decidir qué es lo que deseamos que haga al microcontrolador.

2.4. SISTEMAS EMBEBIDOS

Se entiende por sistema embebido a los sistemas electrónicos auto contenidos que poseen un microcomputador programable y que desarrollan una o más funciones específicas.

Su funcionamiento en términos generales consta de:

- Entrada (sensores y/o periféricos).
- Proceso (Tiempo real)
- Salida (respuesta, resultados, periféricos)

Es un sistema considerado como un todo, combina interfaz o periféricos de entrada, un procesador interno, software de ejecución y periféricos de salida, poseen una cantidad de recursos físicos, como memorias, periféricos, discos duros, procesadores, etc.

Estos sistemas están diseñados para realizar variadas funciones, como por ejemplo funciones repetitivas, se pueden encontrar inmersos en sistemas mayores o sistemas individuales por si solos, son sistemas programables, para distintos y variados propósitos.

Existen básicamente dos tipos de sistemas embebidos:

- Sistemas que corresponde a sistemas autónomos que funcionan por si solos, capaces de realizar varias funciones a la vez.

- Sistemas que son parte de sistemas mayores, los cuales cumplen una funcionalidad específica del sistema mayor.

2.4.1. Aplicaciones

- **Electrónica de consumo:**

  Lavadoras, congeladoras, microondas, relojes, consolas de juegos, control remoto, cámaras de video, fax, CD, DVD, GPS, televisión digital.

- **Sistemas de comunicación:**

  Sistemas de telefonía, contestadores, celulares, beepers, PDAs, enrutadores, infraestructura de redes.

- **Automóviles:**

  Inyección electrónica, frenos, elevadores de vidrios, control de asientos, instrumentación, seguridad.

- **Industria:**

  Instrumentación, monitoreo, control, robótica, control de tráfico, manejo de códigos de barras, ascensores.

- **Medicina:**

  Monitores cardiacos, renales y de apnea, marcapasos, máquina de diálisis.

2.5. SOFTWARE.

2.5.1. Compilador.

La programación en un lenguaje de alto nivel como el C ó el Basic permite disminuir el tiempo de desarrollo de un producto. No obstante, si no se programa con cuidado, el código resultante puede ser mucho más ineficiente que el programado en

ensamblador. Las versiones más potentes suelen ser muy caras, aunque para los microcontroladores más populares pueden encontrarse versiones demo limitadas e incluso compiladores gratuitos.

2.5.2. Depuración.

Debido a que los microcontroladores van a controlar dispositivos físicos, los desarrolladores necesitan herramientas que les permitan comprobar el buen funcionamiento del microcontrolador cuando es conectado al resto de circuitos.

2.5.3. Simulador.

Son capaces de ejecutar en un PC programas realizados para el microcontrolador. Los simuladores permiten tener un control absoluto sobre la ejecución de un programa, siendo ideales para la depuración de los mismos. Su gran inconveniente es que es difícil simular la entrada y salida de datos del microcontrolador. Tampoco cuentan con los posibles ruidos en las entradas, pero, al menos, permiten el paso físico de la implementación de un modo más seguro y menos costoso.

2.6. MICROCODE STUDIO

El software que se utilizó para el desarrollo del proyecto fue MICROCODE STUDIO, una vez instalado el programa, se puede acceder a través del botón Inicio de Windows.

En la figura II.10 podemos observar la pantalla principal del Microcode Studio en la parte izquierda podemos encontrar las diferentes partes de nuestro proyecto también podemos elegir el tipo de microcontrolador a utilizar.

Figura II.10. Instalación de MicroCode Studio

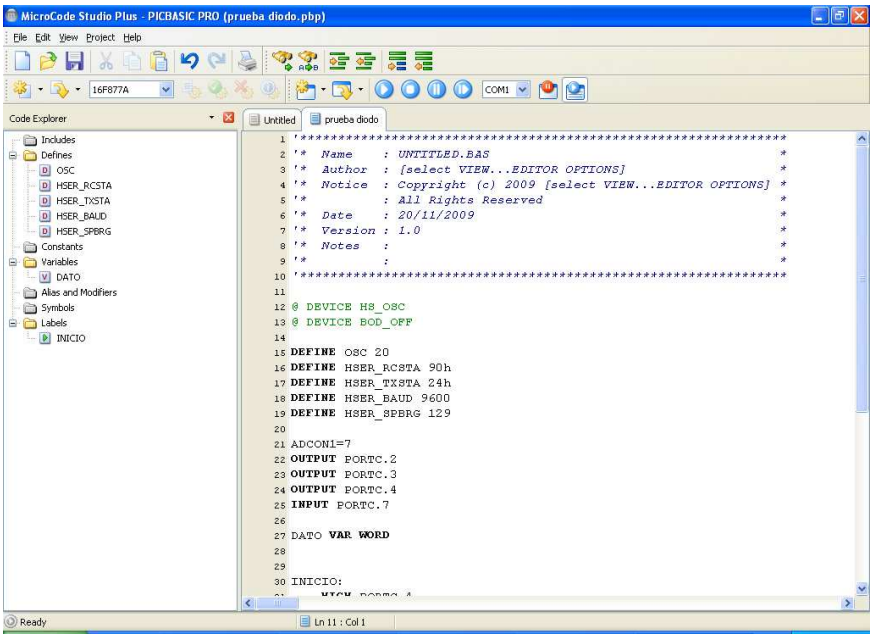


Figura II.11. Pantalla Principal de Microcode

El compilador PicBasic Pro PBP figura II.11, es un lenguaje de programación de nueva generación que hace mas fácil y rápido para usted programar micro controladores Pic micro de Microchip Technology .

El lenguaje Basic es mucho más fácil de leer y escribir que el lenguaje ensamblador Microchip. El pic basic Pro de Micro Engineering Labs Inc. es uno de los más conocidos. Este poderoso compilador pone al alcance del usuario potentes instrucciones para comunicación serie, matemática de 16 bits, mediciones de sensores analógicos, PWM, sonido, y muchísimas más.

Además de general los files “hex” y también es capaz de generar los files “asm”. De tal manera que sí se pueden hacer modificaciones de bajo nivel.

Otra magnífica característica de este compilador es que además de soportar toda la gama PIC también soporta a muchos otros de la gran familia de MICROCHIP.

2.7. MIKROBASIC

También se uso el software para el desarrollo del proyecto fue MIKROBASIC, una vez instalado el programa, se puede acceder a través del botón Inicio de Windows.

En la figura II.12 podemos observar la pantalla principal del mikroBasic en la parte izquierda podemos encontrar la configuración que estamos usando del Pic figura II.13.

Figura II.12. Instalación de MIKROBASIC

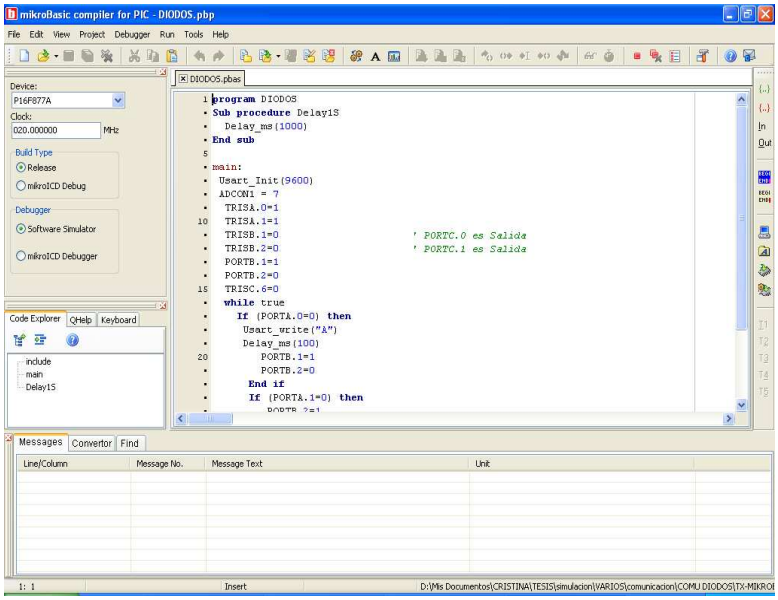


Figura II.13. Pantalla Principal de MikroBasic

Una de las razones para que los microcontroladores de Microchip sean tan populares, es la gran variedad de herramientas que se dispone para realizar aplicaciones con ellos. Entre estas herramientas se tiene a MikroBasic, que es un lenguaje de programación basado en el popular lenguaje Basic, pero se encuentra orientado a los microcontroladores de Microchip.

Si bien es cierto existen otros programas como MpLab, PicBasic, HiTech, CPic, etc., que también pueden ser utilizados para la programación de microcontroladores, MikroBasic ofrece no solo un lenguaje amigable y fácil de utilizar, sino también una amplia variedad de librerías, que permiten controlar de una forma extremadamente sencilla todos los periféricos del microcontrolador así como también periféricos externos tales como pantallas LCD sencillas y graficas. Adicionalmente MikroBasic ofrece un entorna grafico de programación con varias herramientas que facilitan la creación y prueba de aplicaciones de cualquier tipo.

2.8. PROTEUS

El software de diseño y simulación Proteus VSM es una herramienta útil para estudiantes y profesionales que desean acelerar y mejorar sus habilidades para del desarrollo de aplicaciones analógicas y digitales.

Este permite el diseño de circuitos empleando un entorno gráfico figura II.14, en el cual es posible colocar los símbolos representativos de los componentes y realizar la simulación de su funcionamiento sin el riesgo de ocasionar daños a los circuitos.

La simulación puede incluir instrumentos de medición y la inclusión de gráficas que representan las señales obtenidas en la simulación.

Lo que más interés ha despertado es la capacidad de simular adecuadamente el funcionamiento de los microcontroladores más populares (PICS, ATMEL-AVR, MOTOROLA, 8051, etc.)

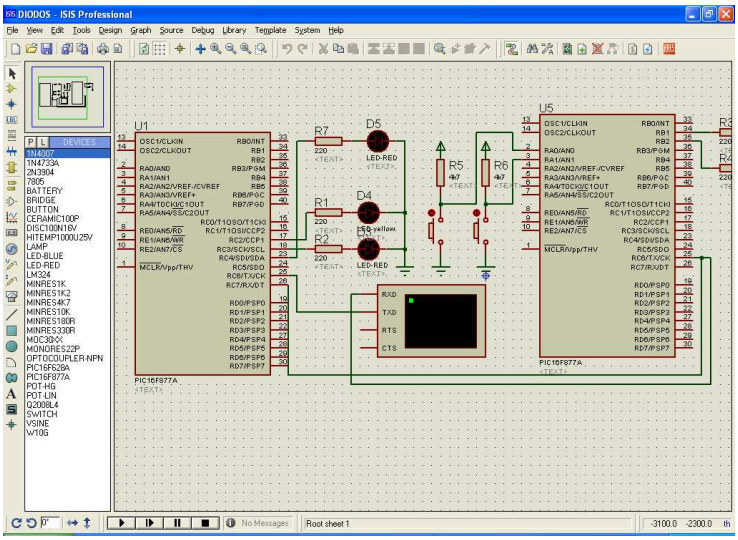


Figura II.14. Pantalla Principal de Proteus ISIS.

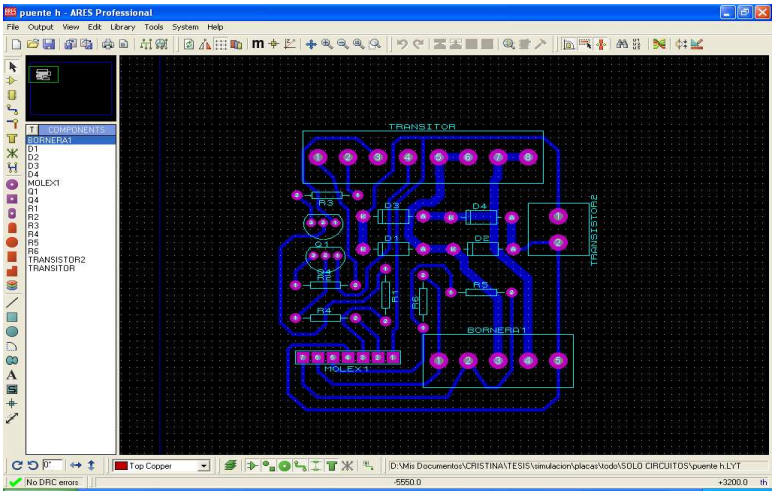


Figura II.15. Pantalla Principal de Proteus ARES.

2.8.1. Principales características del sistema PROTEUS.

- Entorno de diseño gráfico de esquemas electrónicos (ISIS) extremadamente fácil de utilizar y dotado de poderosas herramientas para facilitar el trabajo del diseñador.
- Entorno de simulación prospice mixto entre el estándar SPICE3F5 y la tecnología exclusiva de Proteus de Modelación de Sistemas Virtuales (VSM)
- Entorno de diseño de placas de circuito impreso (ARES) figura II.15, de ultra-altas prestaciones con bases de datos de 32 bits, posicionador automático de elementos y generación automática de pistas con tecnologías de autocorte y regeneración.
- Moderno y atractivo interface de usuario estandarizado a lo largo de todas las herramientas que componen el entorno PROTEUS.
- La mayor parte de los módulos que componen PROTEUS han sido escritos por el mismo equipo, garantizando al máximo nivel posible la compatibilidad e inter-operatividad de todas las herramientas que componen el entorno PROTEUS, asegurando su estabilidad al máximo.
- Ejecutable en los diferentes entornos Windows: 98, Me, 2000, XP.

- Miles de instalaciones vendidas en más de 35 países a todo lo largo del mundo.

2.9. PICKIT

Este programador es del tipo USB figura II.16, soporta las series de PIC's, y algunos dsPIC's. Viene con el código fuente.

Figura II.16. Programador PICkit2.

Cualquier programa que hagamos en el código fuente, lo compilamos y transformamos en un archivo ".hex" que es el que vamos a grabar con el PICkit2 figura II.17, al PIC 16F877A y al 18F4550.

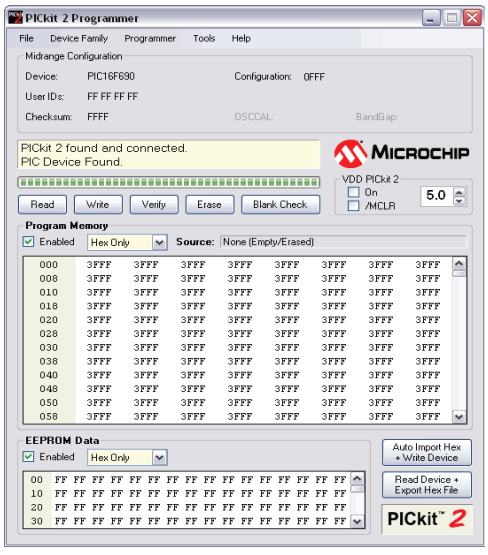


Figura II.17. Pantalla Principal del programador PICkit2 de Microchip.

CAPÍTULO III

ZIGBEE

3.1. ‘ZIGBEE’ Y EL ESTÁNDAR IEEE 802.15.4

Figura III.1. Modulo XBee

ZigBee es un protocolo de comunicaciones inalámbrico similar al Bluetooth, y basado en el estándar para Redes Inalámbricas de Área Personal (WPANs, Wireless Personal Area Networks) IEEE 802.15.4. Este estándar inalámbrico surge del fruto de una alianza, sin ánimo de lucro, de más de 200 empresas, la mayoría de ellas fabricantes de semiconductores, con el objetivo de conseguir el desarrollo e implantación de una tecnología inalámbrica de bajo costo figura III.1.

Destacan empresas como Invensys, Mitsubishi, Honeywell, Philips y Motorola que trabajan para crear un sistema estándar de comunicaciones, vía radio y bidireccional. Los miembros de esta alianza justifican el desarrollo de este estándar para cubrir el vacío que se produce por debajo del Bluetooth.

Este estándar fue diseñado pensando en la sencillez de la implementación y el bajo consumo, sin perder potencia ni posibilidades. ZigBee amplía el estándar IEEE 802.15.4 como se puede ver en la figura III.2, aportando una capa de red (NWK, Network Layer) que gestiona las tareas de enrutado y de mantenimiento de los nodos de la red; y un entorno de aplicación que proporciona una subcapa de aplicación (APS, Application Sublayer) que establece una interfaz para la capa de red, y los objetos de los dispositivos tanto de ZigBee como del diseñador.

Así pues, los estándares IEEE 802.15.4 y ZigBee se complementan proporcionando una pila completa de protocolos que permiten la comunicaciones entre multitud de dispositivos de una forma eficiente y sencilla.

En general, ZigBee resulta ideal para redes estáticas, escalables, con muchos dispositivos, pocos requisitos de ancho de banda, aquellas que son utilizadas con frecuencia relativamente baja, y donde se requiera una duración muy prolongada de la batería.

En ciertas condiciones y para determinadas aplicaciones puede ser una buena alternativa a otras tecnologías inalámbricas ya consolidadas en el mercado, como Wi-Fi y Bluetooth, aunque la falta del soporte de TCP/IP no lo hace adecuado, por si solo, para la interconexión de redes de comunicaciones IP.

Por tanto, la introducción de ZigBee no termina con la funcionalidad de otras tecnologías ya establecidas, pues este estándar trata de convivir con ellas buscando sus propios nichos de aplicación. De hecho, según Wireless Data Research Group, el mercado de redes de baja potencia y baja velocidad superó los 6.000 millones de euros en el año 2007; y comenzó va ser utilizado en áreas industriales como la automatización industrial y la domótica, antes que llegue a integrarse plenamente en las empresas.

Figura III.2. Capas de ZigBee.

3.1.1. Características del Estándar

Como se explicó anteriormente, la tecnología ZigBee está referenciada en el estándar IEEE 802.15.4, el cual fue terminado en mayo de 2003 y ratificado además por la ZigBee Alliance a finales de 2004.

Tal como se aprecia en la figura III.3, el estándar IEEE solo contempla las capas física PHY y de acceso al medio MAC, en las modalidades CSMA/CA y DSSS, mientras que las capas superiores de red y seguridad han sido establecidas por la Alianza ZigBee; finalmente la capa de aplicación corre a cargo de cada fabricante. La norma, basada en un protocolo de gran sencillez, provee un alto rendimiento en la transmisión de paquetes por radio y una alta inmunidad en ambientes con una baja relación

señal/ruido, por lo que los dispositivos ZigBee son más robustos frente a interferencias que los que siguen los estándares Bluetooth o Wi-Fi. Así en entornos de RF agresivos, como es la muy saturada banda de 2,4 GHz, ZigBee se comporta mucho mejor.

Con velocidades de 20, 40 y 250 Kbps y un alcance en el rango de 10 a 75 m, ZigBee puede funcionar en las bandas ISM6 de 2,405-2,480 GHz (16 canales), 902-928 MHz (10 canales) y 868 MHz (1 canal), aunque la mayoría de fabricantes optan por la primera, ya que puede ser usada en todo el mundo, mientras que las dos últimas sólo se aplican en Estados Unidos y Europa, respectivamente

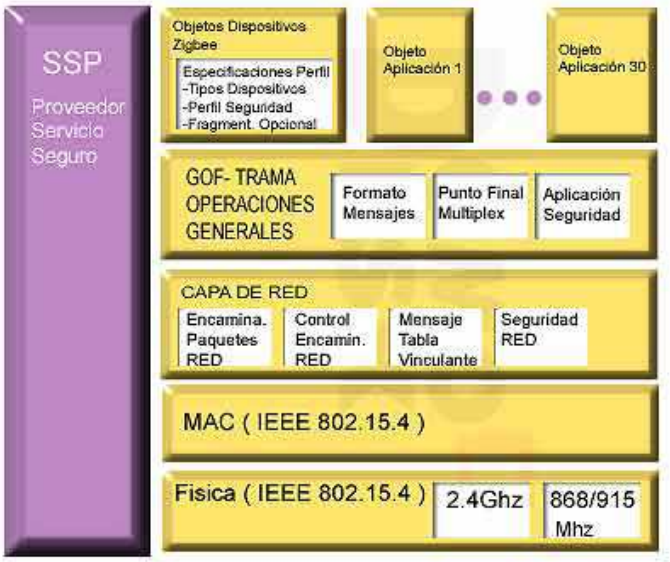


Figura III.3. Arquitectura de ZigBee.

Una red ZigBee, con topología en estrella, árbol o malla, puede escalar hasta 65.535 nodos, agrupados en subredes de hasta 255, lo que la hace más que suficiente para cubrir cualquier necesidad. En cualquiera de las subredes siempre existe un único nodo coordinador central encargado de la adquisición de datos y gestión de rutas entre dispositivos. Éstos pueden tener funcionalidad completa o reducida.

Durante la mayor parte del tiempo, el transceiver ZigBee permanece inactivo o 'dormido' con la finalidad de reducir el consumo de potencia al mínimo, pudiendo pasar al estado activo en menos de 15 milisegundos.

El objetivo es que un sensor equipado con esta tecnología pueda ser alimentado con dos pilas AA un periodo de entre seis meses y dos años, aunque en la práctica se ha verificado que se puede conseguir casi cinco años de duración en las aplicaciones de domótica y seguridad.

3.1.2. Requisitos de Hardware

En la actualidad, el mercado de los microcontroladores de 8 bits se ha desarrollado hasta el punto de estar presentes en casi todo elemento de control electrónico, y han sido la base fundamental para el desarrollo de nuevas tecnologías que están presentes en los aparatos eléctricos de uso cotidiano

Este estándar requiere un grado de implementación medio para poder ser utilizado. Esta implementación puede hacerse por software en multitud de arquitecturas. Sin embargo, independientemente de donde se implemente, necesita recursos mínimos. Ya que los dispositivos pueden efectuar diversos roles, los requisitos también varían de unos a otros.

Los recursos mínimos que debe presentar el sistema para soportar ZigBee son básicamente tres:

- Un microcontrolador de 8 bits.
- Pila completa, menor de 32 Kb.

- Pila sencilla, de 6 Kb aprox.

En cuanto a memoria RAM, cada implementación necesita una cantidad diferente, en función del grado de optimización de la misma, pero es de interés notar que los coordinadores y/o routers tendrán más exigencias puesto que necesitan mantener tablas para los dispositivos de la red, enlazado, etc.

3.1.3. Estructura Del Estándar Zigbee

En la figura III.4 se muestra los campos de los cuatro tipos de paquetes básicos: datos, ACK, MAC y baliza.

La trama de datos tiene una carga de información de hasta 104 bytes. La misma se encuentra numerada para asegurar que todas las tramas llegan, además debemos obtener confirmación de que la trama se ha recibido sin errores en el receptor. Esta estructura aumenta la fiabilidad en condiciones complicadas de transmisión.

Una estructura importante es la de ACK o reconocimiento. Esta trama es una realimentación desde el receptor al emisor, para confirmar que el paquete se ha recibido sin errores. Se puede incluir un 'tiempo de silencio' entre tramas, para enviar un pequeño ACK después de la transmisión de cada trama.

La trama MAC, se utiliza para el control remoto y la configuración de dispositivos y nodos. Una red centralizada utiliza este tipo de tramas para configurar la red a distancia.

Debido a que en esta tecnología los dispositivos pasan la mayor parte del tiempo en estado pasivo, es necesario el uso de tramas que activen los módulos; este tipo de información reciben el nombre de 'balizas'. La trama baliza literalmente despierta los

dispositivos, los cuáles escuchan para luego volver a dormirse si no reciben nada más. Estas tramas son importantes para mantener todos los dispositivos y los nodos sincronizados, sin tener que gastar una gran cantidad de batería estando todo el tiempo encendidos.

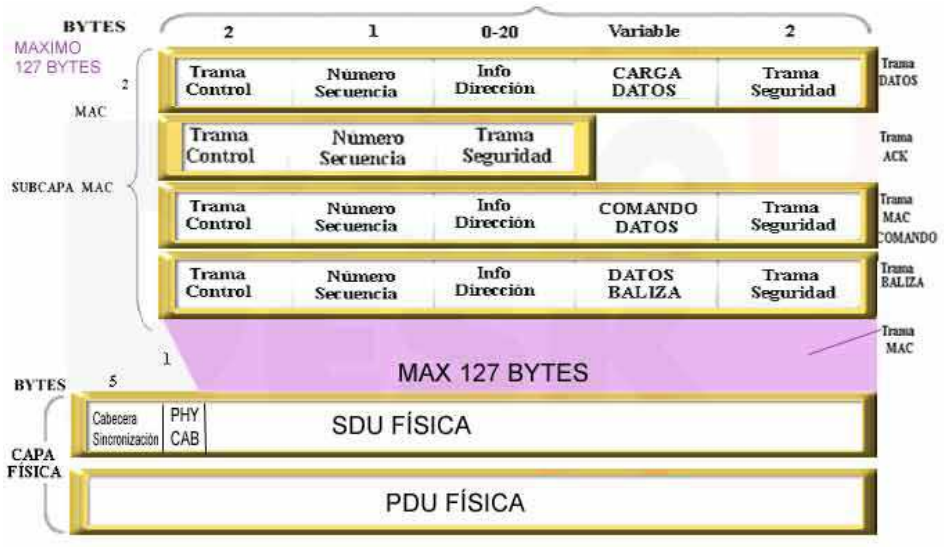


Figura III.4. Formato de Tramas.

Existe una trama denominada Trama General de Operaciones GOF la cual se encuentra entre la capa de aplicaciones y el resto de capas. La GOF suele cubrir varios elementos que son comunes a todos los dispositivos, como el sub direccionamiento para cada nodo y los modos de direccionamientos y la descripción de dispositivos, como el tipo de dispositivo, potencia, modos de 'dormir' y coordinadores de cada uno. Utilizando un modelo, la GOF especifica métodos, eventos, y formatos de datos que son utilizados para constituir comandos y las respuestas a los mismos.

Como muestra en la figura III.5, el típico dispositivo ZigBee incluye una parte con un circuito integrado de radio frecuencia RF IC con una pequeña parte de capa física PHY conectada al bajo consumo/pequeño voltaje del microcontrolador de 8-bits con

periféricos, conectados a una aplicación de sensor o actuador. La pila de protocolos y aplicaciones está implementada en un chip de memoria flash.

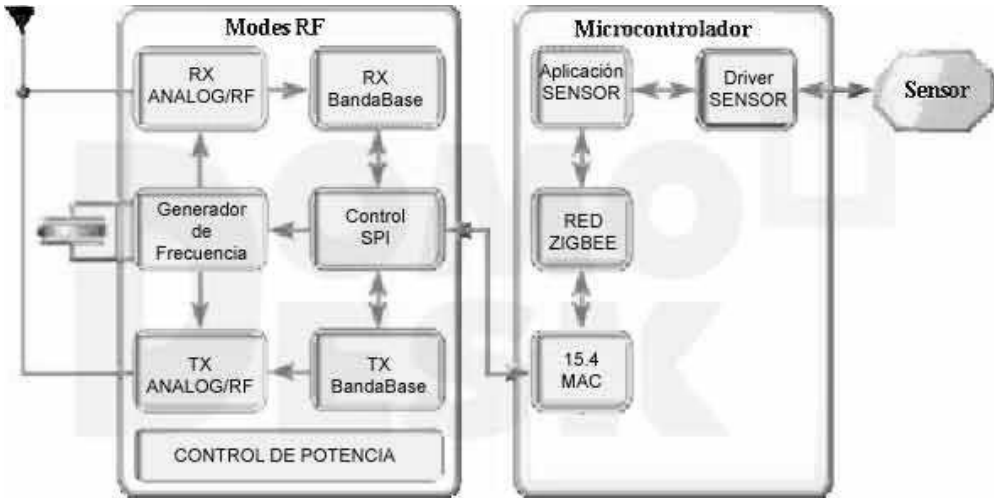


Figura III.5. Esquema típico de un dispositivo ZigBee.

3.1.4. Topologías de Red en el Estándar Zigbee

La capa de red soporta múltiples configuraciones de red incluyendo estrella, árbol y rejilla, como se puede ver en la figura III.6.

En la configuración en estrella, uno de los dispositivos tipo FFD asume el rol de coordinador de red y es responsable de inicializar y mantener los dispositivos en la red. Todos los demás dispositivos ZigBee, conocidos con el nombre de dispositivos finales, 'hablan' directamente con el coordinador.

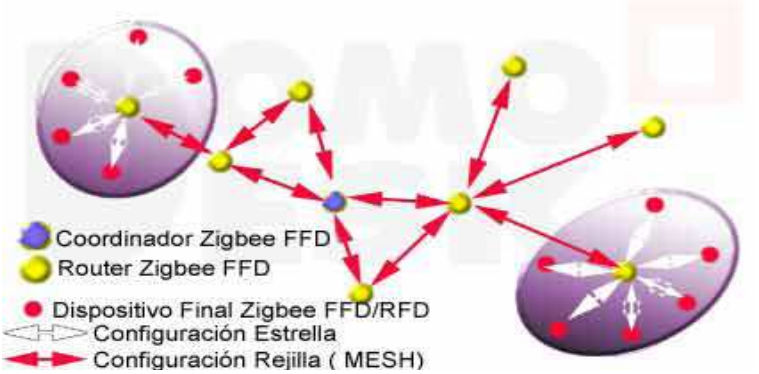


Figura III.6. Modelos de red ZigBee.

En la topología punto a punto, dos nodos solo pueden comunicarse entre sí directamente siempre y cuando se encuentren dentro del radio de alcance mutuo. Esta topología permite a ZigBee crear otras más complejas, como redes en malla o rejilla, siempre y cuando sea posible el enrutado de los datos de un nodo a otro.

En la configuración de rejilla, el coordinador ZigBee es responsable de inicializar la red y de elegir los parámetros de la red, pero la red puede ser ampliada a través del uso de routers ZigBee. El algoritmo de encaminamiento utiliza una protocolo de pregunta-respuesta request-response para eliminar las rutas que no sean óptimas, La red final puede tener hasta 254 nodos probablemente nunca sean necesarios tantos. Utilizando el direccionamiento local, se puede configurar una red de más de 65535 nodos.

3.1.5. Dispositivos del Estándar

De forma puntual, ZigBee tiene tres tipos de dispositivos, los cuales son descritos a continuación:

- **El coordinador de red:**

  Que mantiene en todo momento el control del sistema. Es el más sofisticado de los tipos de dispositivos, requiere memoria y capacidad de computación.

- **El dispositivo de función completa (FFD):**

  El mismo que es capaz de recibir mensajes del estándar 802.15.4. Éste puede funcionar como un coordinador de red. La memoria adicional y la capacidad de computar, lo hacen ideal para hacer las funciones de Router o para ser usado en dispositivos de red que actúen de interface con los usuarios.

- **El dispositivo de función reducida (RFD, Reduced Function Device):**

  Este dispositivo posee capacidad y funcionalidad limitadas especificadas en el estándar para el bajo costo y simplicidad. En pocas palabras, son los sensores/actuadores de la red.

Tanto los dispositivos de función reducida RFD, como los de función completa FFD, están diseñados para propósitos distintos.

El RFD está pensado para aplicaciones muy sencillas, como interruptores de iluminación y sensores infrarrojos, que no necesitan enviar o recibir grandes cantidades de información. Solo puede comunicarse con dispositivos FFD. Todo esto permite que pueda ser implementado usando los mínimos recursos posibles, así como un ahorro energético visible. En cambio, los FFDs pueden actuar como coordinadores o como dispositivos finales. Pueden comunicarse con otros FFDs y RFDs. Para ello necesitan más recursos, han de implementar la pila completa y precisan un consumo más exigente.

ZigBee aprovecha esta diferenciación. Además del coordinador de la red, es posible la existencia de routers, evidentemente han de ser FFD, que aumentan las posibles topologías de red, pudiendo crear no solo redes en estrella y punto a punto sino también rejillas o mallas y árboles.

Para poder tener una red, son necesarios como mínimo dos elementos. Un coordinador FFD que creará la red, le asignará el NWKID, y poseerá los mecanismos necesarios para la incorporación y eliminación de nodos en la red. Además es necesario, como mínimo, un nodo, que puede ser FFD o RFD, con el que comunicarse.

3.1.6. Estrategias de Conexión de los Dispositivos en una Red Zigbee

Las redes ZigBee han sido diseñadas para conservar la potencia en los nodos 'esclavos'. De esta forma se consigue el bajo consumo de potencia. La estrategia consiste en que, durante mucho tiempo, un dispositivo 'esclavo' está en modo inactivo, de tal forma que solo se despierta por una fracción de segundo para confirmar que está activo en la red de dispositivos de la que forma parte. Esta transición del modo dormido al modo despierto modo en el que realmente transmite, dura unos 15ms, y la confirmación y conteo de cuantos 'esclavos' conforman la red dura alrededor de 30ms.

En las redes Zigbee, se pueden usar dos tipos de entornos o sistemas:

Con balizas:

Es un mecanismo de control del consumo de potencia en la red. Permite a todos los dispositivos saber cuándo pueden transmitir. En este modelo, los dos caminos de la red tienen un distribuidor que se encarga de controlar el canal y dirigir las transmisiones. Las balizas que dan nombre a este tipo de entorno, se usan para poder sincronizar todos los dispositivos que conforman la red, identificando la red domótica, y describiendo la estructura de la trama. Los intervalos de las balizas son asignados por el coordinador de red y pueden variar desde los 15ms hasta los 4 minutos.

Este modo es más recomendable cuando el coordinador de red trabaja con una batería. Los dispositivos que conforman la red, escuchan a dicho coordinador durante el "balizamiento" envío de mensajes a todos los dispositivos o broadcast, entre 0,015 y 252 segundos. Un dispositivo que quiera intervenir, lo primero que tendrá que hacer es registrarse para el coordinador, y es entonces cuando mira si hay mensajes para él. En el caso de que no haya mensajes, este dispositivo vuelve a dormir, y se despierta

de acuerdo a un horario que ha establecido previamente el coordinador. En cuanto el coordinador termina el "balizamiento", vuelve a "dormirse".

Sin balizas:

Se usa el acceso múltiple al sistema Zigbee en una red punto a punto cercano. En este tipo, cada dispositivo es autónomo, pudiendo iniciar una conversación, en la cual los otros pueden interferir. A veces, puede ocurrir que el dispositivo destino puede no oír la petición, o que el canal esté ocupado.

Este sistema se usa típicamente en los sistemas de seguridad, en los cuales sus dispositivos sensores, detectores de movimiento o de rotura de cristales, duermen prácticamente todo el tiempo el 99,999%. Para que se les tenga en cuenta, estos elementos se despiertan de forma regular para anunciar que siguen en la red.

Cuando se produce un evento es decir cuando el sistema detecta algo, el sensor despierta instantáneamente y transmite la alarma correspondiente. Es en ese momento cuando el coordinador de red, recibe el mensaje enviado por el sensor, y activa la alarma correspondiente. En este caso, el coordinador de red se alimenta de la red principal durante todo el tiempo.

3.1.7. Comunicaciones

Los datos que realmente se desean enviar empiezan en las capas superiores de la pila, y cada capa añade información propia, formando los PDU (Protocol Data Unit).
Así pues, cuando se envía un conjunto de datos, éste contiene información de control de todas las capas de la pila. Cuando llega a su destino, cada capa extrae los datos que le concierne y, si es posible o necesario, pasa el resto a la capa superior. Así se produce una comunicación virtual entre capas de diferentes dispositivos.

Se debe analizar en primer lugar lo más básico: la comunicación a nivel físico. Los dispositivos inalámbricos envían los datos usando ondas electromagnéticas. En este caso se utiliza modulación por frecuencia FSK, en el espectro de los 2.4 GHz. Se tienen disponibles 16 canales en los que transmitir separados entre si 5 MHz. El acceso al canal se hace utilizando CSMA-CA que es un mecanismo empleado para evitar que dos dispositivos usen el mismo canal a la vez, produciendo una colisión.
Así pues, cuando un dispositivo transmite, el resto espera. Todos los dispositivos que estén en el radio de alcance del transmisor podrán escuchar el mensaje, pero la mayoría de las veces solamente es necesario comunicarse con un solo dispositivo.

Para ello, es imprescindible alguna manera de identificar los dispositivos dentro de la red. Esto lo provee la capa superior o capa MAC. Es decir, cada dispositivo posee una dirección MAC de 64 bits, la cual debe ser única. Se puede usar esta misma en las comunicaciones dentro de la red, o se puede intercambiar con el coordinador de la PAN Personal Area Network por una más corta de 16 bits. Esta dirección es la que identifica el origen y el destino de una trama dentro de la red. Cada trama debe tener un tamaño máximo de 127 bytes, incluyendo las cabeceras MAC, que son como máximo de 25 bytes sin utilizar seguridad.

Teniendo esto en cuenta, y considerando las dos topologías de red posibles a este nivel, a saber estrella y punto a punto, solo se pueden enviar datos a los nodos que estén dentro del radio de alcance. Es más, si el dispositivo es RFD, solo podrá enviar datos al coordinador; esto es muy limitado. Para solucionarlo, se debe confiar en la capa de red NWK, NetWorK.

El NLME coordina las tareas de mantenimiento de la red, crea una nueva red o también para asociarse/desasociarse a una existente, provee direccionamiento a los dispositivos, descubre nodos vecinos y rutas, controla la recepción de datos, etc.

Con las capas PHY-MAC-NWK es factible crear una red completa, permitiendo a todos los nodos, poder comunicarse con otros nodos, de la misma o de distinta red, resolviendo problemas como el acceso simultáneo al canal o el direccionamiento lógico de nodos. Aún así, para permitir más funcionalidad, se suma una capa a la pila que interactúe entre los objetos de la aplicación que desee usarla y la capa de red.
Ésta es la subcapa de Aplicación APS. La capa APS es la encargada de enviar los PDUs de una aplicación entre dos o más dispositivos llamada APSDE, APS Data Entity y de descubrir y enlazar los dispositivos y mantener una base de datos de los objetos controlados, conocida como AIB. Dos dispositivos se enlazan en la AIB en función de los servicios que ofrecen y de sus necesidades. Esto es útil para el direccionamiento indirecto; así, es posible enviar un paquete a un dispositivo en función de la dirección de origen, ya que están vinculados. Esta tabla sólo puede estar presente en el dispositivo coordinador o en uno designado a tal efecto.

Cuando la comunicación es directa, se deben especificar las direcciones de origen y destino del paquete. En cambio, en las comunicaciones indirectas, cuyo origen tenga una entrada en la tabla de enlazado, el emisor sólo necesita especificar el origen y enviar los datos al coordinador, que se encargará de hacerlos llegar al destino, que pueden ser varios dispositivos, en función de la AIB.
La capa MAC ofrece un identificador único de 64 bits para el direccionamiento, y la capa de red otro de 16 bits. Ello implica que es posible tener una gran cantidad de nodos en una misma red. Pero, si únicamente es factible la comunicación con aquellos que se encuentran dentro del radio de alcance, la red está muy limitada.

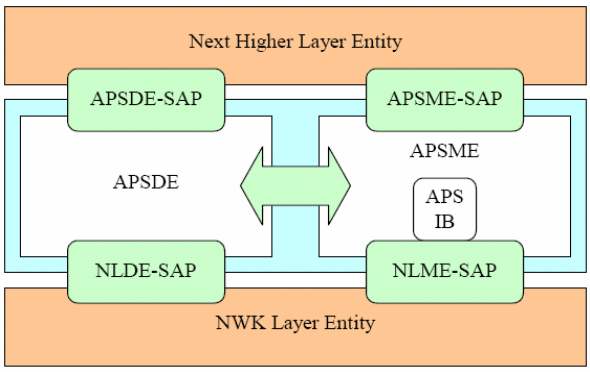


Figura III.7. Estructura de la Subcapa de Aplicación (APS).

FSK (Frequency-shift keying)

Es un tipo de modulación de frecuencia cuya señal modulante es un flujo de pulsos binarios que varía entre valores predeterminados. En los sistemas de modulación por salto de frecuencia, FSK, la señal moduladora hace variar la frecuencia de la portadora, de modo que la señal modulada resultante codifica la información asociándola a valores de frecuencia diferentes.

3.1.8. Encaminamiento Routing en el Estándar Zigbee

Para ello, existen las técnicas de encaminamiento o enrutado. Estas técnica sirven para crear dispositivos que reenvían aquellos mensajes que no van dirigidos a si mismos. Así pues, cualquier nodo de la red pude comunicarse con cualquier otro nodo.

Los routers son dispositivos de propósito específico que deben poseer toda la funcionalidad de un dispositivo FFD. No pueden entrar en modo ‘ahorro de energía’ como los RFD, ya que deben ser capaces de retransmitir los mensajes lo antes posible. Por la misma razón, han de escuchar el tráfico constantemente.

Estos dispositivos deben mantener tablas con las rutas descubiertas y funcionalidad para participar o iniciar el descubrimiento de nuevas o mejores rutas. También han de ser capaces de detectar y corregir errores. Las tablas contienen información sobre el costo de cada ruta. El costo determina cuál es la mejor en un momento dado. La función que elige el coste de una ruta se determina a la hora de crear la implementación de la pila. Se puede basar en la latencia del recorrido de los mensajes, pero es recomendable que se tenga en consideración la carga media de la batería de los dispositivos que participan en la ruta.

En las tablas de enrutado pueden aparecer direcciones de cualquier dispositivo, pero sólo los routers pueden participar en los métodos de enrutado. Si un mensaje llega a un dispositivo FFD que no es un router, comprueba la dirección de destino, y sólo lo reenvía si pertenece a alguno de sus hijos es decir, pertenece a alguno de los RFD que están asociados a él. Si es para él, lo pasa a la capa superior, en otro caso se descarta.

3.1.9. Perfiles De Zigbee

Ya que ZigBee está pensado para la comunicación entre diversos dispositivos, posiblemente de fabricantes diferentes, es necesario un mecanismo para hacer compatibles los mensajes, comandos, etc. que pueden enviarse unos a otros. Para ello existen los perfiles de ZigBee.

Los perfiles son la clave para la comunicación entre dispositivos ZigBee. Definen los métodos de comunicación, el tipo de mensajes a utilizar, los comandos disponibles y las respuestas, etc. que permiten a dispositivos separados comunicarse para crear una aplicación distribuida. Casi todo tipo de operaciones deben estar definidas en un

perfil. Por ejemplo, las tareas típicas de unirse a una red o descubrir dispositivos y servicios están soportadas por el ‘perfil de dispositivos’ ZigBee.

Cada perfil debe tener un identificador, éste es único. Por ello, la ZigBee Alliance se reserva el derecho de asignar identificadores a los diversos perfiles. Si se requiere la creación de un nuevo perfil, es necesario realizar la petición al organismo antes mencionado.

Cada perfil contiene las descripciones de los dispositivos que incluye, los identificadores de cada cluster y en su caso, sus atributos y los tipos de servicio ofrecidos. Las descripciones de los dispositivos están definidas por un valor de 16 bits es decir, hay 65535 posibles descripciones. Los identificadores de cluster son de 8 bits existen 256 posibles clusters. A su vez, si el tipo de servicio ofrecido es orientado a pares clave-valor, cada cluster puede contener atributos por valor de 16 bits o 65535 atributos por cluster. Existen muchas posibilidades dentro de un mismo perfil, y es labor del diseñador del perfil adecuar las necesidades para crear descriptores sencillos y permitir un proceso eficiente de mensajes.

Ya que cada dispositivo puede soportar más de un perfil, y cada perfil puede poseer varios clusters y múltiples descripciones, es necesaria una jerarquía de direccionamiento para acceder a los elementos del dispositivo. En primer lugar, se hace referencia al dispositivo entero usando sus direcciones IEEE y NWK.
Por otro lado, se definen los Endpoints, que son campos de 8 bits que apuntan a cada una de las diferentes aplicaciones que están soportadas por un dispositivo. Por ejemplo, el 0x00 hace referencia al endpoint del perfil de dispositivo, y el 0xFF hace referencia a todos los endpoints activos endpoint broadcasting. Ya que los endpoints

0xF1-0xFE están reservados, es posible tener un total de 240 aplicaciones en los endpoints 0x01-0xF0.

Una vez establecidos los endpoints para las aplicaciones deben definirse los descriptores. Como mínimo, el descriptor simple ha de estar presente y disponible para las tareas de descubrimiento de servicio. Existen otros tipos de descriptores que contienen información acerca de la aplicación, del servicio, etc. Pueden contener información sobre el endpoint al que hacen referencia, el perfil o la versión, pero también sobre el tipo de alimentación del dispositivo, el nivel de la batería, el fabricante, número de serie y hasta un icono o la dirección de éste para representar el nodo en PCs, PDAs etc.

3.1.10. El Estándar Zigbee y la Domótica

Una vez analizado el estándar ZigBee, es evidente que la principal conclusión que se puede obtener es que el mismo fue creado para aplicaciones que no requieren de una tasa de comunicación alta sino que utilizan el ancho de banda del canal de forma limitada y en algunos casos, solo en ciertos momentos, pues los dispositivos de la red permanecen inactivos la mayor parte del tiempo.

En la tabla.III.1. Se encuentran representadas de forma comparativa tres de las tecnologías inalámbricas más conocidas (incluyendo a ZigBee) y ya en proceso de expansión.

Si se toma en cuenta la clase de comunicaciones que se producen en una red de sensores o actuadores de un Sistema Domótico, se determina que muchas de estas comunicaciones se realizan con pequeños paquetes de datos para enviar información de un sensor, o simplemente para controlar el estado de los sensores. Además de ser

paquetes pequeños de información, la gran mayoría de los dispositivos pueden estar dormidos hasta que envíen la información y activarse al detectar algo.

Tabla III.I. Comparación de las tres principales tecnologías Wireless.

Estándar	Wi-Fi	Bluetooth	ZigBee
Tasa de Bits	Hasta 54 Mbps	1 Mbps	250 Kbps (2.4 GHz) 40 Kbps (915 MHz) 20 Kbps (868 GHz)
Consumo de Potencia	400mA transmitiendo, 20mA en reposo	40mA transmitiendo, 0.2mA en reposo	30mA transmitiendo, 3mA en reposo
Ventajas	Gran tasa de bits	Inter operactividad, sustituto del cable	Batería de larga duración, bajo costo
Aplicaciones	Navegar por Internet, redes de ordenadores, transferencia de ficheros, dentro de un edificio	Wireless USB, móviles, informática casera, teléfonos	Control e bajo costo y monitoreo, control remoto, productos dependientes de la batería, sensores , juguetería
Alcance	De 20 a 200 metros	100 metros en interior	30 metros en interior

Banda de Frecuencia	2.4 GHz	2.4GHz	2.4GHz, 868/915MHz
Tamaño de la pila	~1Mb	~1Mb	~20kb
Numero de canales	11 – 14	79	16 (2.4 GHz) 10 (915 MHz) 1 (868 GHz)
Tipos de Datos	Digital	Digital, Audio	Digital (texto)
Números de Dispositivos	32	8	255/655535
Requisitos de Alimentación	Media- Horas de batería	Media- Días de batería	Muy Baja- Años de batería
Introducción al Mercado	Alta	Media	baja
Arquitecturas	Estrella	Estrella	Estrella, Árbol, Punto a Punto, y Malla
Precio	Costoso	Accesible	Bajo
Complejidad	Complejo	Complejo	Simple

Las principales características de estos sensores son:

- Un consumo de potencia extremadamente bajo.
- La posibilidad de estar ‘dormidos’ durante grandes periodos de tiempo.
- Su sencillez.
- Su bajo costo.

Tomando en cuenta las presentes circunstancias de análisis, ZigBee resulta una tecnología más que adecuada para ser utilizada en un Sistema Domótico, pues facilita

la comunicación entre sensores o actuadores y cumple con las velocidades de comunicación que necesitan este tipo de dispositivos. Es importante anotar que uno de los propósitos básicos de la ZigBee Alliance al momento de crear esta tecnología, fue el proporcionar un estándar que sirva para la automatización y control de Sistemas Domóticos.

3.1.11. Futuro de Zigbee

Se espera que los módulos ZigBee sean los transmisores inalámbricos más baratos de la historia, y además producidos de forma masiva. Tendrán un costo bajo, y dispondrán de una antena integrada, control de frecuencia y una pequeña batería. Ofrecerán una solución tan económica porque la radio se puede fabricar con muchos menos circuitos analógicos de los que se necesitan habitualmente.

3.2. DIFERENCIAS CON EL ESTÁNDAR BLUETOOTH

Pese a que ZigBee y Bluetooth son estándares inalámbricos cuyas características son parecidas, cabe resaltar aquellos aspectos que los diferencian entre sí y los vuelven ideales para aplicaciones completamente diferentes.

Dentro de las principales diferencias que es posible anotar, se mencionan las siguientes:

- Una red ZigBee puede constar de un máximo de 65.535 nodos, frente a los 8 máximos de una red Bluetooth.
- Menor consumo eléctrico que el ya de por sí bajo del Bluetooth. En términos exactos, ZigBee tiene un consumo de 30 mA transmitiendo y de 3 mA en reposo, frente a los 40 mA transmitiendo y 0.2 mA en reposo que tiene Bluetooth. Este menor consumo se debe a que el sistema ZigBee se queda la

mayor parte del tiempo dormido, mientras que en una comunicación Bluetooth esto no se puede dar, pues los dispositivos siempre se encontrarán transmitiendo y/o recibiendo.

- ZigBee posee una velocidad de transmisión de RF de 250 Kbps, mientras que la velocidad de Bluetooth es de 1 Mbps.
- Debido al ancho de banda de cada tecnología, una es más apropiado que la otra para ciertas cosas. Por ejemplo, mientras que Bluetooth se usa para aplicaciones como Wireless USB, los teléfonos móviles y la informática casera; el ancho de banda de ZigBee se hace insuficiente para estas tareas, desviándolo a usos tales como la Domótica, los productos dependientes de la batería, los sensores médicos, y en artículos de juguetería, en los cuales la transferencia de datos es menor.

CAPÍTULO IV

DISEÑO DEL PROYECTO.

4.1. REQUERIMIENTOS DEL SISTEMA

En este proyecto fue diseñado e implementado para el control domótico utilizando una nueva tecnología como lo es ZigBee para la sala de conferencia de la Empresa Eléctrica Riobamba S.A. Esta sala de conferencias esta distribuida de la siguiente manera como se puede ver en la figura IV.1, posee dos puertas de ingreso que se ubican en la parte delantera, dos persianas que están en la parte posterior, tres ventanas que se encuentran cerca de las persianas, luces fijas que están distribuidas en el centro y los dos juegos de dimmers que se ubican alrededor de la sala.

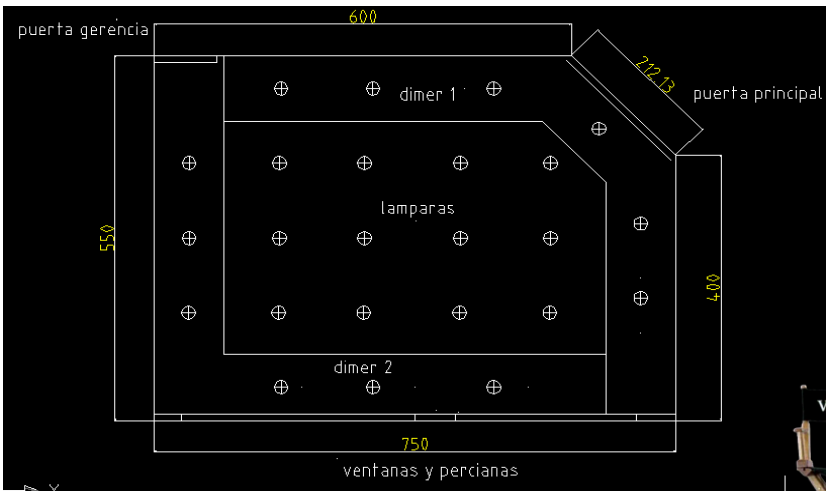


Figura IV.1. Plano de la Sala.

Este sistema esta conformado por un modulo de mando que esta en el control remoto el que fue diseñado con una pantalla táctil y módulos de respuesta que controlan los diferentes actuadores como son los motores para la apertura y cerradura de ventanas y cortinas en sus dos formas de abertura y cerradura, los dimmers y el encendido y apagado de la luz. Con este proyecto probamos las ventajas de la tecnología ZigBee para el control domótico, por lo que este producto se puede aplicar tanto en casas como en empresas y centros comerciales por su largo alcance y su extenso número de nodos que puede llegar hasta 655535 con ruteador.

Control Remoto

El control remoto como se puede observar en la figura IV.2 esta realizado con una pantalla táctil en la que se visualizan los diferentes menús para la ejecución de la tarea seleccionada, en donde el microcontrolador PIC18F4550 recepta y procesa las instrucciones para poder transmitir con el estándar Zigbee.

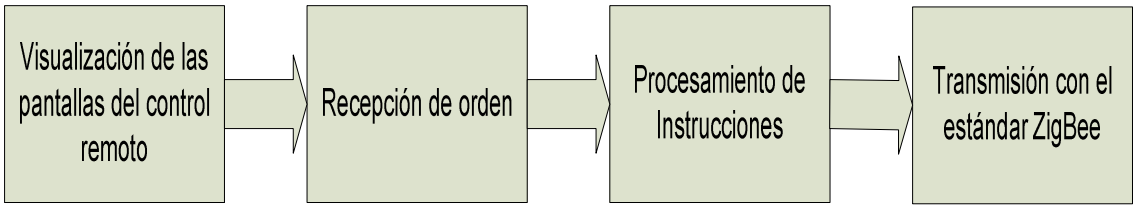


Figura IV.2. Diagrama de Bloques del Control Remoto.

Luces

Para el control de las luces el Modulo Xbee recepta las órdenes enviadas por el control remoto las que ingresan al microcontrolador PIC16F877A conjuntamente con la señal de corriente alterna de un transformador de 5 Voltios, en donde el microcontrolador activa los circuitos de los dimmer y el de la luz fija como se puede observar en la figura IV.3.

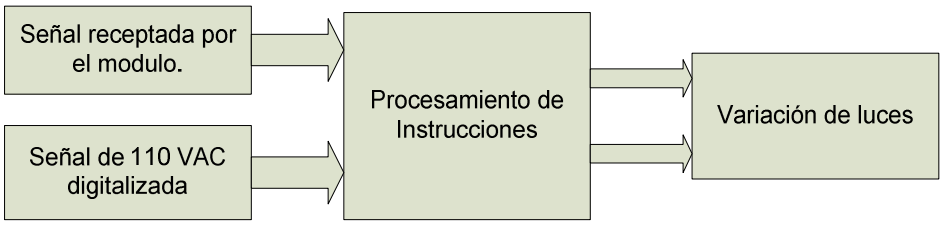


Figura IV.3. Diagrama de Bloques de las luces.

Ventanas y Cortinas

Para abrir y cerrar las cortinas y ventanas, por medio del Modulo Xbee como se puede ver en la figura IV.4. se recepto la señal enviada por el control remoto la que ingresa al microcontrolador PIC16F877A, el microcontrolador es el encargado de activar una de las siete placas de los Puentes H para hacer girar al motor de 12V de corriente directa.

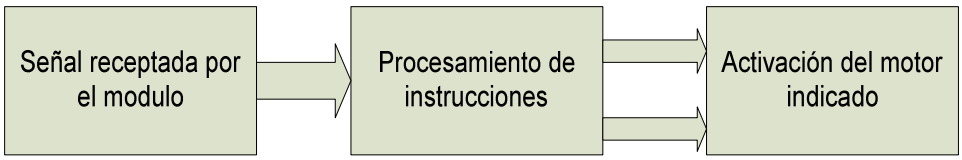


Figura IV.4. Diagrama de Bloques de los motores.

4.2. PARTES DEL SISTEMA

En el desarrollo de cualquier proyecto siempre hay que tener un orden de actuación determinado, es decir una buena planificación. En este caso, se ha recurrido a dividirlo en dos niveles o etapas:

Electrónica de control. Incluye los circuitos más básicos que relacionan las salidas de los sensores con las restantes unidades. Partiendo de una simple lógica digital hasta potentes microcontroladores, se busca dotar al proyecto de la capacidad para

procesar la información, así como actuar de una manera controlada e inmediata sobre las unidades actuadoras.

Estructura mecánica. Comprende la estructura física para reducción de velocidad de algunos motores.

4.2.1. Estructura Mecánica

Para diseñar el proyecto se ha tenido en cuenta todas las ideas anteriores y alguna otra añadida por la propia experiencia de los autores. Los Piñones y Poleas serán de aluminio al igual que los brazos de las ventanas.

4.2.2. Diseño Electrónico

El proyecto depende de un control ejecutado por tres módulos xbee y tres microcontroladores PIC, uno PIC18F4550 para el control remoto y dos PIC16F877A para las luces, y los motores de las cortinas y persianas, los cuales con la señal de encendido se sincronizan.

4.2.2.1. Control Remoto

El control remoto esta construido con el microcontrolador PIC18F4550, por tener la memoria suficiente para mostrar las pantallas requeridas como las podemos ver en las distintas figuras que se presentan a continuación, estas pantallas son mostradas en un LCD grafico de 128x64 pixeles así como también para recibir las posiciones detectadas por el touch screen que viene incorporado en el GLCD, con dichas posiciones nos permite saber la necesidad del usuario para que el PIC pueda transmitir la orden necesaria por el Modulo Xbee.

Para el correcto funcionamiento del microcontrolador lo activaremos mediante una resistencia de 10KΩ conectado a 5V y ésta a su vez al pin 1 del pic que es el MCLR.

Para el modo de oscilación utilizamos el modo HS, un cristal de 20 MHz que es conectado a los pines OSC1 y OSC2 para establecer una oscilación. La Figura IV.5 muestra la conexión de estos pines a un oscilador de cristal. El diseño del oscilador requiere el uso de un cristal que sea de tipo para operación resonante en paralelo con 2 capacitores de 22pF.

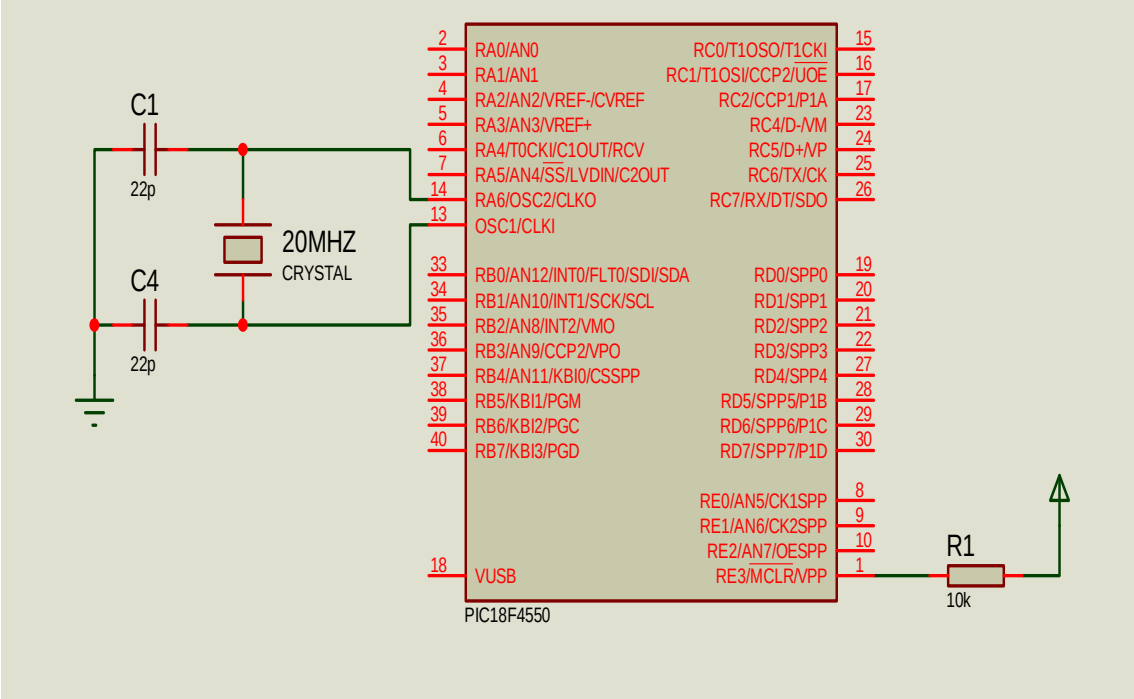


Figura IV.5. Esquema del oscilador del PIC del Control Remoto.

Para la transmisión el modulo para la transmisión por el modulo XBee la conexión será por el pin 25 RC7 (TX) del pic que realiza la función de comunicación serial (Usart), que solo se puede utilizar con la sentencia de programación Usart_write en MikroBasic para la transmisión de los datos como podemos ver en la figura IV.6.

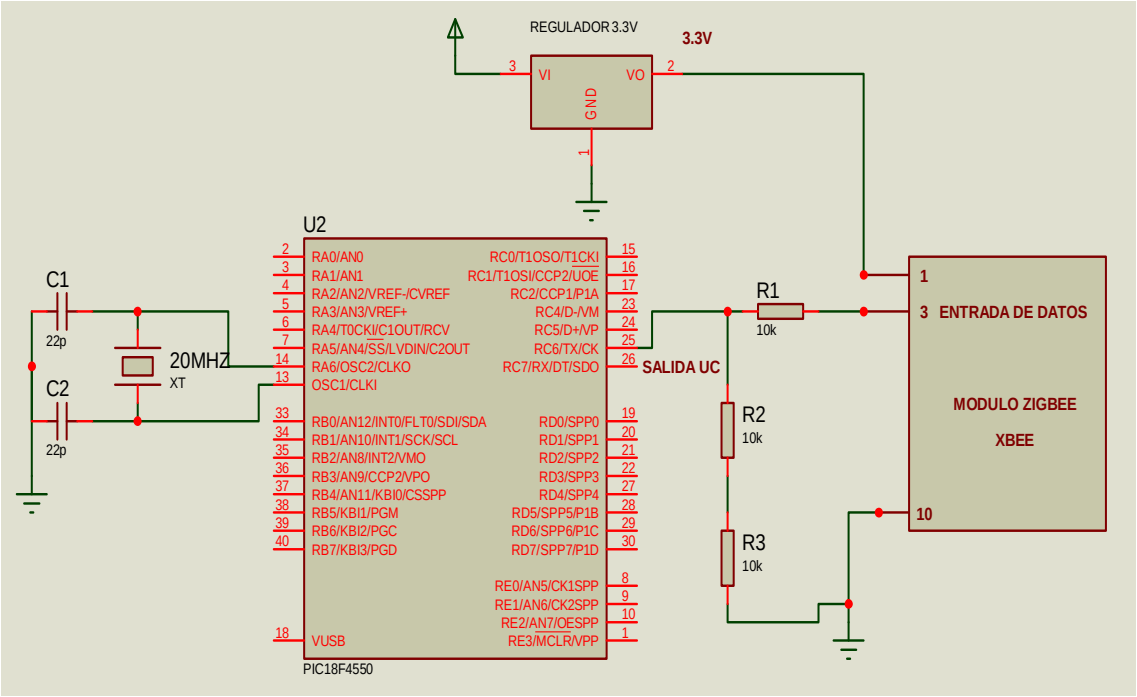


Figura IV.6. Conexión de Transmisor ZigBee.

Para visualizar las pantallas de los distintos menús en el GLCD, utilizaremos el puerto D y el puerto RB0 hasta el RB5, para la recepción del touch screen utilizaremos los pines de puerto RA0, RA1, RC0, RC1 como lo podemos ver en la figura IV.7.

En la figura IV.11 se puede ver la pantalla de inicio que durara unos segundos al encenderse el control remoto, en la figura IV.12 se encuentra el menú en el que se puede escoger las luces, ventanas, persianas y el off que nos permite apagar todas las luces, cerrar todas las ventanas y cortinas, en la figura IV.13 podemos ver el menú de luces que permite apagar y encender la luz, y también aumentar o disminuir indistintamente la intensidad de cada dimmer, en la figura IV.14 esta el menú de persianas con las flechas para arriba podemos abrir , las flechas para abajo podemos cerrar transversalmente, las flechas a la izquierda podemos abrir y con las flechas a la derecha podemos cerrar horizontalmente, en la figura IV.15 esta el menú de ventanas con el signo mas cerramos las ventanas y con el menos cerramos las ventanas.

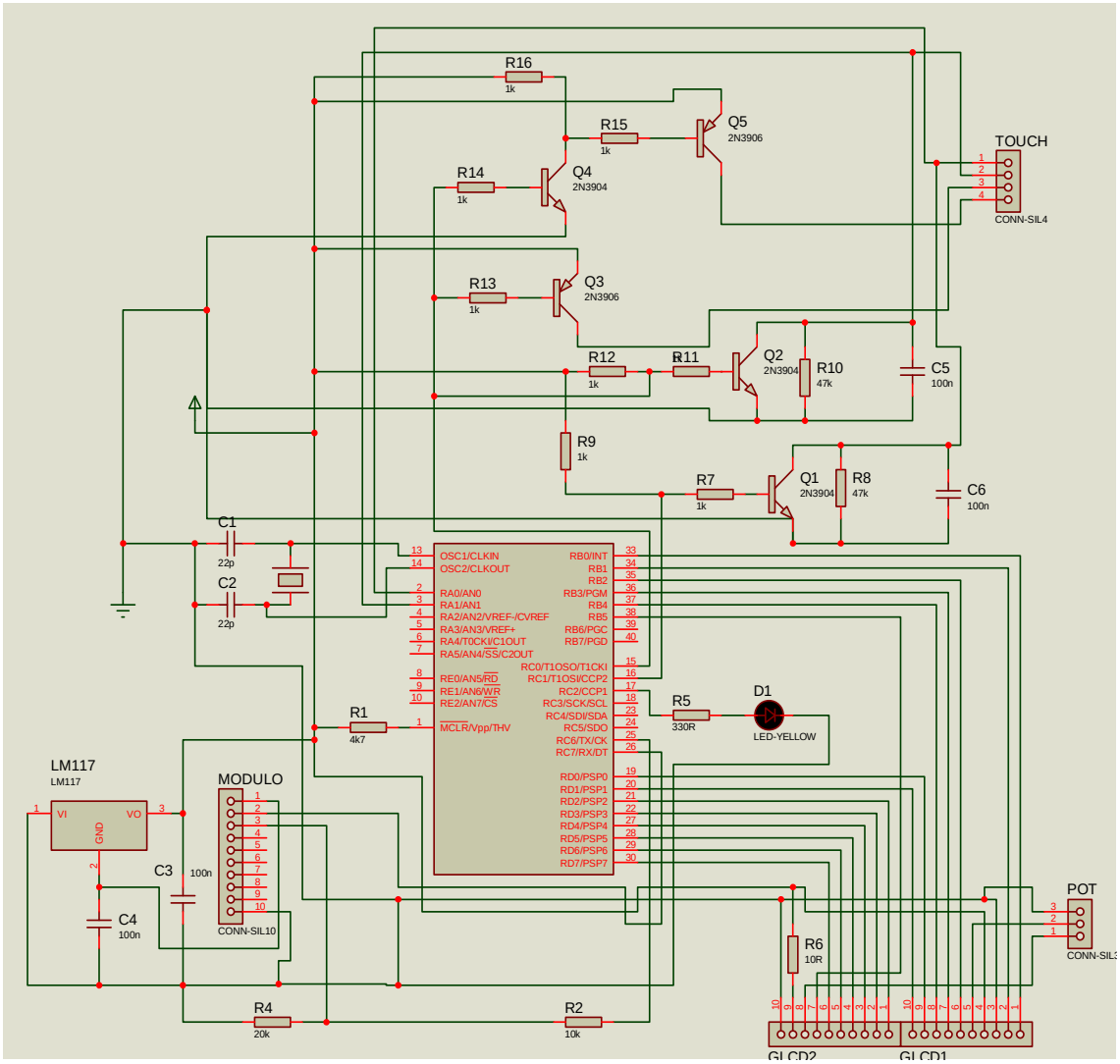


Figura IV.7. Diseño electrónico del control remoto.

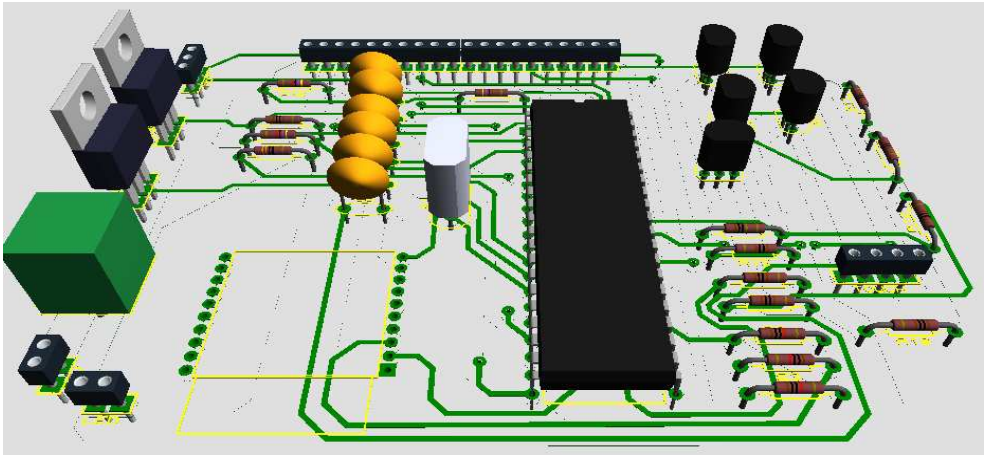


Figura IV.8. Distribución de la placa del control remoto.

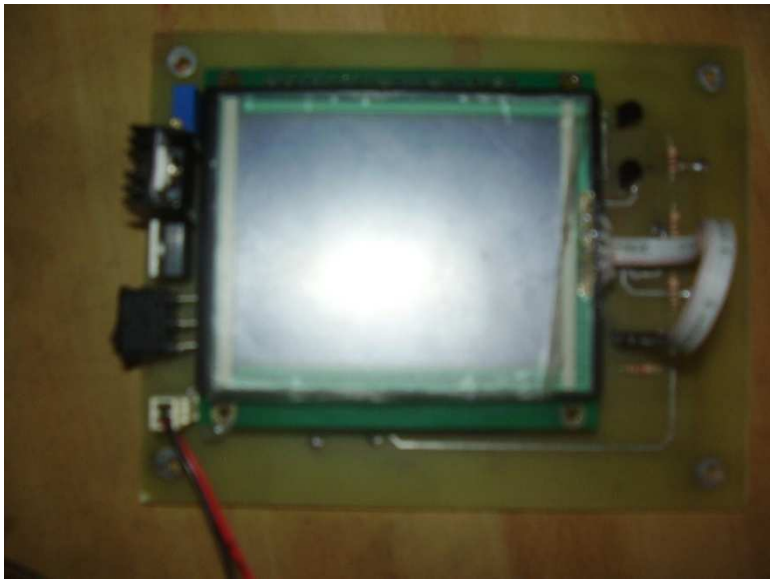


Figura IV.9. Placa del Control remoto.

Usart_write

Envía uno o varios datos en serie que es una comunicación consecutiva asincrónica. El Usart_write es una de varias funciones consecutivas asincrónicas incorporadas, sólo puede ser usado con dispositivos que tienen un hardware USART.

Los parámetros consecutivos y la velocidad de transmisión se configuran automáticamente en Mikrobasic.

Figura IV.10. Pantalla de Inicio.

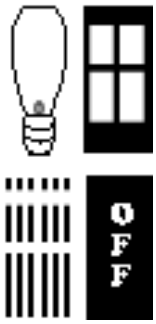


Figura IV.11. Menú del Control remoto.

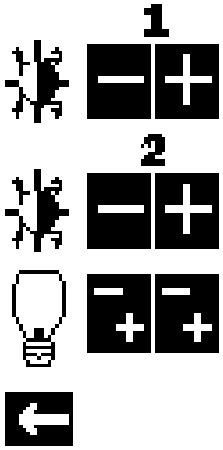


Figura IV.12. Menú de luces.

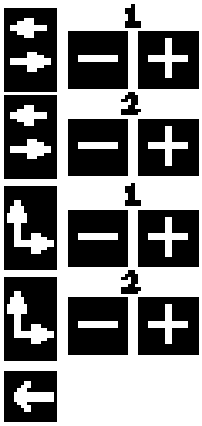


Figura IV.13. Menú de persianas.

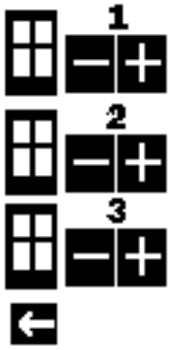


Figura IV.14. Menú de ventanas.

4.2.2.2. Luces

Para el control de las luces se utilizo el microcontrolador PIC16F7877A, para su correcto funcionamiento lo activaremos mediante una resistencia de 10KΩ conectado a 5V y ésta a su vez al pin 1 del pic que es el MCLR.

Para el modo de oscilación utilizamos el modo HS, un cristal de 20 MHz que es conectado a los pines OSC1 y OSC2 para establecer una oscilación. La Figura IV.16 muestra la conexión de estos pines a un oscilador de cristal. El diseño del oscilador

requiere el uso de un cristal que sea de tipo para operación resonante en paralelo con 2 capacitores de 22pF.

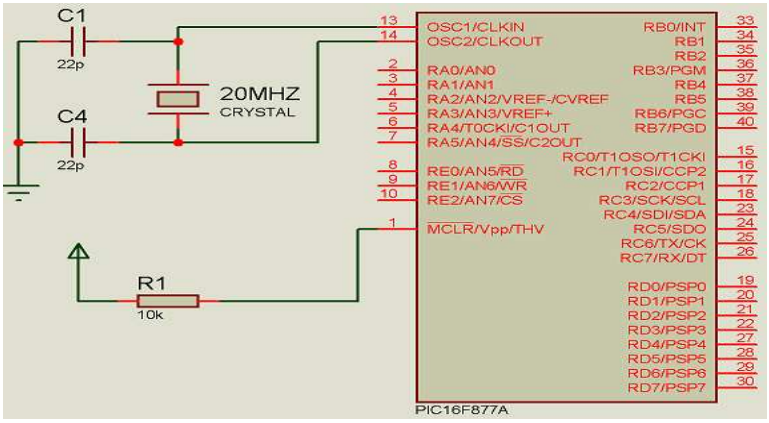


Figura IV.15. Esquema del oscilador del PIC del control de las luces.

Para el control de las luces se utilizo un modulo Xbee Configurado como receptor como se puede ver en la figura IV.17, la recepción del pic se realizo por el pin 26 RC7 (RX) del pic que realiza la comunicación serial (Usart), que solo se puede utilizar con la sentencia de programación HSerin en MicroCode.

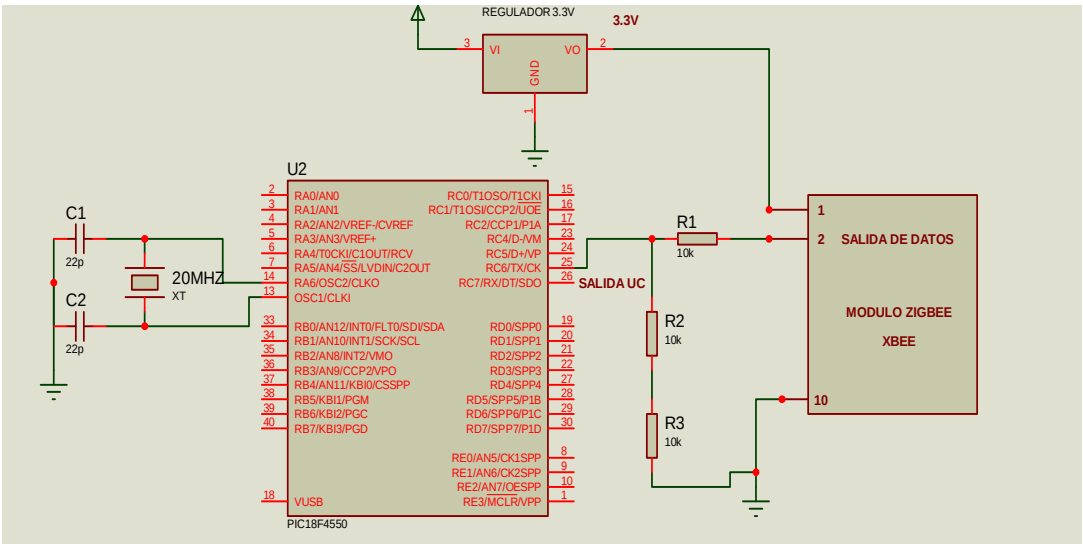


Figura IV.16. Conexión de Transmisor ZigBee.

HSERIN

Envíe uno o varios datos en serie que es una comunicación consecutiva asincrónica. El HSERIN es una de varias funciones consecutivas asincrónicas incorporadas, sólo puede ser usado con dispositivos que tienen un hardware USART.

Los parámetros consecutivos y la velocidad de transmisión son especificados de la siguiente manera:

DEFINE OSC 20	Determinar la frecuencia de operación.
DEFINA HSER_RCSTA 90	Determinar la habilitación del registro de recepción.
DEFINA HSER_TXSTA 24h	Determinación de la velocidad de transmisión.
DEFINA HSER_BAUD 9600	Determinar la rata de transmisión y recepción.
DEFINA HSER_SPBRG 129	Determina directamente el SPBRG (normalmente puesto por HSER_BAUD)

Para que el programa funcione es necesario trabajar con interrupciones de hardware como de periféricos, de hardware es necesario para detectar el cruce por cero para regular la luz de los dimmers, y la interrupción de periféricos es necesaria para que el pic pueda seguir trabajando mientras esta esperando recibir los datos receptados por el modulo Xbee.

Los registros se configuraron de la siguiente manera para las interrupciones:

ON INTERRUPT GoTo REGULAR

PIE1.5=1

INTCON=%11010000

OPTION_REG.6=1

El procedimiento de interrupción fu el siguiente:

Disable

REGULAR:

```
IF PIR1.5=1 Then          'RCIF=1
  HSerin[DATO]
EndIF
 PIE1.5=1
 INTCON=%11010000
 Resume
```

Enable

En la figura IV.18 podemos observar la disposición de los componentes utilizados en el diagrama principal el ingreso de la señal para detectar el cruce por cero fue por el puerto RB0 en el pin 33, el puerto RB3 en el pin 36 regula la placa del dimmer 1, el puerto RB4 en el pin 37 regula la placa del dimmer 2 y el puerto RD0 en el pin 19 para encender y apagar la luz fija.

Dimmer

El circuito como podemos ver en la figura IV.19 y su vista real en la figura IV.20 contiene un transformador de corriente alterna cuya salida es conectada al microcontrolador a fin de sensar el cruce por cero de la onda sinosoidal y poder sincronizar la operación del software como se puede ver en la figura IV.18. El primer pin del molex es utilizado como salida de los pulsos que disparan el optoacoplador que estamos utilizando en nuestro caso el optoacoplador K3020 que produce el disparo del triac de potencia BT139 tanto en el semiciclo positivo como en el negativo. Haciendo

variar la fase del disparo podemos regular la potencia en la carga para elevar y bajar respectivamente la luminosidad del los focos.

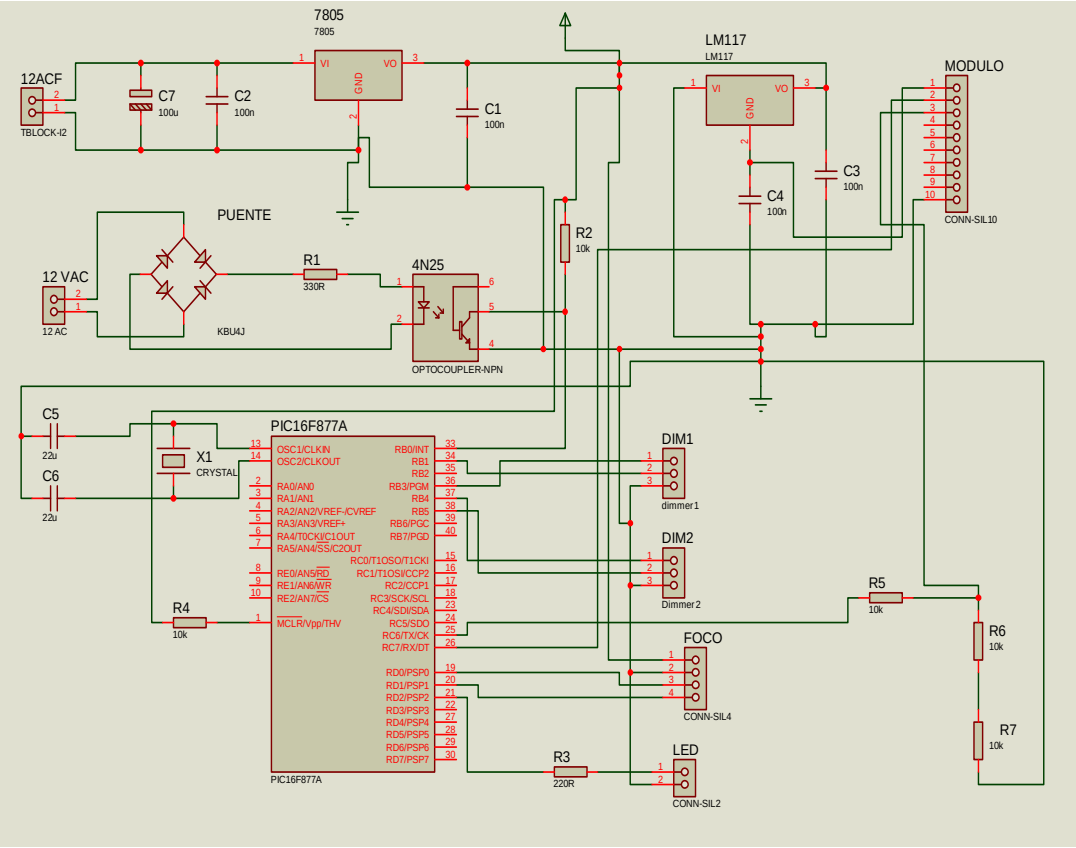


Figura IV.17. Diseño de el diagrama principal del control de luces.

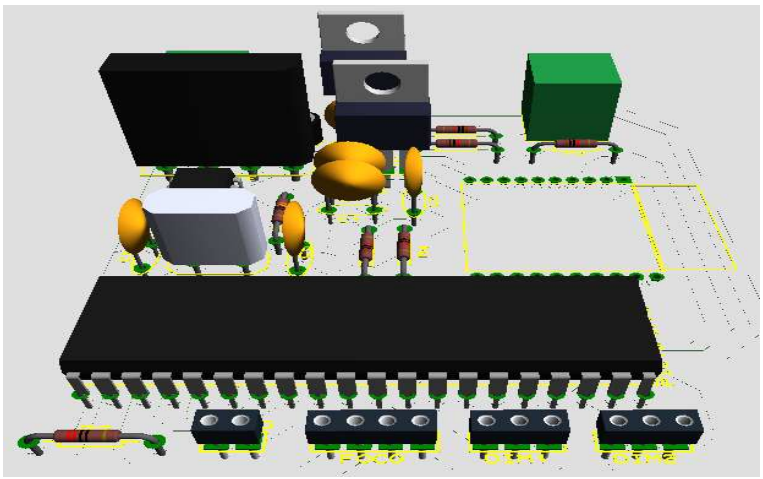


Figura IV.18. Distribución de la placa principal del control de luces.

El triac: es un interruptor de silicio de tres terminales que se puede disparar con impulsos positivos y negativos de la puerta cuando los potenciales del nodo son positivos o negativos respectivamente, es decir pueden conducir en las dos semiondas de la corriente alterna.

Optoacoplador: este tiene un triac el cual al ser disparado por la luz del diodo auto contenido produce el disparo.

del triac de potencia BT136 tanto en el semiciclo positivo como en el negativo. Haciendo variar la fase del disparo podemos regular la potencia en la carga.

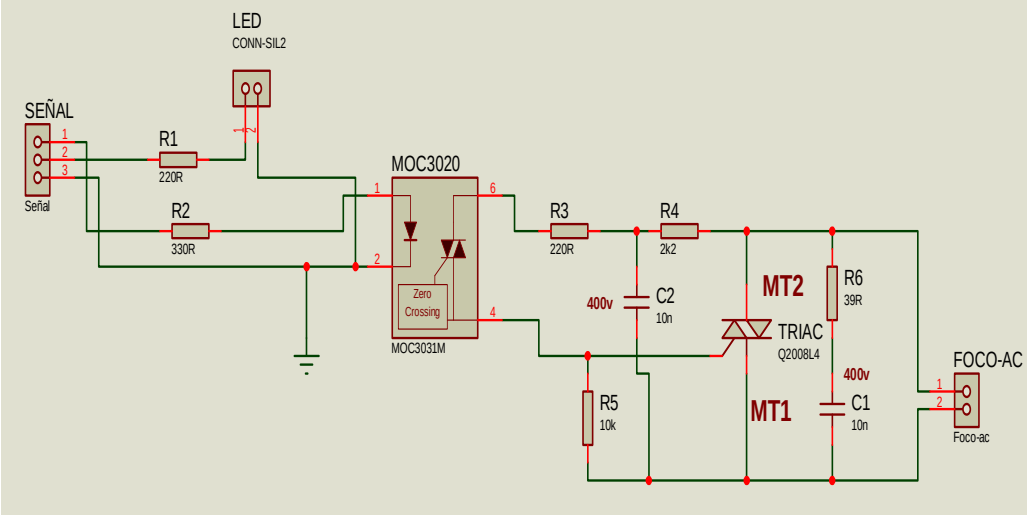


Figura IV.19. Diseño del circuito del Dimmer.

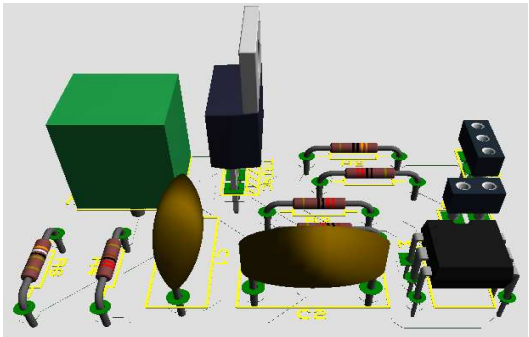


Figura IV.20. Distribución de la placa del circuito del Dimmer.

Luz Fija

En la figura IV.21 podemos observar el diagrama para encender y apagar una lámpara digitalmente, este proceso se realiza mediante un relé de 5 y su diseño real en la figura IV.22.

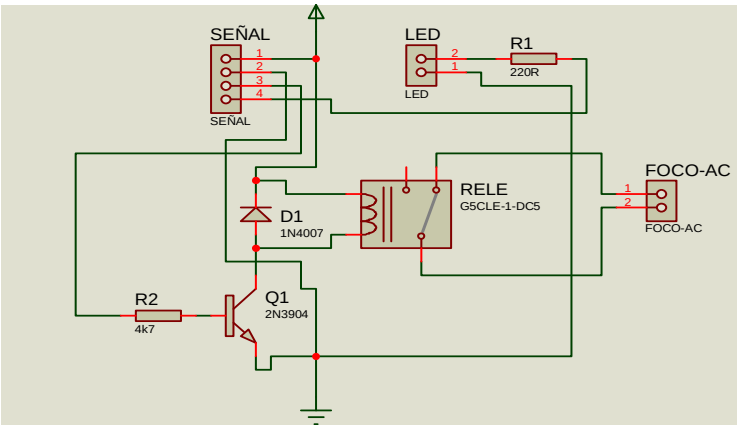


Figura IV.21. Diseño del circuito de encendido y apagado de una lámpara.

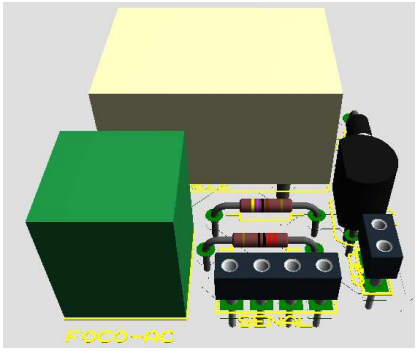


Figura IV.22. Distribución de la placa de la lámpara.

4.2.2.3. Ventanas y Cortinas

Para el control de las ventanas y cortinas también se utilizo el microcontrolador PIC16F7877A, la configuración del microcontrolador se la realizo igual que en control

de las luces, para recibir las ordenes se utilizo un modulo Xbee, el cual esta conectado y alimentado de la misma manera que en la figura IV.17.

Este circuito es el encargado de procesar las instrucciones recividas para todos los motores, en la figura IV.24 podemos ver como el microcontrolador a mas de conectarse al modulo Xbee, envía las ordenes a las siete placas de circuitos puente que son las encargadas de abrir y cerrar las ventanas y persianas.

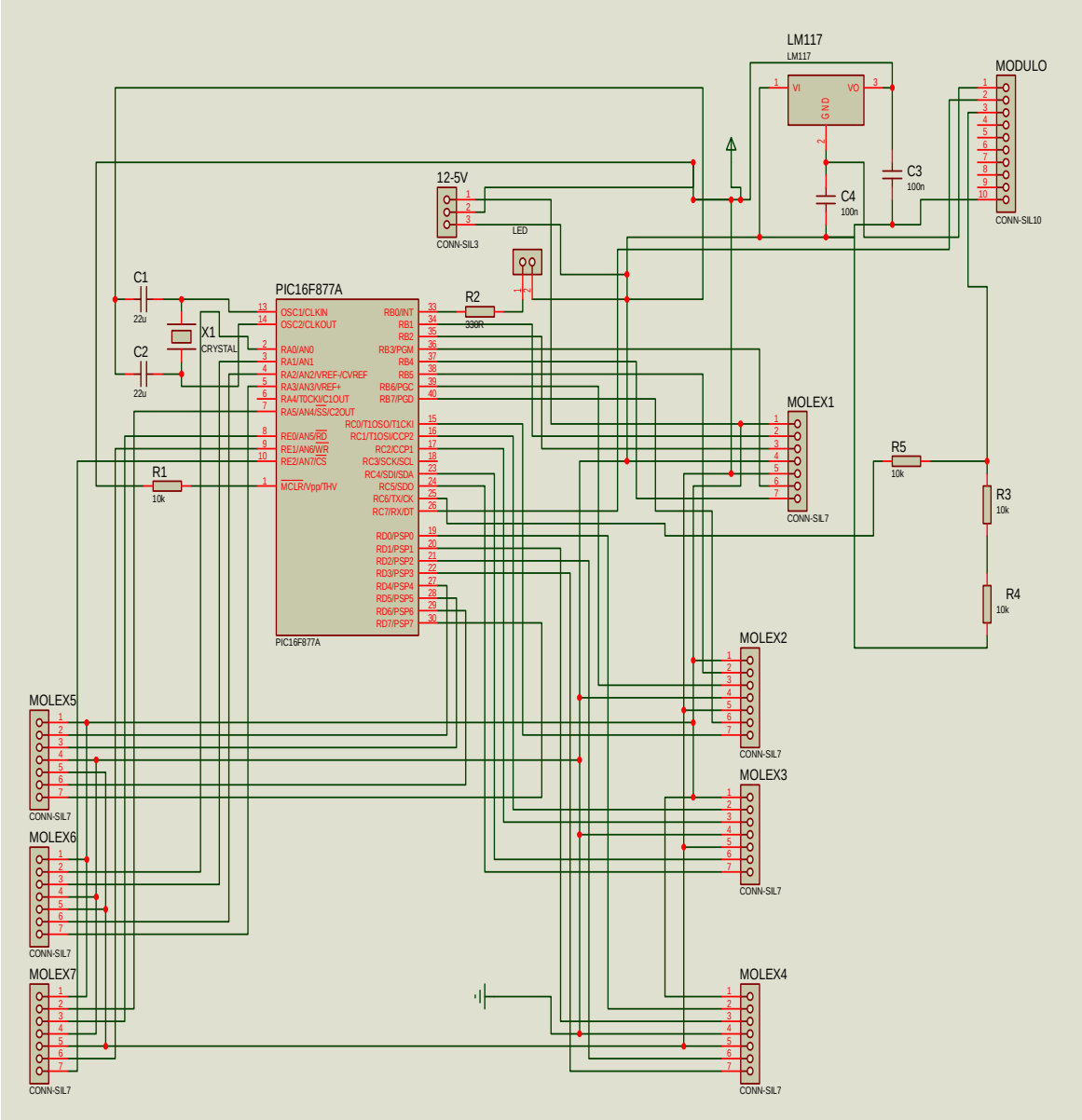


Figura IV.23. Diseño del circuito de control de motores.

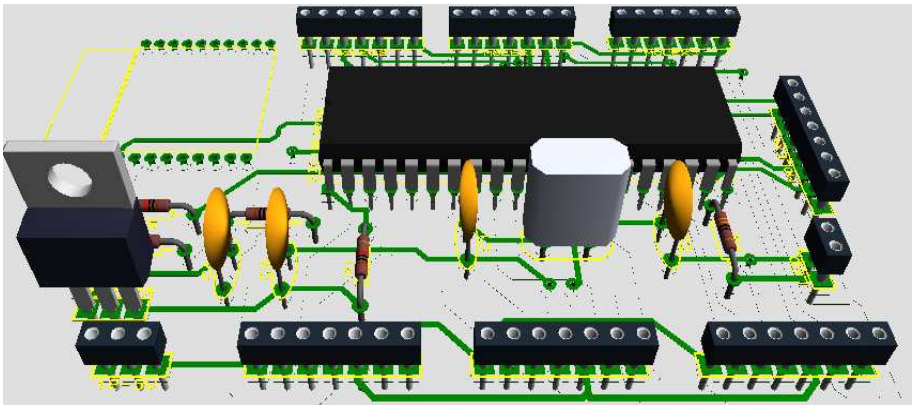


Figura IV.24. Distribución de la placa del control de motores.

Puente H

La finalidad de los puentes H es la de controlar el funcionamiento de los motores de nuestras ventanas y cortinas, ya que por medio de dos señales de control se controla el sentido de giro del motor.

En el puente H que se utilizó consiste de dos pares de transistores 2N3792 y MJ15015 por ser transistores de potencia y un par de transistores TIP122 que sirven como interruptores para los transistores de potencia, el diagrama de este puente H se observa en la figura IV.26 y posteriormente en la figura IV.27 se observa la vista real del puente H. Su funcionamiento es bastante simple, mientras el voltaje en las entradas sea 0 la corriente no puede circular por el motor ya que los dos transistores de la parte superior de la figura IV.26 no permiten el paso de corriente, cuando una entrada se alimenta con 5 V. y la otra se mantiene en 0 dos de los 4 transistores de potencia permiten que la corriente circule por el motor y llegue a tierra, dependiendo de la entrada que se esté alimentando con los 5 V. es el sentido por el que la corriente circula por el motor.

Cuando las dos entradas están alimentadas le llega voltaje a las dos entradas del motor pero la corriente no circula por que los transistores que la conducen a tierra permanecen abiertos.

Se usó este puente H debido a que es sencillo de hacer y porque es capaz de darle a nuestros motores suficiente corriente para moverse.

Transistor: Dispositivo electrónico de material semiconductor ya sea germanio o silicio, capaz de controlar una corriente eléctrica, amplificándola y/o conmutándola. Posee tres conexiones: Colector, Emisor y Base.

Motor de CC: Motor que requiere corriente continúa para su funcionamiento. Tiene la capacidad de girar en ambos sentidos, con sólo cambiar la polaridad de la alimentación. Es posible regular su velocidad haciendo circular mayor o menor corriente.

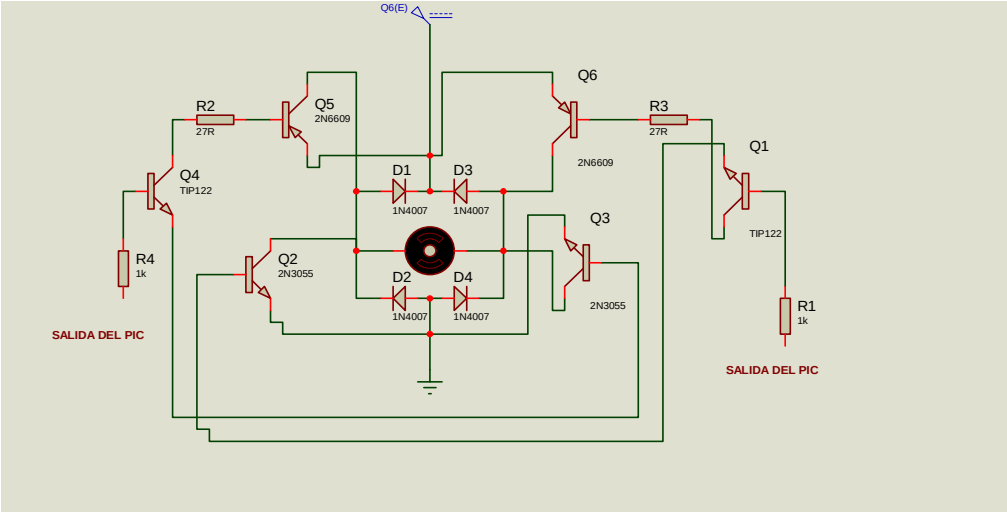


Figura IV.25. Diseño del Puente H.

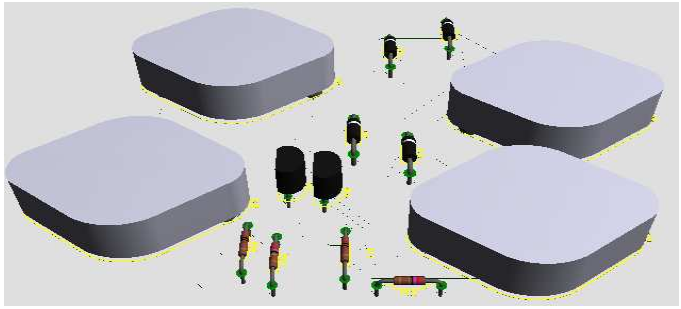


Figura IV.26. Distribución de la placa del Puente H.

4.2.3. Diseño Logico

Control Remoto

Para programar el control remoto se utilizaron funciones individuales para cada proceso como se puede ver en la figura IV.27.

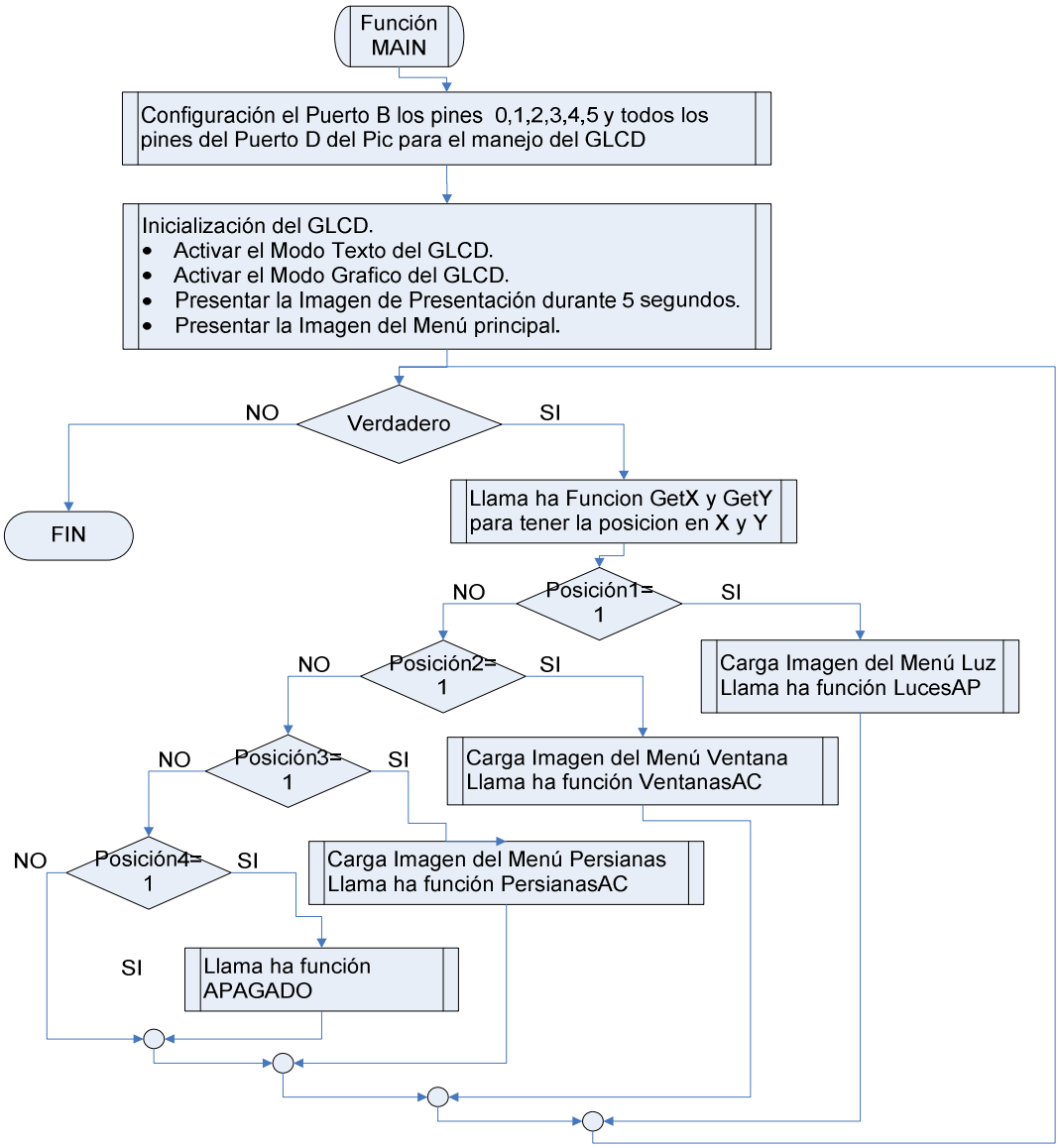


Figura IV.27. Diagrama de flujo del Control Remoto

En la función lucesAP como se puede ver en la figura IV.28 al presionar una posición del control remoto del menú luces, el pic lee las coordenadas y de acuerdo a estas se envía un carácter para que el receptor de las luces lo interprete.

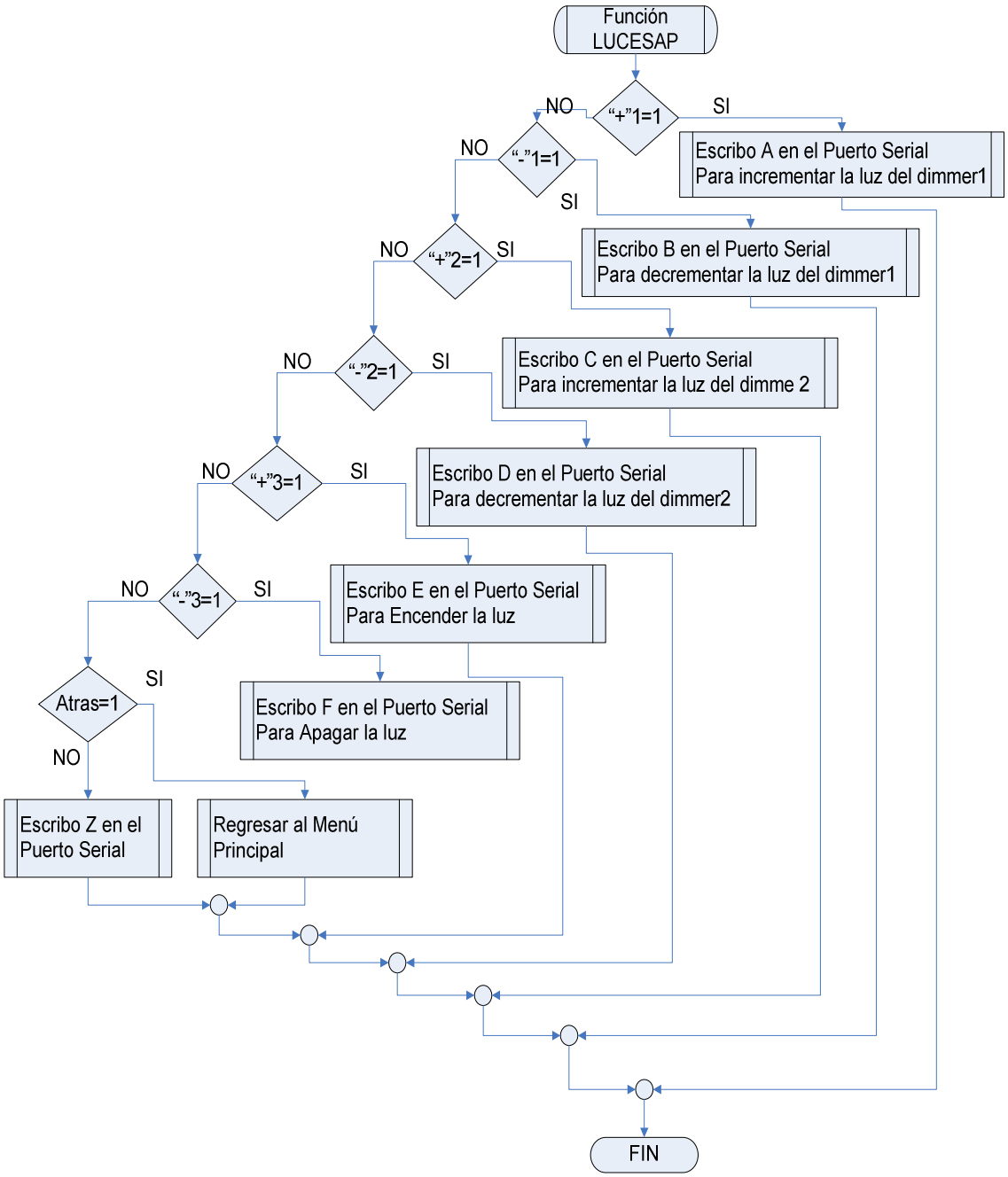


Figura IV.28. Diagrama de flujo Función LUCESAP

En la función VentanasAC como se puede ver en la figura IV.29 al presionar una posición del control remoto del menú ventanas, el pic lee las coordenadas y de acuerdo a estas se envía un carácter para que el receptor de las Ventanas lo interprete.

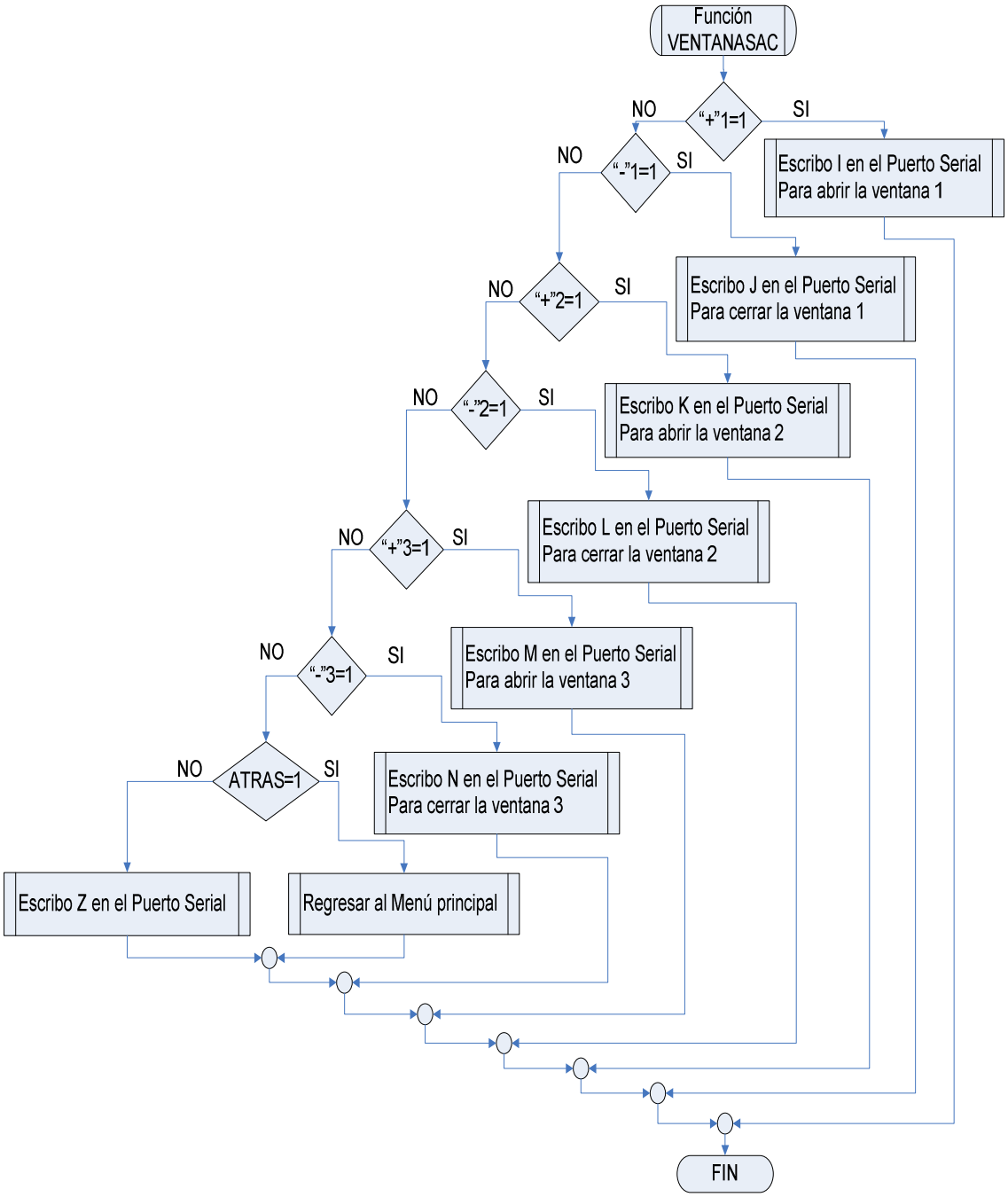


Figura IV.29. Diagrama de flujo Función VENTANASAP

En la función PersianasAC como se puede ver en la figura IV.30 al presionar una posición del control remoto del menú persianas, el pic lee las coordenadas y de

acuerdo a estas se envía un carácter para que el receptor de las persianas lo interprete.

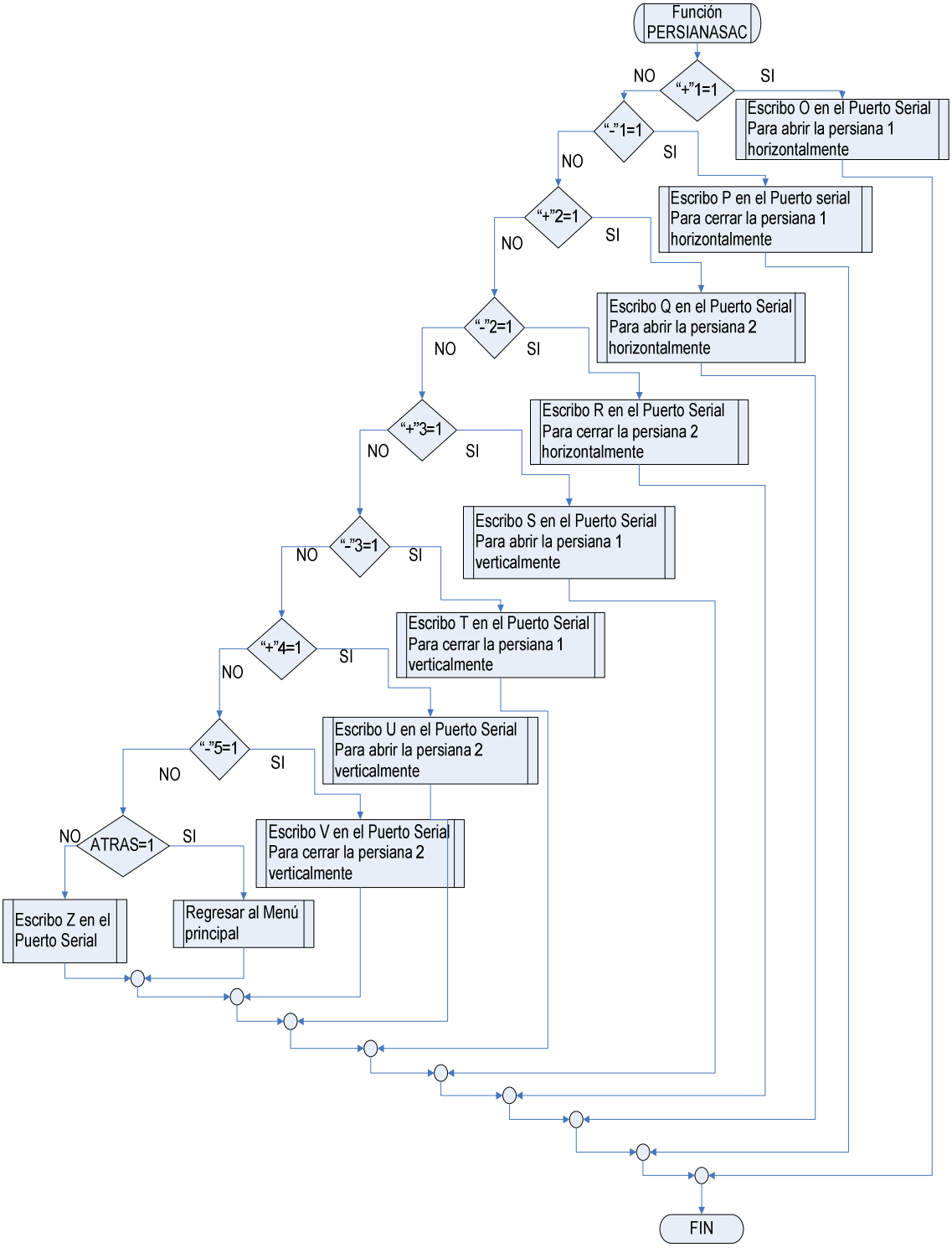


Figura IV.30. Diagrama de flujo Función PERSIANASAP

CAPÍTULO V

IMPLEMENTACIÓN DEL PROYECTO.

5.1. CONSTRUCCIÓN DEL PROYECTO

Para la implementación del proyecto se ha planificado de la siguiente manera:

- Implementación del diseño del control remoto.
- Implementación del diseño de control de luminosidad
- Implementación del diseño de control de potencia de motores.

5.1.1. Implementación del diseño del control remoto.

a) Para la implementación del control remoto posterior a la adquisición de los materiales se busco los datos técnicos de cada uno de los elementos para de esta manera procedes a la implementación como se muestra en la figura V.1.
b) Diseñar las distintas imágenes que se presentaran en los diferentes menús.

c) Mediante la utilización de la herramienta GLCD Bitmap Generator de MikroBasic como se puede ver en la figura V.2 transformamos las imágenes hechas en paint de extensión .bmp monocromáticas de 128x64pixels a código.

d) Una vez implementado el diseño electrónico como la transformación de las imágenes se procedió a realizar el código necesario para poder mostrar las imágenes en la pantalla del GLCD.

e) Ahora tomamos las posiciones en X y Y que se detectan en el touch que esta incorporado al GLCD.

f) Al tener las coordenadas de todas las posiciones programamos para que cuando se pulse cierta posición imagen envié un código por el modulo Xbee para que se ejecute la acción requerida.

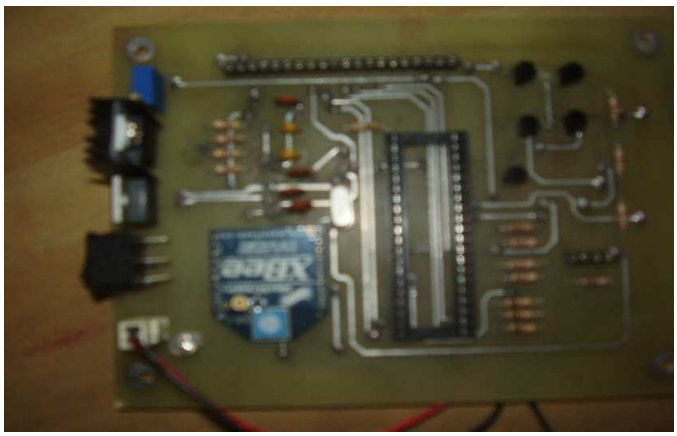


Figura V.1. Diseño del Control remoto

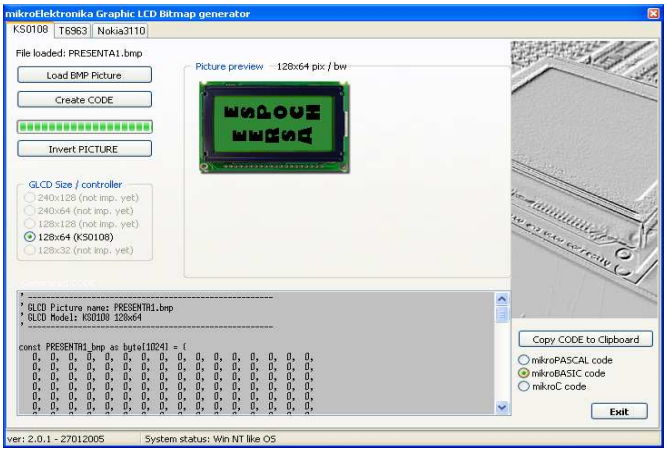


Figura V.2. GLCD Generator de MikroBasic

5.1.2. Implementación del diseño de control de luminosidad

a) Para poder controlar la variación de las luces tenemos que tener en cuenta que realizar el diseño del cruce por cero como se puede ver en la figura V.3.
b) Realizamos el diseño de los dos dimmers figura V.4., y del el encendido y apagado de la luz figura V.5.
c) Una vez que tenemos lista la parte electrónica procedemos a programar para esto hemos utilizado MicroCode.
d) El modulo Xbee recepta las órdenes y las transmite al pic para que este las ejecuta y se pueda variar la luminosidad del lugar.

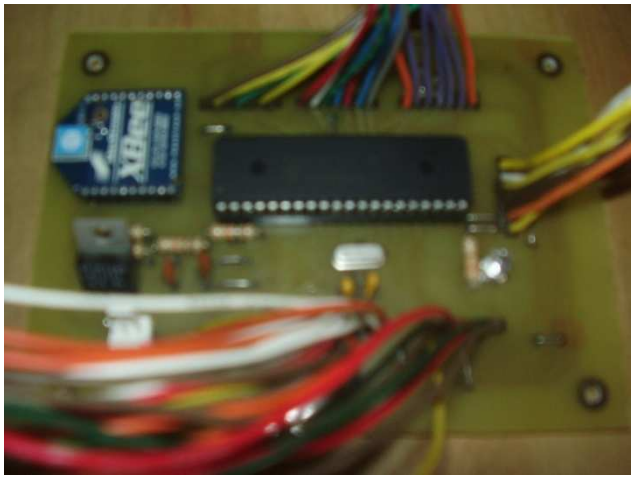


Figura V.3. Diseño del Cruce por cero.

Figura V.4. Diseño del Dimmer.

Figura V.5. Diseño del Onn-Off.

5.1.3. Implementación del diseño de control de potencia de motores.

a) Para poder controlar los siete motores de corriente continua de 12VAC que son los que estamos utilizando lo primero que diseñamos es el circuito puente con los transistores de potencia como se puede ver en la figura V.6., como son siete motores hemos diseñado siete placas similares una para cada motor, con su respectivo motor para su comunicación con la placa central, y su bornera para poder dar señales al motor.
b) Realizamos el diseño del de la placa que controle los motores como se puede ver en la figura V.7, esta es la encargada de enviar las señales a las placas para que de esta manera cada placa pueda hacer girar un motor en sentido horario o anti horario.
c) Ahora realizamos la programación del pic para esto hemos utilizado MicroCode.
d) Utilizamos un modulo Xbee configurado como receptor para transmitir al pic, y este pueda procese las ordenes y active el motor necesario para ejecutar las peticiones del usuario.

Figura V.6. Diseño del Cruce puente H.

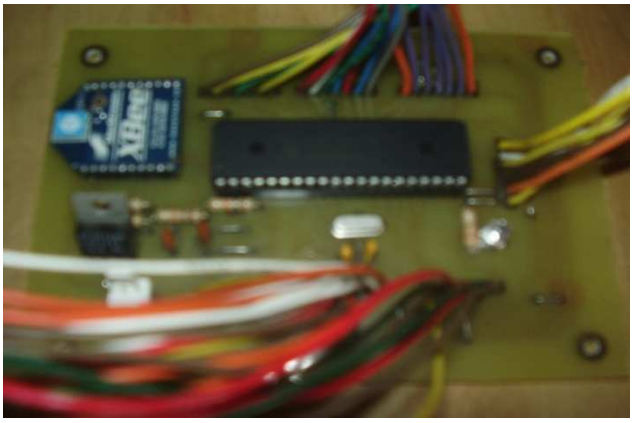


Figura V.7. Diseño de la placa de control de motores.

CAPÍTULO VI

PRUEBAS Y ANÁLISIS.

6.1. FUNCIONAMIENTO DEL PROYECTO

Durante las pruebas realizadas a los motores, luces, glcd y touch, se armó en el protoboard cada circuito con su respectivo microcontolador y módulo por medio del cual se envió las tramas correspondientes para la comunicación Xbee esta se realizo con el protocolo usart (asíncrona) del Pic con una velocidad de 9600 bps que es la velocidad por defecto que maneja el modulo Xbee.

Una vez que todos los circuitos y sus respectivas placas han sido conectados, encendemos las fuentes de alimentación de los receptores y se encenderá el switch del control remoto en el que se visualizara la pantalla de bienvenida, posteriormente el menú de control. A continuación se detalla las pruebas realizadas en el proyecto de la sala de conferencias de la Empresa Eléctrica Riobamba S.A. como se puede ver en la figura VI.1.

Figura VI.1. Sala de Conferencias Inteligente.

6.2. FUNCIONAMIENTO DEL CONTROL REMOTO

Empezaremos comprobando el correcto funcionamiento del control remoto que podemos ver en la figura VI.2, para esto lo primero que hicimos fue presionar cada opción del menú y verificar que nos envié al nuevo menú deseado, posteriormente la comunicación entre la placa transmisora y las placas receptoras, mediante el envío de tramas.

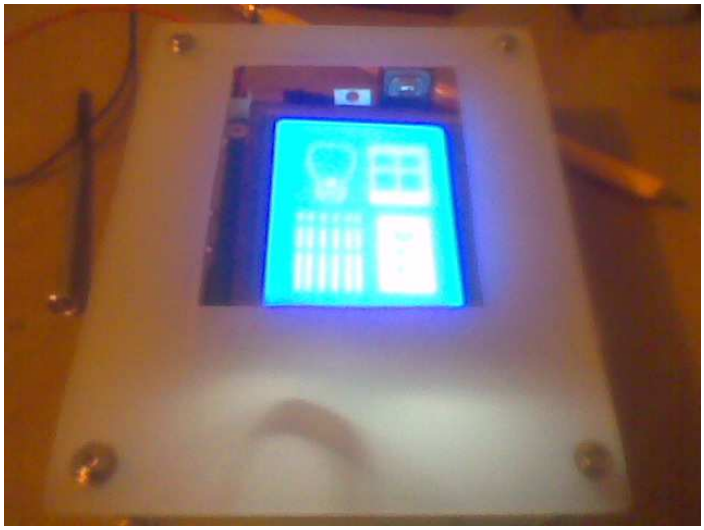


Figura VI.2. Control remoto.

6.3. MOVIMIENTO DE MOTORES

Durante las pruebas realizadas a los motores se verifico que cada motor se mueva en ambos sentidos para poder abrir y cerrar las ventanas o cortinas se utilizo el circuito llamado Puente H que permite el movimiento del motor dependiendo de la combinación de datos enviada por el Pic.

6.4. PRUEBAS DE LOS RECEPTORES

6.4.1. Pruebas de Motores de Cortinas y Ventanas.

Para probar los receptores de ventanas figura VI.3, persianas figura VI.4 y la persiana con sus respectivos motores en la figura VI.5, se procedió después de su respectiva instalación la programación con tiempos reales, para que de esta manera se puedan abrir y cerrar correctamente las ventanas y cortinas, y los motores no se atasquen.

A continuación se muestra las fotos de algunos movimientos de las ventanas y cortinas.

Figura VI.3. Ventanas.

Figura VI.4. Persianas.

Figura VI.5. Persianas con Motores.

6.4.2. Pruebas de Luces.

Para probar los receptores de las luces, después de su instalación se procedió a realizar las pruebas tanto en la luz fija figura VI.6 como la de los dimmers figura VI.7.

A continuación se muestra las fotos dos estados de iluminación de la sala de sesiones.

Figura VI.6. Luces Fijas.

Figura VI.7. Dimmers.

6.5. FUENTES DE ALIMENTACIÓN

La alimentación de los circuitos receptores se realizo mediante fuentes de alimentación y reguladores a 5 Voltios a 1 Amperio como se puede ver en la figura VI.1 y para los motores 12 Voltios a 3 Amperios, el control remoto se alimento con una batería de 9 Voltios que por medio de un regulador se tuvo los 5 Voltios, para la alimentación de los módulos se utilizo reguladores de 3.3 Voltios.

Figura VI.8. Fuente de Alimentación de circuito receptor.

CONCLUSIONES.

1. Los Módulos ZigBee son muy versátiles, pues pueden ser utilizados en muchas aplicaciones en las cuales el uso de cables sea un impedimento para montar una red, además trabajan de manera excelente en aquellas situaciones en las que no se tengan flujos de información grandes, sino más bien se envíen o reciban datos muy puntuales cada cierto tiempo
2. Este Proyecto sirvió para comprobar que ZigBee es un estándar de comunicación inalámbrico válido para la Domótica, pues para un Edificio Estándar se pueden cubrir todos los puntos distantes dentro de una misma red e incluso con un solo dispositivo remoto; el único condicionante sería el número de elementos que un solo proyecto logre controlar.
3. En el presente Proyecto, la mejor forma para el Modulo de mando XBEE se comunique con los receptores fue el Modo de conexión Punto a Punto, pero es importante mencionar que se utilizo un Modo Broadcast, donde el Dispositivo Base transmite su señal a todos los Remotos conectados a la Red, para que sean éstos los que validen o descarten la información recibida.
4. Debido a que los Módulos emiten señales RF en una frecuencia ISM o de libre acceso (2.4 GHz), el uso de aparatos tales como teléfonos inalámbricos afectarán el desempeño del Sistema, pues se podrían presentar pérdidas datos o interferencia de información.
5. Uno de los beneficios tangibles que se puede obtener de este Proyecto, es el hecho de que ZigBee mejora la comunicación entre dispositivos tomando en cuenta las distancias entre ellos, pues si comparamos estas longitudes con respecto a Bluetooth, está claro que se pueden cubrir más habitaciones con menos dispositivos.

6. El control de intensidad de luz implementado en este proyecto, es un servicio que contribuye a la comodidad del usuario, ya que éste puede disponer de la intensidad luminosa de acuerdo a sus necesidades, si se llegase al caso de implementar comercialmente el dispositivo, este factor extra resultaría atractivo a posibles interesados de esta idea.
7. Las antenas que poseen los módulos ZigBee deben permanecer completamente descubiertas y erguidas para obtener resultados favorables tanto en la transmisión como en la recepción, pues si éstas se encuentran cubiertas con objetos extraños el nivel de señal no sería el mismo y la información recibida o transmitida sería incorrecta, causando fallos en el funcionamiento del sistema.
8. El estándar ZigBee ofrece una baja complejidad y una baja exigencia de recursos. Permite operar en tres bandas de frecuencia no licenciadas, lo cual reduce los costos de operación e instalación, con diversas topologías de red y puede trabajar interactuando con otras tecnologías que funcionan sobre dichas bandas.
9. Para los dispositivos finales el Estándar ZigBee tiene la capacidad de entrar en modo “pasivo” (idle) y en modo “desactivado” (sleep), consumiendo poca energía; lo cual hace diferencia con otros protocolos de redes inalámbricas que no poseen estas características.
10. El microcontrolador PIC16F877A y el PIC18f4550 es un elemento resistente y de fácil programación. Estas características nos permiten concluir que al momento de utilizarlo como elemento de procesamiento de datos provee a nuestro sistema anti intrusos la robustez necesaria para cumplir con las tareas encomendadas.
11. Para el detector de cruces por cero es necesario que la señal analogía de entrada se a menor a 9VAC, ya que el puerto o el PIC podrían sufrir averías.

RECOMENDACIONES

1. Los productos ZigBee irradian energía en la banda de la microonda. Aunque los niveles se consideran bajos menor que 2mW, por lo que es necesario proteger al módulo de un posible daño por impacto del campo electromagnético. Cuando entra en funcionamiento los módulos, se debe evitar tocar la antena y la tarjeta misma.
2. Es recomendable, para un proyecto donde se ha diseñado y elaborado una placa que incluya circuitos integrados susceptibles de cambios como microprocesadores, etc., como en el presente caso, el uso de zócalos con el número de pines correspondiente al elemento que va a ser colocado; de esta forma se podrán realizar posibles cambios o incluso reemplazar un elemento dañado de manera pronta y sin tener que causar perjuicios a la fibra de vidrio que pudiesen acarrear alguna consecuencia negativa en el desarrollo final y por tanto en los resultados deseados.
3. Antes de tomar alguna decisión en cuanto a la manera como se van a conectar los elementos de un sistema, es importante realizar las pruebas necesarias con el fin de observar el comportamiento del diseño y determinar aquella que convenga más a los requerimientos del proyecto.
4. Es importante, antes de utilizar un dispositivo de características parecidas a los Módulos XBEE empleados en el presente Proyecto, determinar en su Datasheet las características de los voltajes a los cuales se comunica con elementos externos tecnología CMOS o TTL, pues de esta forma se evitaría el riesgo de dañar los elementos con voltajes incorrectos de transmisión o recepción.
5. Se debe determinar si las fuentes de poder que se vayan a utilizar en un circuito serán capaces de soportar la carga a las que van a ser sometidas,

pues en algunas oportunidades este aspecto podría no cumplirse, lo que ocasionaría un funcionamiento deficiente de los elementos que formen parte del diseño así como del circuito propiamente dicho, debido a voltajes de polarización inadecuados.

6. Para hacer uso de los Módulos XBEE es necesario diseñar una fuente de alimentación muy confiable y precisa, pues los dispositivos ZigBee son sensibles a fallas de voltaje y pueden resultar averiados de forma permanente, lo aconsejable es el uso regulador de 3.3V para asegurar un voltaje constante.
7. Se recomienda ubicar los dispositivos Xbee de tal forma que puedan mantener línea de vista y de esta forma garantizar que las comunicaciones sean eficientes.

RESUMEN.

El presente proyecto ha sido implementado en la sala de conferencias de gerencia de la Empresa Eléctrica Riobamba S.A. Se ha automatizado el control de persianas, ventanas y luces a través de un solo comando, utilizando tecnología ZigBee. Esta automatización permite dar mayor confort y facilidad en el manejo de la sala.

Se cuenta con un módulo que trabaja como transmisor en un control remoto controlado con microcontrolador 18F4550, y módulos como receptores controlados con microcontroladores 16F877A.

La comunicación se realiza con protocolos inalámbricos con estándar IEEE 802.15.4 Zigbee, como aplicación domótica.

Fue reemplazado el medio manual por medio digital, es decir a través de control remoto táctil con diferentes menús que puede encender y apagar la luz fija y los dimmers, también se puede abrir y cerrar ventanas y persianas con dos movimientos, de 90° y recorrido completo.

La tecnología Zigbee, que fue implementada permitió manejar pequeñas cantidades de datos, bajo consumo de potencia, bajo costo y fácil manejo en la buena ejecución del control domótico de varios objetos (ventanas, persianas, luces) de la sala de conferencias. Se recomienda realizar mantenimiento periódico asegurando su correcto funcionamiento.

SUMMARY

The present project has been implemented for the conference room of the Empresa Electrica Riobamba S.A. It has automated the control of blinds, windows and lights through a single command, using ZigBee technology. This automation allows to give major comfort and facility in the managing of the room.

It has a module that works as transmitter in a remote control controlled with a microcontroller 18F4550, and modules as receivers controlled with microcontrollers 16F877A.

The communication is done using wireless protocols with standard IEEE 802.15.4 Zigbee, as a domotic application.

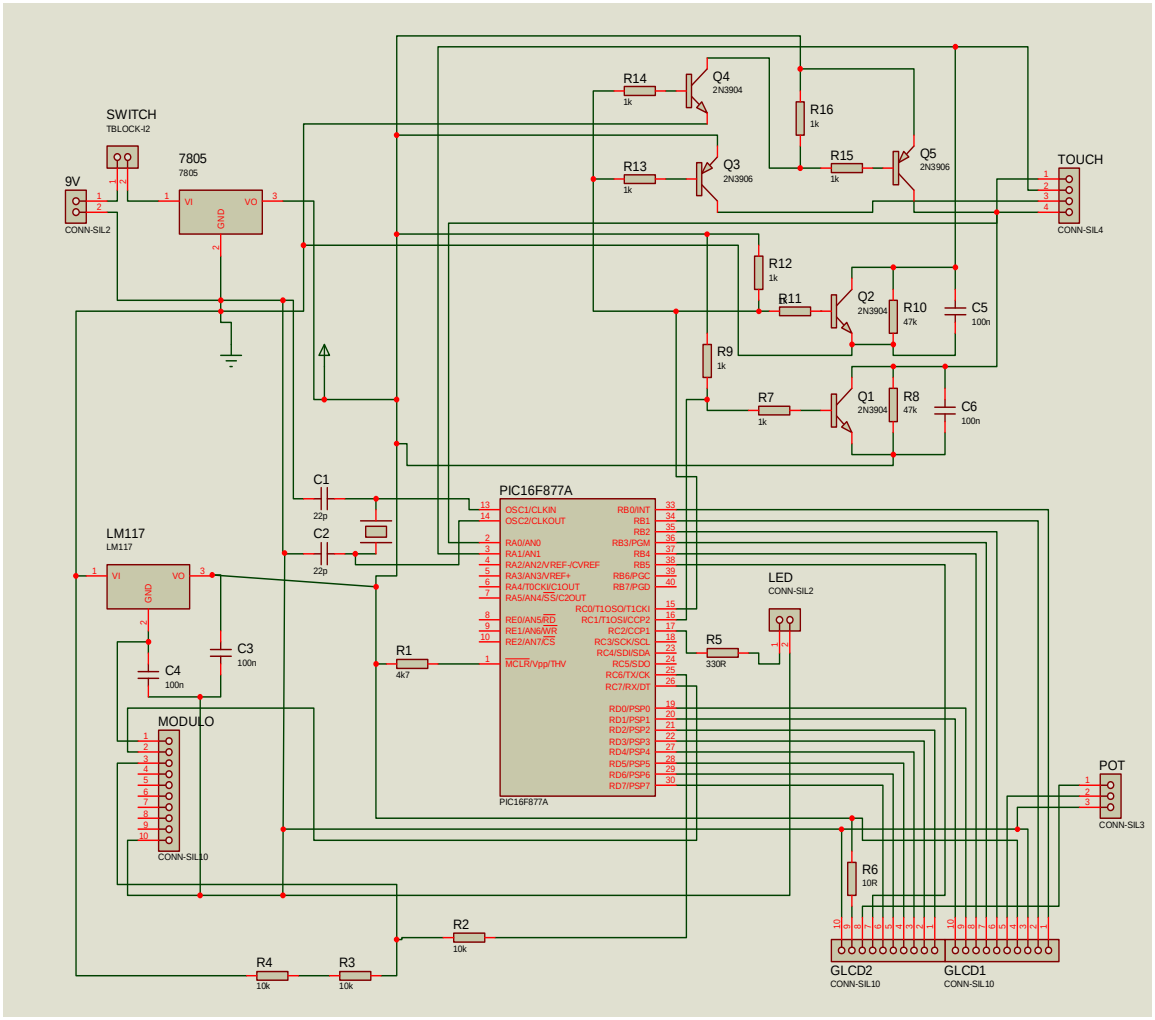
Manual medium was replaced by digital, through a tactile remote control with different menus that can to turn on or off fixed lights and the dimmers, it can also to open and close windows and blinds with two movements of 90° and complete path.

The Zigbee technology, that was implemented allows to handle small quantities of information, low power consumption, low cost, easy handling and a good domotic control execution for several objects (windows, blinds, lights) of conference room. It is recommended to realize periodic maintenance, assuring his correct functioning.

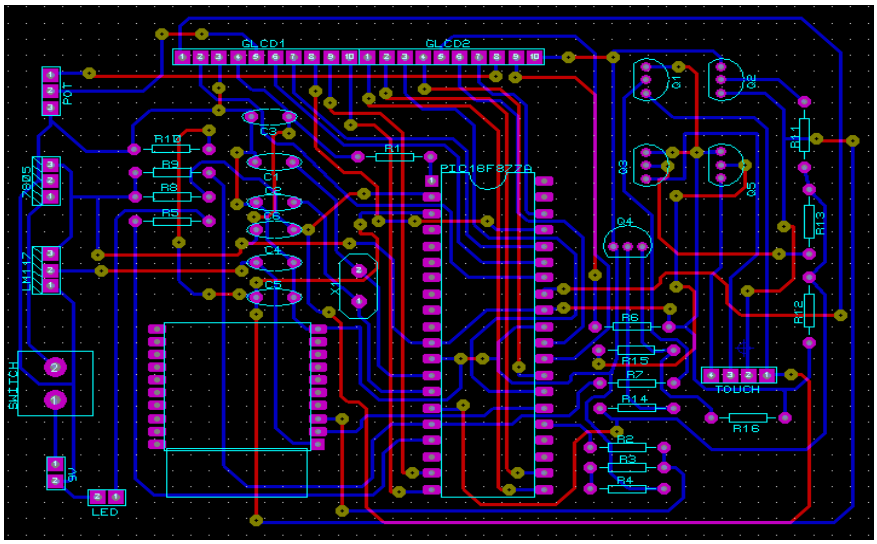
ANEXOS

Anexo A. Diagrama y circuito impreso del control remoto

- Diagrama del circuito

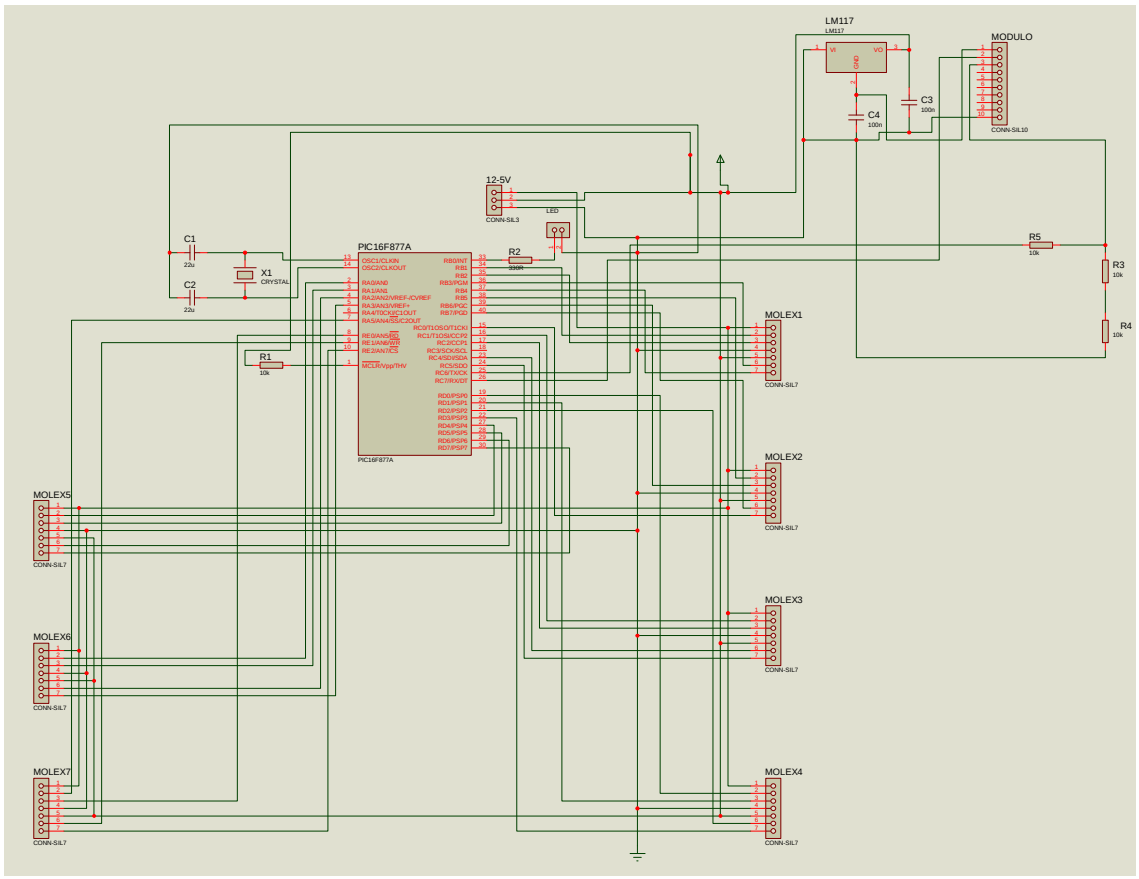


- Circuito impreso

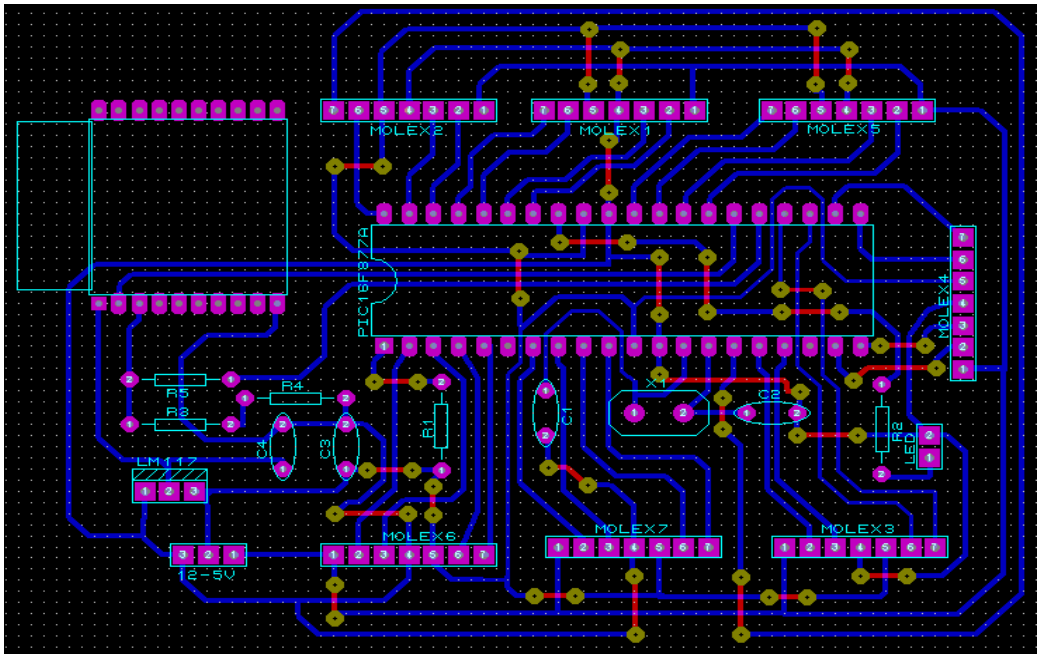


Anexo B. Diagrama y circuito impreso del control de motores

- Diagrama del circuito

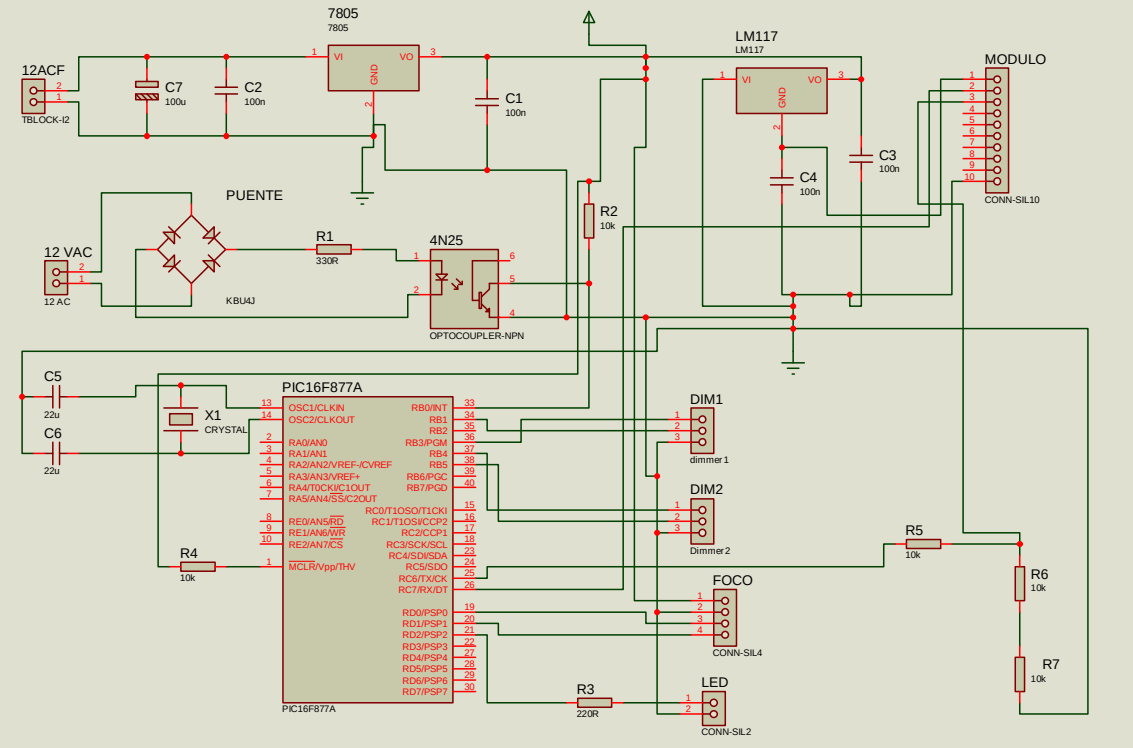


- Circuito Impreso

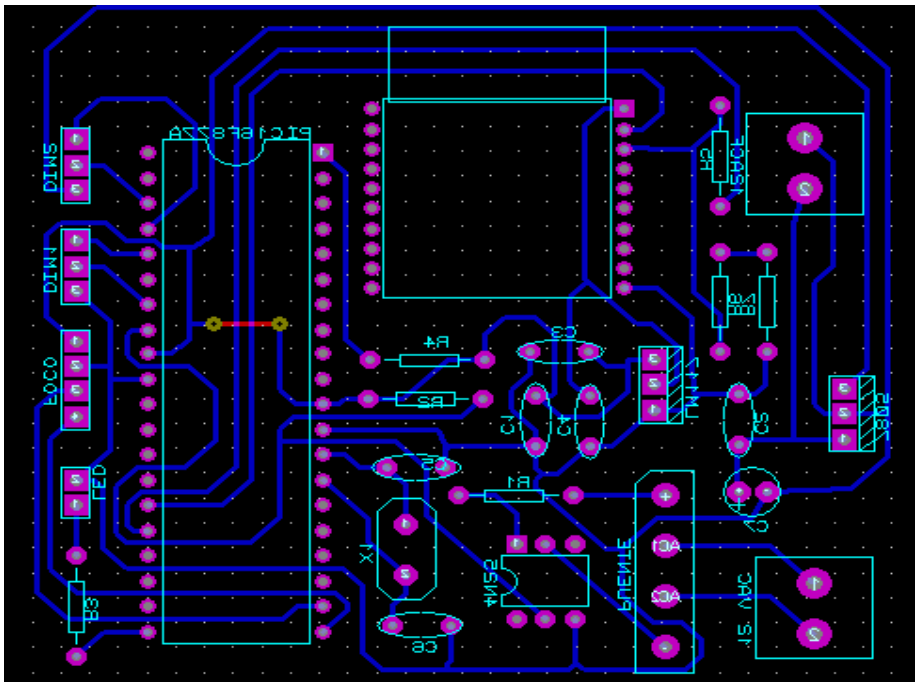


Anexo C. Diagrama y circuito impreso del control de luces

- Diagrama del circuito



- Circuito Impreso



Anexo D. Datasheet del PIC 16F877A

PIC16F87X
Data Sheet

28/40-Pin 8-Bit CMOS FLASH

Microcontrollers

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

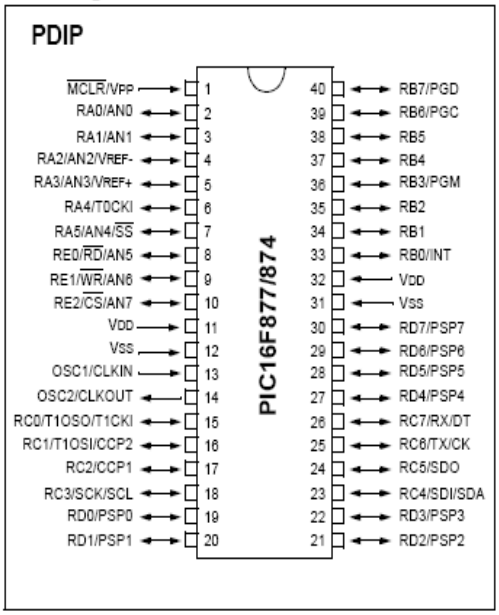
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- 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 the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram

PDIP



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
- 10-bit multi-channel Analog-to-Digital converter
- 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 RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Instruction Set	35 instructions	35 instructions	35 instructions	35 instructions

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	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.
$\overline{MCLR}$/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/ counter. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be soft-ware programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming 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.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
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.

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 2-1.

The Special Function Registers can be classified into two sets: core (CPU) and peripheral. Those registers associated with the core functions are described in detail in this section. Those related to the operation of the peripheral features are described in detail in the peripheral features section.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 0											
00h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
01h	TMR0	Timer0 Module Register								xxxx xxxx	47
02h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
03h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
04h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	29
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	31
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	33
08h(4)	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	35
09h(4)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	36
0Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
0Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	22
0Dh	PIR2	—	(5)	—	EEIF	BCLIF	—	—	CCP2IF	-r-0 0--0	24
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	51
11h	TMR2	Timer2 Module Register								0000 0000	55
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	55
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	70, 73
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	67
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	57
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	57
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	58
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	96
19h	TXREG	USART Transmit Data Register								0000 0000	99
1Ah	RCREG	USART Receive Data Register								0000 0000	101
1Bh	CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	57
1Ch	CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	57
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	58
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	116
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	—	ADON	0000 00-0	111

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 1											
80h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	19
82h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
83h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
84h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	29
86h	TRISB	PORTB Data Direction Register								1111 1111	31
87h	TRISC	PORTC Data Direction Register								1111 1111	33
88h(4)	TRISD	PORTD Data Direction Register								1111 1111	35
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	37
8Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
8Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	21
8Dh	PIE2	—	(5)	—	EEIE	BCLIE	—	—	CCP2IE	-r-0 0--0	23
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	---- --qq	25
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	68
92h	PR2	Timer2 Period Register								1111 1111	55
93h	SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	73, 74
94h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	66
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	95
99h	SPBRG	Baud Rate Generator Register								0000 0000	97
9Ah	—	Unimplemented								—	—
9Bh	—	Unimplemented								—	—
9Ch	—	Unimplemented								—	—
9Dh	—	Unimplemented								—	—
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	116
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	0--- 0000	112

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

2.2.2.1 STATUS Register

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

The STATUS register can be the destination for any instruction, as with any other register. 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 the 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).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect the Z, C or DC bits from the STATUS register. For other instructions not affecting any status bits, see the "Instruction Set Summary."

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

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

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 **IRP:** Register Bank Select bit (used for indirect addressing)
1 = Bank 2, 3 (100h - 1FFh)
0 = Bank 0, 1 (00h - FFh)

bit 6-5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
11 = Bank 3 (180h - 1FFh)
10 = Bank 2 (100h - 17Fh)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)
Each bank is 128 bytes

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)
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: For $\overline{borrow}$, the polarity is reversed. 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

2.2.2.2 OPTION_REG Register

The OPTION_REG Register is a readable and writable register, which contains various control bits to configure the TMR0 prescaler/WDT postscaler (single assignable register known also as the prescaler), the External INT Interrupt, TMR0 and the weak pull-ups on PORTB.

Note:	To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

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

Note: When using low voltage ICSP programming (LVP) and the pull-ups on PORTB are enabled, bit 3 in the TRISB register must be cleared to disable the pull-up on RB3 and ensure the proper operation of the device

2.2.2.3 INTCON Register

The INTCON Register is a readable and writable register, which contains various enable and flag bits for the TMR0 register overflow, RB Port change and External RB0/INT pin interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **T0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

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

2.2.2.4 PIE1 Register

The PIE1 register contains the individual enable bits for the peripheral interrupts.

Note:	Bit PEIE (INTCON<6>) must be set to enable any peripheral interrupt.

REGISTER 2-4: PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE(1):** Parallel Slave Port Read/Write Interrupt Enable bit
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D converter interrupt
0 = Disables the A/D converter interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE**: Synchronous Serial Port Interrupt Enable bit
1 = Enables the SSP interrupt
0 = Disables the SSP interrupt

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Note 1: PSPIE is reserved on PIC16F873/876 devices; always maintain this bit clear.

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

2.2.2.5 PIR1 Register

The PIR1 register contains the individual flag bits for the peripheral interrupts.

Note:	Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt bits are clear prior to enabling an interrupt.

REGISTER 2-5: PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **PSPIF(1):** Parallel Slave Port Read/Write Interrupt Flag bit
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

bit 6 **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion completed
0 = The A/D conversion is not complete

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer is full
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty
0 = The USART transmit buffer is full

bit 3 **SSPIF**: Synchronous Serial Port (SSP) Interrupt Flag
1 = The SSP interrupt condition has occurred, and must be cleared in software before returning from the Interrupt Service Routine. The conditions that will set this bit are:
- SPI
  - A transmission/reception has taken place.
- I^2C Slave
  - A transmission/reception has taken place.
- I^2C Master
  - A transmission/reception has taken place.
  - The initiated START condition was completed by the SSP module.
  - The initiated STOP condition was completed by the SSP module.
  - The initiated Restart condition was completed by the SSP module.
  - The initiated Acknowledge condition was completed by the SSP module.
  - A START condition occurred while the SSP module was idle (Multi-Master system).
  - A STOP condition occurred while the SSP module was idle (Multi-Master system).

0 = No SSP interrupt condition has occurred.

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode

bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Note 1: PSPIF is reserved on PIC16F873/876 devices; always maintain this bit clear.

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

2.2.2.6 PIE2 Register

The PIE2 register contains the individual enable bits for the CCP2 peripheral interrupt, the SSP bus collision interrupt, and the EEPROM write operation interrupt.

REGISTER 2-6: PIE2 REGISTER (ADDRESS 8Dh)

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	Reserved	—	EEIE	BCLIE	—	—	CCP2IE
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6 **Reserved**: Always maintain this bit clear

bit 5 **Unimplemented:** Read as '0'

bit 4 **EEIE**: EEPROM Write Operation Interrupt Enable
1 = Enable EE Write Interrupt
0 = Disable EE Write Interrupt

bit 3 **BCLIE**: Bus Collision Interrupt Enable
1 = Enable Bus Collision Interrupt
0 = Disable Bus Collision Interrupt

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **CCP2IE**: CCP2 Interrupt Enable bit
1 = Enables the CCP2 interrupt
0 = Disables the CCP2 interrupt

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

3.0 I/O PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Additional information on I/O ports may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

3.1 PORTA and the TRISA Register

PORTA is a 6-bit wide, bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, the value is modified and then written to the port data latch.

Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open drain output. All other PORTA pins have TTL input levels and full CMOS output drivers.

Other PORTA pins are multiplexed with analog inputs and analog VREF input. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register1).

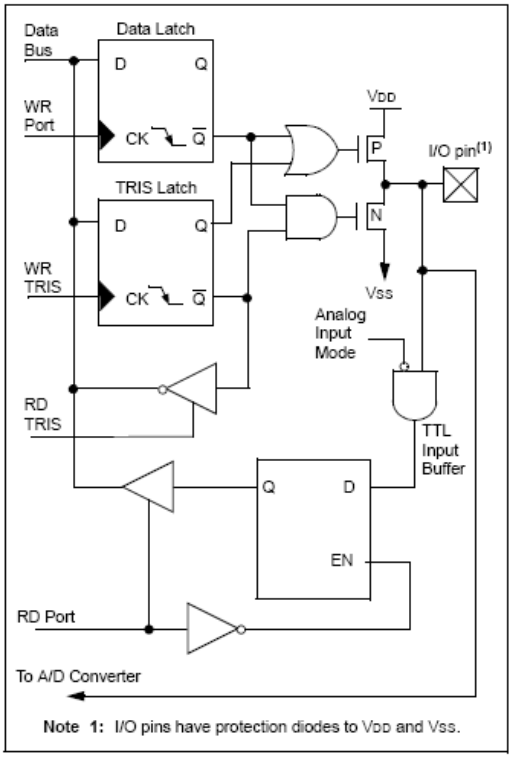
Note:	On a Power-on Reset, these pins are configured as analog inputs and read as '0'.

The TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

EXAMPLE 3-1: INITIALIZING PORTA

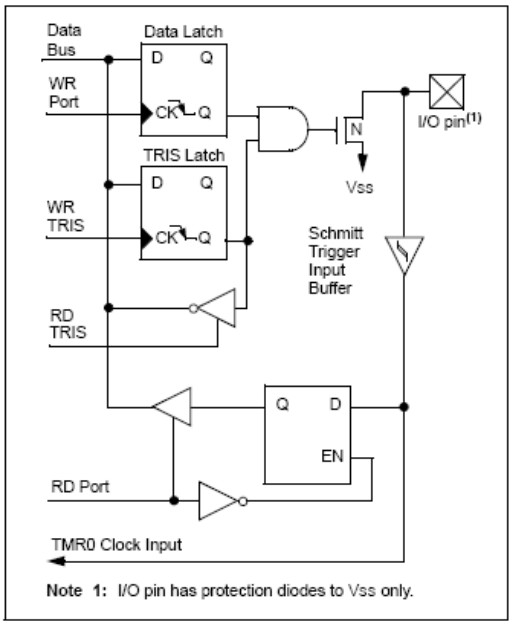
```
BCF     STATUS, RP0   ;
BCF     STATUS, RP1   ; Bank0
CLRF    PORTA         ; Initialize PORTA by
                      ; clearing output
                      ; data latches
BSF     STATUS, RP0   ; Select Bank 1
MOVLW   0x06          ; Configure all pins
MOVWF   ADCON1        ; as digital inputs
MOVLW   0xCF          ; Value used to
                      ; initialize data
                      ; direction
MOVWF   TRISA         ; Set RA<3:0> as inputs
                      ; RA<5:4> as outputs
                      ; TRISA<7:6>are always
                      ; read as '0'.
```

FIGURE 3-1: BLOCK DIAGRAM OF RA3:RA0 AND RA5 PINS



Note 1: I/O pins have protection diodes to VDD and Vss.

FIGURE 3-2: BLOCK DIAGRAM OF RA4/T0CKI PIN



Note 1: I/O pin has protection diodes to Vss only.

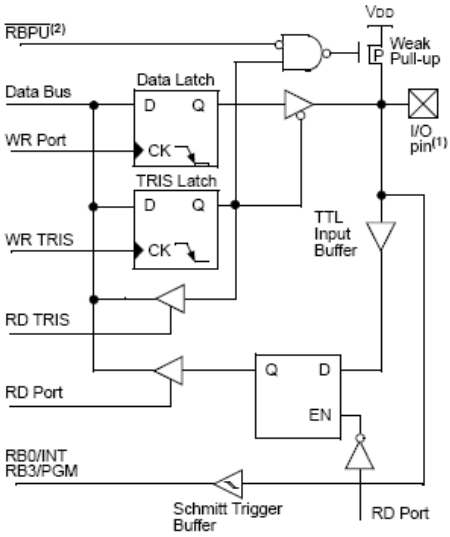
3.2 PORTB and the TRISB Register

PORTB is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

Three pins of PORTB are multiplexed with the Low Voltage Programming function: RB3/PGM, RB6/PGC and RB7/PGD. The alternate functions of these pins are described in the Special Features Section.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit RBPU (OPTION_REG<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

FIGURE 3-3: BLOCK DIAGRAM OF RB3:RB0 PINS



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (OPTION_REG<7>).

Four of the PORTB pins, RB7:RB4, have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON<0>).

This interrupt can wake the device from SLEEP. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB. This will end the mismatch condition.
b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

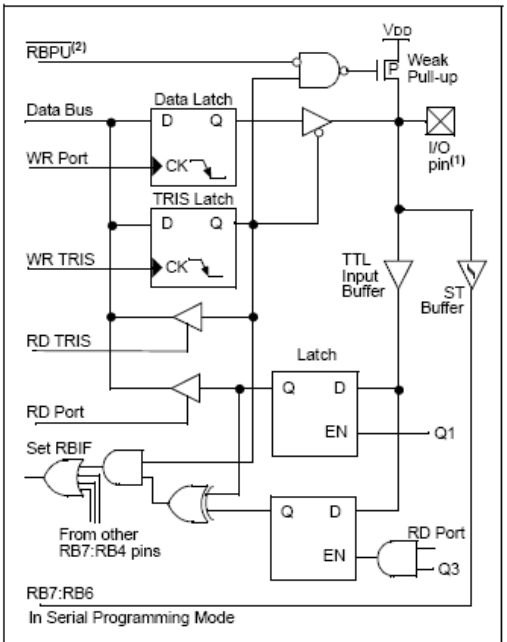
The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

This interrupt-on-mismatch feature, together with software configureable pull-ups on these four pins, allow easy interface to a keypad and make it possible for wake-up on key depression. Refer to the Embedded Control Handbook, *"Implementing Wake-up on Key Strokes"* (AN552).

RB0/INT is an external interrupt input pin and is configured using the INTEDG bit (OPTION_REG<6>).

RB0/INT is discussed in detail in Section 12.10.1.

FIGURE 3-4: BLOCK DIAGRAM OF RB7:RB4 PINS



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (OPTION_REG<7>).

TABLE 3-3: PORTB FUNCTIONS

Name	Bit#	Buffer	Function
RB0/INT	bit0	TTL/ST(1)	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3/PGM(3)	bit3	TTL	Input/output pin or programming pin in LVP mode. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5	bit5	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB6/PGC	bit6	TTL/ST(2)	Input/output pin (with interrupt-on-change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming clock.
RB7/PGD	bit7	TTL/ST(2)	Input/output pin (with interrupt-on-change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming data.

Legend: 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: Low Voltage ICSP Programming (LVP) is enabled by default, which disables the RB3 I/O function. LVP must be disabled to enable RB3 as an I/O pin and allow maximum compatibility to the other 28-pin and 40-pin mid-range devices.

TABLE 3-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
81h, 181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

3.3 PORTC and the TRISC Register

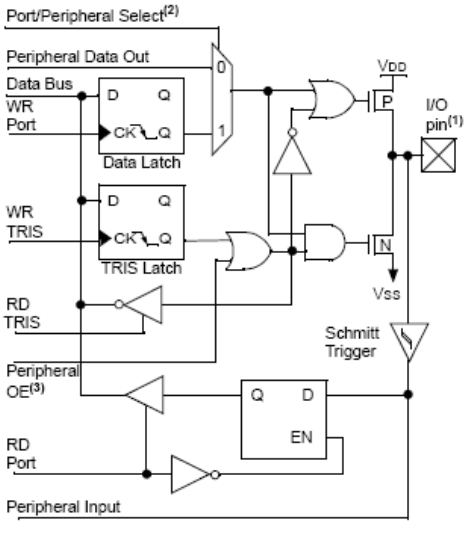
PORTC is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

PORTC is multiplexed with several peripheral functions (Table 3-5). PORTC pins have Schmitt Trigger input buffers.

When the I^2C module is enabled, the PORTC<4:3> pins can be configured with normal I^2C levels, or with SMBus levels by using the CKE bit (SSPSTAT<6>).

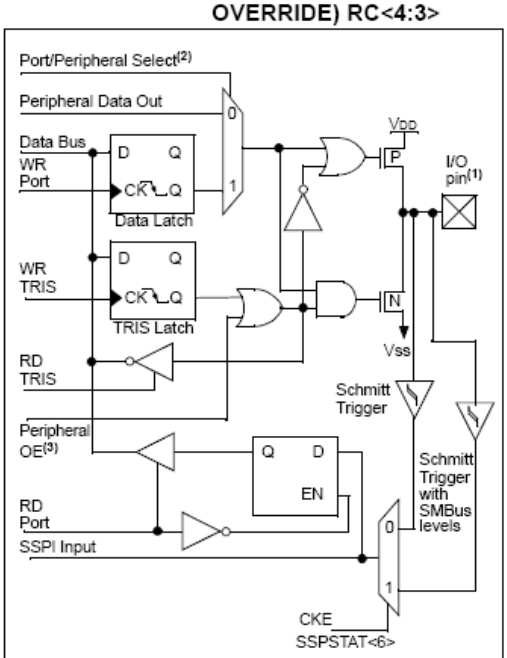
When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. Since the TRIS bit override is in effect while the peripheral is enabled, read-modify-write instructions (`BSF, BCF, XORWF`) with TRISC as destination, should be avoided. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

FIGURE 3-5: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<2:0>, RC<7:5>



Note 1: I/O pins have diode protection to VDD and Vss.
2: Port/Peripheral select signal selects between port data and peripheral output.
3: Peripheral OE (output enable) is only activated if peripheral select is active.

FIGURE 3-6: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<4:3>



Note 1: I/O pins have diode protection to VDD and Vss.
2: Port/Peripheral select signal selects between port data and peripheral output.
3: Peripheral OE (output enable) is only activated if peripheral select is active.

TABLE 3-5: PORTC FUNCTIONS

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input.
RC1/T1OSI/CCP2	bit1	ST	Input/output port pin or Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output.
RC2/CCP1	bit2	ST	Input/output port pin or Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	bit3	ST	RC3 can also be the synchronous serial clock for both SPI and I^2C modes.
RC4/SDI/SDA	bit4	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	bit5	ST	Input/output port pin or Synchronous Serial Port data output.
RC6/TX/CK	bit6	ST	Input/output port pin or USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	bit7	ST	Input/output port pin or USART Asynchronous Receive or Synchronous Data.

Legend: ST = Schmitt Trigger input

TABLE 3-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

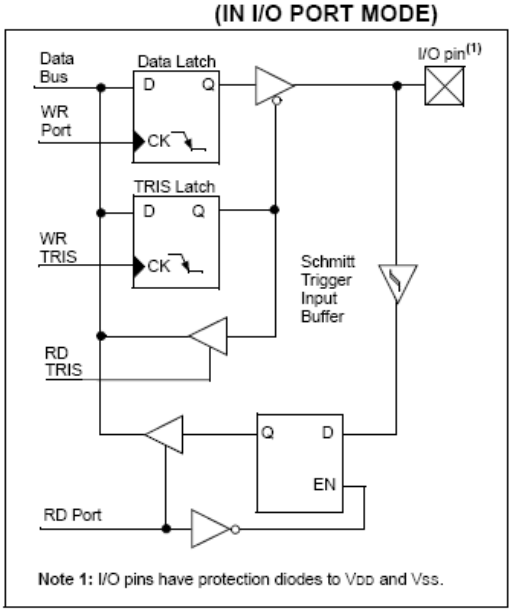
3.4 PORTD and TRISD Registers

PORTD and TRISD are not implemented on the PIC16F873 or PIC16F876.

PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configureable as an input or output.

PORTD can be configured as an 8-bit wide microprocessor port (parallel slave port) by setting control bit PSPMODE (TRISE<4>). In this mode, the input buffers are TTL.

FIGURE 3-7: PORTD BLOCK DIAGRAM (IN I/O PORT MODE)



Note 1: I/O pins have protection diodes to VDD and VSS.

TABLE 3-7: PORTD FUNCTIONS

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit0	ST/TTL(1)	Input/output port pin or parallel slave port bit0.
RD1/PSP1	bit1	ST/TTL(1)	Input/output port pin or parallel slave port bit1.
RD2/PSP2	bit2	ST/TTL(1)	Input/output port pin or parallel slave port bit2.
RD3/PSP3	bit3	ST/TTL(1)	Input/output port pin or parallel slave port bit3.
RD4/PSP4	bit4	ST/TTL(1)	Input/output port pin or parallel slave port bit4.
RD5/PSP5	bit5	ST/TTL(1)	Input/output port pin or parallel slave port bit5.
RD6/PSP6	bit6	ST/TTL(1)	Input/output port pin or parallel slave port bit6.
RD7/PSP7	bit7	ST/TTL(1)	Input/output port pin or parallel slave port bit7.

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 3-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

3.5 PORTE and TRISE Register

PORTE and TRISE are not implemented on the PIC16F873 or PIC16F876.

PORTE has three pins (RE0/$\overline{RD}$/AN5, RE1/$\overline{WR}$/AN6, and RE2/$\overline{CS}$/AN7) which are individually configureable as inputs or outputs. These pins have Schmitt Trigger input buffers.

The PORTE pins become the I/O control inputs for the microprocessor port when bit PSPMODE (TRISE<4>) is set. In this mode, the user must make certain that the TRISE<2:0> bits are set, and that the pins are configured as digital inputs. Also ensure that ADCON1 is configured for digital I/O. In this mode, the input buffers are TTL.

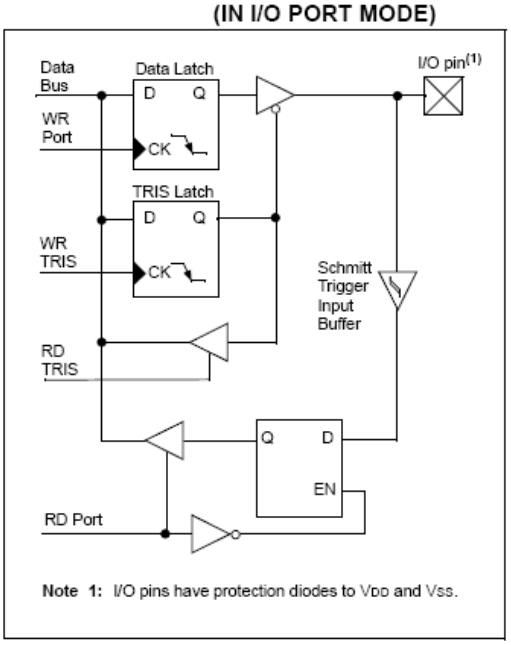
Register 3-1 shows the TRISE register, which also controls the parallel slave port operation.

PORTE pins are multiplexed with analog inputs. When selected for analog input, these pins will read as '0's.

TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

Note:	On a Power-on Reset, these pins are configured as analog inputs, and read as '0'.

FIGURE 3-8: PORTE BLOCK DIAGRAM (IN I/O PORT MODE)



Note 1: I/O pins have protection diodes to VDD and VSS.

TABLE 3-9: PORTE FUNCTIONS

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$/AN5	bit0	ST/TTL(1)	I/O port pin or read control input in Parallel Slave Port mode or analog input: $\overline{RD}$ 1 = Idle 0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected)
RE1/$\overline{WR}$/AN6	bit1	ST/TTL(1)	I/O port pin or write control input in Parallel Slave Port mode or analog input: $\overline{WR}$ 1 = Idle 0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected)
RE2/$\overline{CS}$/AN7	bit2	ST/TTL(1)	I/O port pin or chip select control input in Parallel Slave Port mode or analog input: $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger input, TTL = TTL input
Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 3-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTE.

10.0 ADDRESSABLE UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI.) The USART can be configured as a full duplex asynchronous system that can communicate with peripheral devices such as CRT terminals and personal computers, or it can be configured as a half duplex synchronous system that can communicate with peripheral devices such as A/D or D/A integrated circuits, serial EEPROMs etc.

The USART can be configured in the following modes:

- Asynchronous (full duplex)
- Synchronous - Master (half duplex)
- Synchronous - Slave (half duplex)

Bit SPEN (RCSTA<7>) and bits TRISC<7:6> have to be set in order to configure pins RC6/TX/CK and RC7/RX/DT as the Universal Synchronous Asynchronous Receiver Transmitter.

The USART module also has a multi-processor communication capability using 9-bit address detection.

REGISTER 10-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled

Note: SREN/CREN overrides TXEN in SYNC mode.

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **Unimplemented:** Read as '0'

bit 2 **BRGH**: High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data, can be parity bit

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

REGISTER 10-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode - master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - slave:
Don't care

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit

bit 2 **FERR**: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data (can be parity bit, but must be calculated by user firmware)

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

10.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. In Asynchronous mode, bit BRGH (TXSTA<2>) also controls the baud rate. In Synchronous mode, bit BRGH is ignored. Table 10-1 shows the formula for computation of the baud rate for different USART modes which only apply in Master mode (internal clock).

Given the desired baud rate and Fosc, the nearest integer value for the SPBRG register can be calculated using the formula in Table 10-1. From this, the error in baud rate can be determined.

It may be advantageous to use the high baud rate (BRGH = 1), even for slower baud clocks. This is because the Fosc/(16(X + 1)) equation can reduce the baud rate error in some cases.

Writing a new value to the SPBRG register causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

10.1.1 SAMPLING

The data on the RC7/RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

TABLE 10-1: BAUD RATE FORMULA

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = Fosc/(64(X+1))	Baud Rate = Fosc/(16(X+1))
1	(Synchronous) Baud Rate = Fosc/(4(X+1))	N/A

X = value in SPBRG (0 to 255)

TABLE 10-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used by the BRG.

TABLE 10-3: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	1.221	1.75	255	1.202	0.17	207	1.202	0.17	129
2.4	2.404	0.17	129	2.404	0.17	103	2.404	0.17	64
9.6	9.766	1.73	31	9.615	0.16	25	9.766	1.73	15
19.2	19.531	1.72	15	19.231	0.16	12	19.531	1.72	7
28.8	31.250	8.51	9	27.778	3.55	8	31.250	8.51	4
33.6	34.722	3.34	8	35.714	6.29	6	31.250	6.99	4
57.6	62.500	8.51	4	62.500	8.51	3	52.083	9.58	2
HIGH	1.221	-	255	0.977	-	255	0.610	-	255
LOW	312.500	-	0	250.000	-	0	156.250	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	0.300	0	207	0.3	0	191
1.2	1.202	0.17	51	1.2	0	47
2.4	2.404	0.17	25	2.4	0	23
9.6	8.929	6.99	6	9.6	0	5
19.2	20.833	8.51	2	19.2	0	2
28.8	31.250	8.51	1	28.8	0	1
33.6	-	-	-	-	-	-
57.6	62.500	8.51	0	57.6	0	0
HIGH	0.244	-	255	0.225	-	255
LOW	62.500	-	0	57.6	-	0

TABLE 10-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	-	-	-	-	-	-	-	-	-
2.4	-	-	-	-	-	-	2.441	1.71	255
9.6	9.615	0.16	129	9.615	0.16	103	9.615	0.16	64
19.2	19.231	0.16	64	19.231	0.16	51	19.531	1.72	31
28.8	29.070	0.94	42	29.412	2.13	33	28.409	1.36	21
33.6	33.784	0.55	36	33.333	0.79	29	32.895	2.10	18
57.6	59.524	3.34	20	58.824	2.13	16	56.818	1.36	10
HIGH	4.883	-	255	3.906	-	255	2.441	-	255
LOW	1250.000	-	0	1000.000		0	625.000	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-
1.2	1.202	0.17	207	1.2	0	191
2.4	2.404	0.17	103	2.4	0	95
9.6	9.615	0.16	25	9.6	0	23
19.2	19.231	0.16	12	19.2	0	11
28.8	27.798	3.55	8	28.8	0	7
33.6	35.714	6.29	6	32.9	2.04	6
57.6	62.500	8.51	3	57.6	0	3
HIGH	0.977	-	255	0.9	-	255
LOW	250.000	-	0	230.4	-	0

10.2 USART Asynchronous Mode

In this mode, the USART uses standard non-return-to-zero (NRZ) format (one START bit, eight or nine data bits, and one STOP bit). The most common data format is 8-bits. An on-chip, dedicated, 8-bit baud rate generator can be used to derive standard baud rate frequencies from the oscillator. The USART transmits and receives the LSb first. The transmitter and receiver are functionally independent, but use the same data format and baud rate. The baud rate generator produces a clock, either x16 or x64 of the bit shift rate, depending on bit BRGH (TXSTA<2>). Parity is not supported by the hardware, but can be implemented in software (and stored as the ninth data bit). Asynchronous mode is stopped during SLEEP.

Asynchronous mode is selected by clearing bit SYNC (TXSTA<4>).

The USART Asynchronous module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver

10.2.1 USART ASYNCHRONOUS TRANSMITTER

The USART transmitter block diagram is shown in Figure 10-1. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the STOP bit has been transmitted from the previous load. As soon as the STOP bit is transmitted, the TSR is loaded with new data from the TXREG register (if available). Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and flag bit TXIF (PIR1<4>) is set. This interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE1<4>). Flag bit TXIF will be set, regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicates the status of the TXREG register, another bit TRMT (TXSTA<1>) shows the status of the TSR register. Status bit TRMT is a read only bit, which is set when the TSR register is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty.

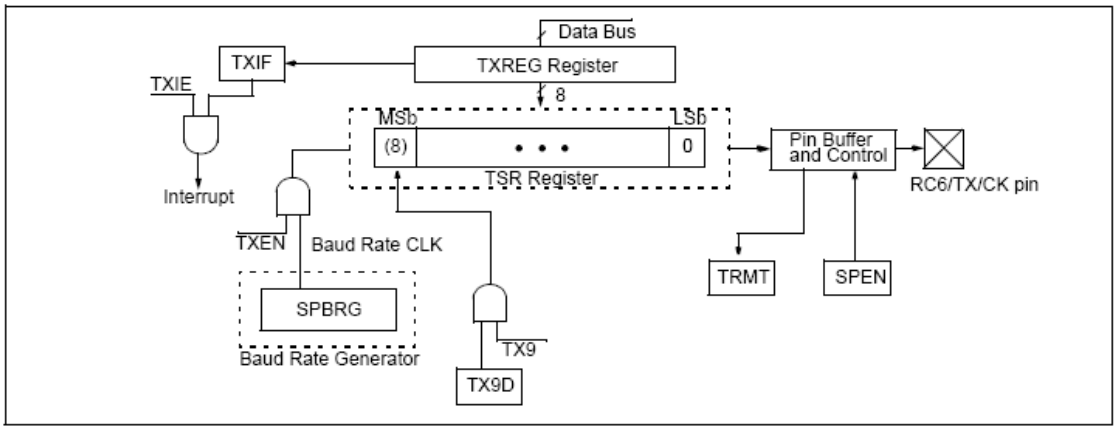
Note 1: The TSR register is not mapped in data memory, so it is not available to the user.

2: Flag bit TXIF is set when enable bit TXEN is set. TXIF is cleared by loading TXREG.

Transmission is enabled by setting enable bit TXEN (TXSTA<5>). The actual transmission will not occur until the TXREG register has been loaded with data and the baud rate generator (BRG) has produced a shift clock (Figure 10-2). The transmission can also be started by first loading the TXREG register and then setting enable bit TXEN. Normally, when transmission is first started, the TSR register is empty. At that point, transfer to the TXREG register will result in an immediate transfer to TSR, resulting in an empty TXREG. A back-to-back transfer is thus possible (Figure 10-3). Clearing enable bit TXEN during a transmission will cause the transmission to be aborted and will reset the transmitter. As a result, the RC6/TX/CK pin will revert to hi-impedance.

In order to select 9-bit transmission, transmit bit TX9 (TXSTA<6>) should be set and the ninth bit should be written to TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to the TXREG register. This is because a data write to the TXREG register can result in an immediate transfer of the data to the TSR register (if the TSR is empty). In such a case, an incorrect ninth data bit may be loaded in the TSR register.

FIGURE 10-1: USART TRANSMIT BLOCK DIAGRAM



10.2.2 USART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 10-4. The data is received on the RC7/RX/DT pin and drives the data recovery block. The data recovery block is actually a high speed shifter, operating at x16 times the baud rate; whereas, the main receive serial shifter operates at the bit rate or at Fosc.

Once Asynchronous mode is selected, reception is enabled by setting bit CREN (RCSTA<4>).

The heart of the receiver is the receive (serial) shift register (RSR). After sampling the STOP bit, the received data in the RSR is transferred to the RCREG register (if it is empty). If the transfer is complete, flag bit RCIF (PIR1<5>) is set. The actual interrupt can be enabled/disabled by setting/clearing enable bit RCIE (PIE1<5>). Flag bit RCIF is a read only bit, which is cleared by the hardware. It is cleared when the RCREG register has been read and is empty. The RCREG is a double buffered register (i.e., it is a two deep FIFO). It is possible for two bytes of data to be received and transferred to the RCREG FIFO and a third byte to begin shifting to the RSR register. On the detection of the STOP bit of the third byte, if the RCREG register is still full, the overrun error bit OERR (RCSTA<1>) will be set. The word in the RSR will be lost. The RCREG register can be read twice to retrieve the two bytes in the FIFO. Overrun bit OERR has to be cleared in software. This is done by resetting the receive logic (CREN is cleared and then set). If bit OERR is set, transfers from the RSR register to the RCREG register are inhibited, and no further data will be received. It is therefore, essential to clear error bit OERR if it is set. Framing error bit FERR (RCSTA<2>) is set if a STOP bit is detected as clear. Bit FERR and the 9th receive bit are buffered the same way as the receive data. Reading the RCREG will load bits RX9D and FERR with new values, therefore, it is essential for the user to read the RCSTA register before reading the RCREG register in order not to lose the old FERR and RX9D information.

FIGURE 10-4: USART RECEIVE BLOCK DIAGRAM

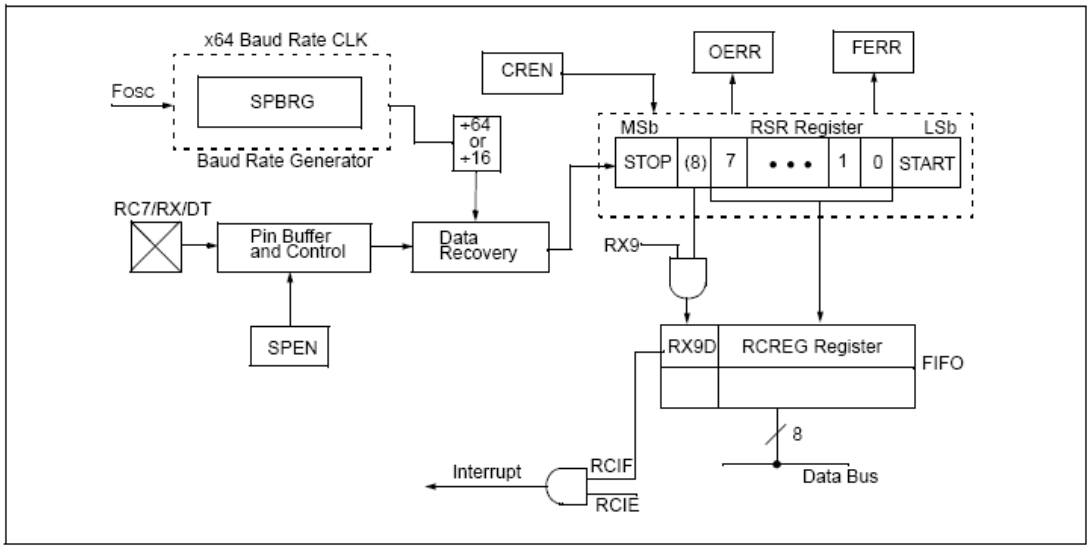
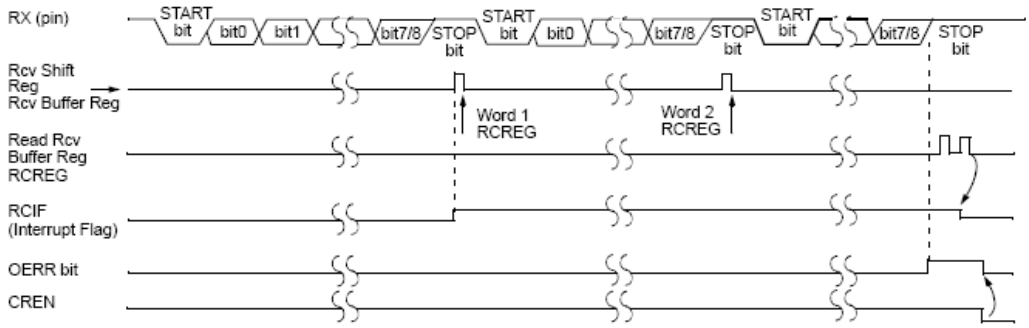


FIGURE 10-5: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

When setting up an Asynchronous Reception, follow these steps:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH (Section 10.1).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, then set enable bit RCIE.
4. If 9-bit reception is desired, then set bit RX9.
5. Enable the reception by setting bit CREN.
6. Flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE is set.
7. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If any error occurred, clear the error by clearing enable bit CREN.
10. If using interrupts, ensure that GIE and PEIE (bits 7 and 6) of the INTCON register are set.

TABLE 10-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.
Note 1: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

12.10 Interrupts

The PIC16F87X family has up to 14 sources of interrupt. The interrupt control register (INTCON) records individual interrupt requests in flag bits. It also has individual and global interrupt enable bits.

Note:	Individual interrupt flag bits are set, regardless of the status of their corresponding mask bit, or the GIE bit.

A global interrupt enable bit, GIE (INTCON<7>) enables (if set) all unmasked interrupts, or disables (if cleared) all interrupts. When bit GIE is enabled, and an interrupt's flag bit and mask bit are set, the interrupt will vector immediately. Individual interrupts can be disabled through their corresponding enable bits in various registers. Individual interrupt bits are set, regardless of the status of the GIE bit. The GIE bit is cleared on RESET.

The "return from interrupt" instruction, `RETFIE`, exits the interrupt routine, as well as sets the GIE bit, which re-enables interrupts.

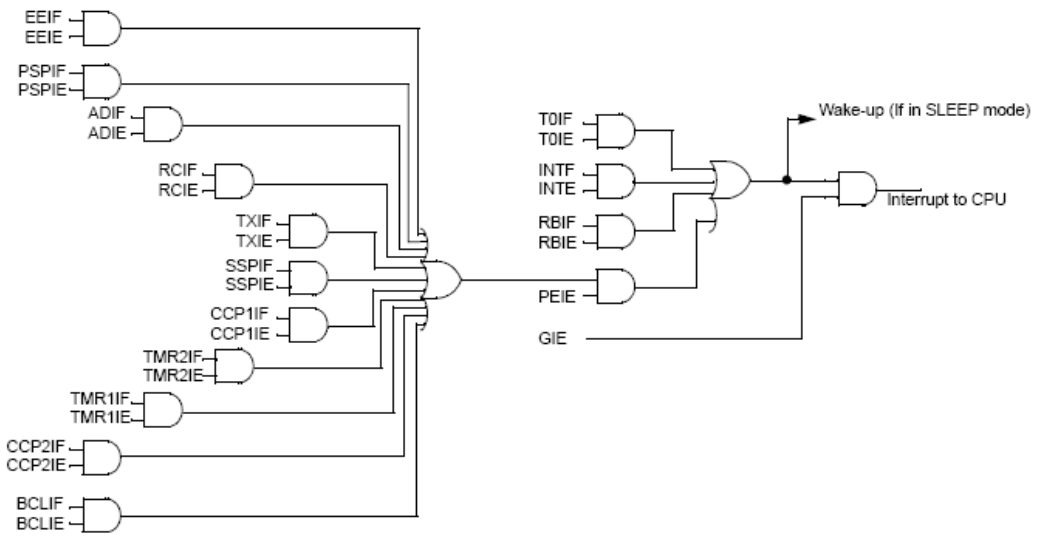
The RB0/INT pin interrupt, the RB port change interrupt, and the TMR0 overflow interrupt flags are contained in the INTCON register.

The peripheral interrupt flags are contained in the special function registers, PIR1 and PIR2. The corresponding interrupt enable bits are contained in special function registers, PIE1 and PIE2, and the peripheral interrupt enable bit is contained in special function register INTCON.

When an interrupt is responded to, the GIE bit is cleared to disable any further interrupt, the return address is pushed onto the stack and the PC is loaded with 0004h. Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

For external interrupt events, such as the INT pin or PORTB change interrupt, the interrupt latency will be three or four instruction cycles. The exact latency depends when the interrupt event occurs. The latency is the same for one or two-cycle instructions. Individual interrupt flag bits are set, regardless of the status of their corresponding mask bit, PEIE bit, or GIE bit.

FIGURE 12-9: INTERRUPT LOGIC



The following table shows which devices have which interrupts.

Device	T0IF	INTF	RBIF	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	EEIF	BCLIF	CCP2IF
PIC16F876/873	Yes	Yes	Yes	—	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PIC16F877/874	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

12.10.1 INT INTERRUPT

External interrupt on the RB0/INT pin is edge triggered, either rising, if bit INTEDG (OPTION_REG<6>) is set, or falling, if the INTEDG bit is clear. When a valid edge appears on the RB0/INT pin, flag bit INTF (INTCON<1>) is set. This interrupt can be disabled by clearing enable bit INTE (INTCON<4>). Flag bit INTF must be cleared in software in the Interrupt Service Routine before re-enabling this interrupt. The INT interrupt can wake-up the processor from SLEEP, if bit INTE was set prior to going into SLEEP. The status of global interrupt enable bit, GIE, decides whether or not the processor branches to the interrupt vector following wake-up. See Section 12.13 for details on SLEEP mode.

12.10.2 TMR0 INTERRUPT

An overflow (FFh → 00h) in the TMR0 register will set flag bit T0IF (INTCON<2>). The interrupt can be enabled/disabled by setting/clearing enable bit T0IE (INTCON<5>) (Section 5.0).

12.10.3 PORTB INTCON CHANGE

An input change on PORTB<7:4> sets flag bit RBIF (INTCON<0>). The interrupt can be enabled/disabled by setting/clearing enable bit RBIE (INTCON<4>) (Section 3.2).

12.11 Context Saving During Interrupts

During an interrupt, only the return PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt, (i.e., W register and STATUS register). This will have to be implemented in software.

For the PIC16F873/874 devices, the register W_TEMP must be defined in both banks 0 and 1 and must be defined at the same offset from the bank base address (i.e., If W_TEMP is defined at 0x20 in bank 0, it must also be defined at 0xA0 in bank 1). The registers, PCLATH_TEMP and STATUS_TEMP, are only defined in bank 0.

Since the upper 16 bytes of each bank are common in the PIC16F876/877 devices, temporary holding registers W_TEMP, STATUS_TEMP, and PCLATH_TEMP should be placed in here. These 16 locations don't require banking and therefore, make it easier for context save and restore. The same code shown in Example 12-1 can be used.

EXAMPLE 12-1: SAVING STATUS, W, AND PCLATH REGISTERS IN RAM

```
MOVWF     W_TEMP            ;Copy W to TEMP register
SWAPF     STATUS,W          ;Swap status to be saved into W
CLRF      STATUS            ;bank 0, regardless of current bank, Clears IRP,RP1,RP0
MOVWF     STATUS_TEMP       ;Save status to bank zero STATUS_TEMP register
MOVF      PCLATH, W         ;Only required if using pages 1, 2 and/or 3
MOVWF     PCLATH_TEMP       ;Save PCLATH into W
CLRF      PCLATH            ;Page zero, regardless of current page
:
:(ISR)                      ;(Insert user code here)
:
MOVF      PCLATH_TEMP, W    ;Restore PCLATH
MOVWF     PCLATH            ;Move W into PCLATH
SWAPF     STATUS_TEMP,W     ;Swap STATUS_TEMP register into W
                            ;(sets bank to original state)
MOVWF     STATUS            ;Move W into STATUS register
SWAPF     W_TEMP,F          ;Swap W_TEMP
SWAPF     W_TEMP,W          ;Swap W_TEMP into W
```

Anexo E. Datasheet del PIC 18F458

PIC18FXX8
Data Sheet

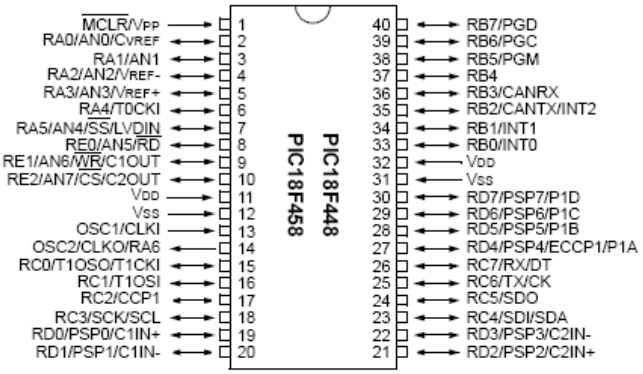
28/40-Pin High-Performance, Enhanced Flash Microcontrollers with CAN Module

PIC18FXX8

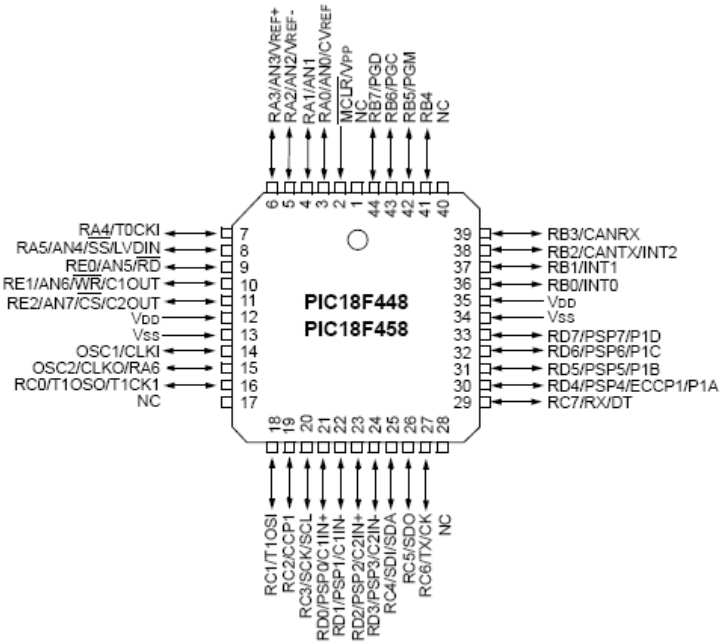
Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	Comparators	CCP/ ECCP (PWM)	MSSP		USART	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI™	Master I^2C™		
PIC18F248	16K	8192	768	256	22	5	—	1/0	Y	Y	Y	1/3
PIC18F258	32K	16384	1536	256	22	5	—	1/0	Y	Y	Y	1/3
PIC18F448	16K	8192	768	256	33	8	2	1/1	Y	Y	Y	1/3
PIC18F458	32K	16384	1536	256	33	8	2	1/1	Y	Y	Y	1/3

Pin Diagrams

PDIP



PLCC



REGISTER 5-1: EECON1: EEPROM CONTROL REGISTER 1

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR	WREN	$\overline{WR}$	$\overline{RD}$
bit 7							bit 0

bit 7 **EEPGD:** Flash Program or Data EEPROM Memory Select bit
1 = Access program Flash memory
0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EE or Configuration Select bit
1 = Access Configuration registers
0 = Access program Flash or data EEPROM memory

bit 5 **Unimplemented:** Read as '0'

bit 4 **FREE:** Flash Row Erase Enable bit
1 = Erase the program memory row addressed by TBLPTR on the next $\overline{WR}$ command (reset by hardware)
0 = Perform write only

bit 3 **WRERR:** Write Error Flag bit
1 = A write operation is prematurely terminated (any $\overline{MCLR}$ or any WDT Reset during self-timed programming in normal operation)
0 = The write operation completed

Note: When a WRERR occurs, the EEPGD or FREE bits are not cleared. This allows tracing of the error condition.

bit 2 **WREN:** Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the EEPROM or Flash memory

bit 1 **$\overline{WR}$:** Write Control bit
1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle (The operation is self-timed and the bit is cleared by hardware once write is complete. The $\overline{WR}$ bit can only be set (not cleared) in software.)
0 = Write cycle is complete

bit 0 **$\overline{RD}$:** Read Control bit
1 = Initiates an EEPROM read (Read takes one cycle. $\overline{RD}$ is cleared in hardware. The $\overline{RD}$ bit can only be set (not cleared) in software. $\overline{RD}$ bit cannot be set when EEPGD = 1.)
0 = Does not initiate an EEPROM read

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

8.0 INTERRUPTS

The PIC18FXX8 devices have multiple interrupt sources and an interrupt priority feature that allows each interrupt source to be assigned a high priority level or a low priority level. The high priority interrupt vector is at 000008h and the low priority interrupt vector is at 000018h. High priority interrupt events will override any low priority interrupts that may be in progress.

There are 13 registers that are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3
- PIE1, PIE2, PIE3
- IPR1, IPR2, IPR3

It is recommended that the Microchip header files, supplied with MPLAB® IDE, be used for the symbolic bit names in these registers. This allows the assembler/ compiler to automatically take care of the placement of these bits within the specified register.

Each interrupt source has three bits to control its operation. The functions of these bits are:

- Flag bit to indicate that an interrupt event occurred
- Enable bit that allows program execution to branch to the interrupt vector address when the flag bit is set
- Priority bit to select high priority or low priority

The interrupt priority feature is enabled by setting the IPEN bit (RCON register). When interrupt priority is enabled, there are two bits that enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts. Setting the GIEL bit (INTCON register) enables all interrupts that have the priority bit cleared. When the interrupt flag, enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 000008h or 000018h, depending on the priority level. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PICmicro® mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. The PEIE bit (INTCON register) enables/disables all peripheral interrupt sources. The GIE bit (INTCON register) enables/disables all interrupt sources. All interrupts branch to address 000008h in Compatibility mode.

When an interrupt is responded to, the global interrupt enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIEH or GIEL bit. High priority interrupt sources can interrupt a low priority interrupt.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (000008h or 000018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

The "return from interrupt" instruction, `RETFIE`, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used), which re-enables interrupts.

For external interrupt events, such as the INT pins or the PORTB input change interrupt, the interrupt latency will be three to four instruction cycles. The exact latency is the same for one or two-cycle instructions. Individual interrupt flag bits are set regardless of the status of their corresponding enable bit or the GIE bit.

Note:	Do not use the `MOVFF` instruction to modify any of the interrupt control registers while **any** interrupt is enabled. Doing so may cause erratic microcontroller behavior.

8.1 INTCON Registers

The INTCON registers are readable and writable registers which contain various enable, priority and flag bits. Because of the number of interrupts to be controlled, PIC18FXX8 devices have three INTCON registers. They are detailed in Register 8-1 through Register 8-3.

Note:	Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global interrupt enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows software polling.

REGISTER 8-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

bit 7 **GIE/GIEH:** Global Interrupt Enable bit

When IPEN (RCON<7>) = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts

When IPEN (RCON<7>) = 1:
1 = Enables all high priority interrupts
0 = Disables all priority interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit

When IPEN (RCON<7>) = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

When IPEN (RCON<7>) = 1:
1 = Enables all low priority peripheral interrupts
0 = Disables all low priority peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared in software by reading PORTB)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Note: A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.

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

REGISTER 8-2: INTCON2: INTERRUPT CONTROL REGISTER 2

R/W-1	R/W-1	R/W-1	U-0	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	—	—	TMR0IP	—	RBIP
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**: PORTB Pull-up Enable bit
1 = All PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG0**: External Interrupt 0 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 5 **INTEDG1**: External Interrupt 1 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 4-3 **Unimplemented:** Read as '0'

bit 2 **TMR0IP**: TMR0 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **Unimplemented:** Read as '0'

bit 0 **RBIP**: RB Port Change Interrupt Priority bit
1 = High priority
0 = Low priority

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

Note: Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global interrupt enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows software polling.

REGISTER 8-3: INTCON3: INTERRUPT CONTROL REGISTER 3

R/W-1	R/W-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF
bit 7							bit 0

bit 7 **INT2IP:** INT2 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **INT1IP:** INT1 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **Unimplemented:** Read as '0'

bit 4 **INT2IE:** INT2 External Interrupt Enable bit
1 = Enables the INT2 external interrupt
0 = Disables the INT2 external interrupt

bit 3 **INT1IE:** INT1 External Interrupt Enable bit
1 = Enables the INT1 external interrupt
0 = Disables the INT1 external interrupt

bit 2 **Unimplemented:** Read as '0'

bit 1 **INT2IF:** INT2 External Interrupt Flag bit
1 = The INT2 external interrupt occurred (must be cleared in software)
0 = The INT2 external interrupt did not occur

bit 0 **INT1IF:** INT1 External Interrupt Flag bit
1 = The INT1 external interrupt occurred (must be cleared in software)
0 = The INT1 external interrupt did not occur

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

Note: Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global interrupt enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows software polling.

8.3 PIE Registers

The Peripheral Interrupt Enable (PIE) registers contain the individual enable bits for the peripheral interrupts (Register 8-7 through Register 8-9). Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Enable registers (PIE1, PIE2, PIE3). When IPEN is clear, the PEIE bit must be set to enable any of these peripheral interrupts.

REGISTER 8-7: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit(1)
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE**: Master Synchronous Serial Port Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Note 1: This bit is only available on PIC18F4X8 devices. For PIC18F2X8 devices, this bit is unimplemented and reads as '0'.

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

REGISTER 8-8: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	CMIE(1)	—	EEIE	BCLIE	LVDIE	TMR3IE	ECCP1IE(1)
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6 **CMIE:** Comparator Interrupt Enable bit(1)
1 = Enables the comparator interrupt
0 = Disables the comparator interrupt

bit 5 **Unimplemented:** Read as '0'

bit 4 **EEIE**: EEPROM Write Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **BCLIE**: Bus Collision Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **LVDIE**: Low-Voltage Detect Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **TMR3IE**: TMR3 Overflow Interrupt Enable bit
1 = Enables the TMR3 overflow interrupt
0 = Disables the TMR3 overflow interrupt

bit 0 **ECCP1IE**: ECCP1 Interrupt Enable bit(1)
1 = Enables the ECCP1 interrupt
0 = Disables the ECCP1 interrupt

Note 1: This bit is only available on PIC18F4X8 devices. For PIC18F2X8 devices, this bit is unimplemented and reads as '0'.

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

REGISTER 8-9: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
IRXIE	WAKIE	ERRIE	TXB2IE	TXB1IE	TXB0IE	RXB1IE	RXB0IE
bit 7							bit 0

bit 7 **IRXIE:** Invalid CAN Message Received Interrupt Enable bit
1 = Enables the invalid CAN message received interrupt
0 = Disables the invalid CAN message received interrupt

bit 6 **WAKIE:** Bus Activity Wake-up Interrupt Enable bit
1 = Enables the bus activity wake-up interrupt
0 = Disables the bus activity wake-up interrupt

bit 5 **ERRIE:** CAN bus Error Interrupt Enable bit
1 = Enables the CAN bus error interrupt
0 = Disables the CAN bus error interrupt

bit 4 **TXB2IE:** Transmit Buffer 2 Interrupt Enable bit
1 = Enables the Transmit Buffer 2 interrupt
0 = Disables the Transmit Buffer 2 interrupt

bit 3 **TXB1IE:** Transmit Buffer 1 Interrupt Enable bit
1 = Enables the Transmit Buffer 1 interrupt
0 = Disables the Transmit Buffer 1 interrupt

bit 2 **TXB0IE:** Transmit Buffer 0 Interrupt Enable bit
1 = Enables the Transmit Buffer 0 interrupt
0 = Disables the Transmit Buffer 0 interrupt

bit 1 **RXB1IE:** Receive Buffer 1 Interrupt Enable bit
1 = Enables the Receive Buffer 1 interrupt
0 = Disables the Receive Buffer 1 interrupt

bit 0 **RXB0IE:** Receive Buffer 0 Interrupt Enable bit
1 = Enables the Receive Buffer 0 interrupt
0 = Disables the Receive Buffer 0 interrupt

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

17.4.3.2 Reception

When the R/$\overline{W}$ bit of the address byte is clear and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register and the SDA line is held low ($\overline{ACK}$).

When the address byte overflow condition exists, then the no Acknowledge ($\overline{ACK}$) pulse is given. An overflow condition is defined as either bit BF (SSPSTAT<0>) is set or bit SSPOV (SSPCON1<6>) is set.

An MSSP interrupt is generated for each data transfer byte. Flag bit SSPIF (PIR1<3>) must be cleared in software. The SSPSTAT register is used to determine the status of the byte.

If SEN is enabled (SSPCON2<0> = 1), RC3/SCK/SCL will be held low (clock stretch) following each data transfer. The clock must be released by setting bit CKP (SSPCON1<4>). See **Section 17.4.4 "Clock Stretching"** for more detail.

17.4.3.3 Transmission

When the R/$\overline{W}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The $\overline{ACK}$ pulse will be sent on the ninth bit and pin RC3/SCK/SCL is held low regardless of SEN (see **Section 17.4.4 "Clock Stretching"** for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data. The transmit data must be loaded into the SSPBUF register, which also loads the SSPSR register. Then, pin RC3/SCK/SCL should be enabled by setting bit CKP (SSPCON1<4>). The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time (Figure 17-9).

The $\overline{ACK}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. If the SDA line is high (not $\overline{ACK}$), then the data transfer is complete. In this case, when the $\overline{ACK}$ is latched by the slave, the slave logic is reset (resets SSPSTAT register) and the slave monitors for another occurrence of the Start bit. If the SDA line was low ($\overline{ACK}$), the next transmit data must be loaded into the SSPBUF register. Again, pin RC3/SCK/SCL must be enabled by setting bit CKP.

An MSSP interrupt is generated for each data transfer byte. The SSPIF bit must be cleared in software and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

REGISTER 18-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive (this bit is cleared after reception is complete)
Synchronous mode – Slave:
Unused in this mode.

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN**: Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit

bit 2 **FERR**: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data
Can be address/data bit or a parity bit.

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

18.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit Baud Rate Generator. The SPBRG register controls the period of a free running, 8-bit timer. In Asynchronous mode, bit BRGH (TXSTA register) also controls the baud rate. In Synchronous mode, bit BRGH is ignored. Table 18-1 shows the formula for computation of the baud rate for different USART modes which only apply in Master mode (internal clock).

Given the desired baud rate and Fosc, the nearest integer value for the SPBRG register can be calculated using the formula in Table 18-1. From this, the error in baud rate can be determined.

Example 18-1 shows the calculation of the baud rate error for the following conditions:

Fosc = 16 MHz
Desired Baud Rate = 9600
BRGH = 0
SYNC = 0

It may be advantageous to use the high baud rate (BRGH = 1) even for slower baud clocks. This is because the Fosc/(16(X + 1)) equation can reduce the baud rate error in some cases.

Writing a new value to the SPBRG register causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

18.1.1 SAMPLING

The data on the RC7/RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

EXAMPLE 18-1: CALCULATING BAUD RATE ERROR

Desired Baud Rate = Fosc/(64 (X + 1))

Solving for X:

X = ((Fosc/Desired Baud Rate)/64) – 1
X = ((16000000/9600)/64) – 1
X = [25.042] = 25

Calculated Baud Rate = 16000000/(64 (25 + 1))
= 9615

$$\text{Error} = \frac{\text{Calculated Baud Rate} - \text{Desired Baud Rate}}{\text{Desired Baud Rate}}$$
= (9615 – 9600)/9600
= 0.16%

TABLE 18-1: BAUD RATE FORMULA

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = Fosc/(64 (X + 1))	Baud Rate = Fosc/(16 (X + 1))
1	(Synchronous) Baud Rate = Fosc/(4 (X + 1))	NA

Legend: X = value in SPBRG (0 to 255)

TABLE 18-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000u
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used by the BRG.

TABLE 18-3: BAUD RATES FOR SYNCHRONOUS MODE

BAUD RATE (Kbps)	Fosc = 40 MHz		SPBRG value (decimal)	33 MHz		SPBRG value (decimal)	25 MHz		SPBRG value (decimal)	20 MHz		SPBRG value (decimal)
	KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR	
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	NA	-	-	NA	-	-
9.6	NA	-	-	NA	-	-	NA	-	-	NA	-	-
19.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
76.8	76.92	+0.16	129	77.10	+0.39	106	77.16	+0.47	80	76.92	+0.16	64
96	96.15	+0.16	103	95.93	-0.07	85	96.15	+0.16	64	96.15	+0.16	51
300	303.03	+1.01	32	294.64	-1.79	27	297.62	-0.79	20	294.12	-1.96	16
500	500	0	19	485.30	-2.94	16	480.77	-3.85	12	500	0	9
HIGH	10000	-	0	8250	-	0	6250	-	0	5000	-	0
LOW	39.06	-	255	32.23	-	255	24.41	-	255	19.53	-	255

BAUD RATE (Kbps)	Fosc = 16 MHz		SPBRG value (decimal)	10 MHz		SPBRG value (decimal)	7.15909 MHz		SPBRG value (decimal)	5.0688 MHz		SPBRG value (decimal)
	KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR	
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	NA	-	-	NA	-	-
9.6	NA	-	-	NA	-	-	9.62	+0.23	185	9.60	0	131
19.2	19.23	+0.16	207	19.23	+0.16	129	19.24	+0.23	92	19.20	0	65
76.8	76.92	+0.16	51	75.76	-1.36	32	77.82	+1.32	22	74.54	-2.94	16
96	95.24	-0.79	41	96.15	+0.16	25	94.20	-1.88	18	97.48	+1.54	12
300	307.70	+2.56	12	312.50	+4.17	7	298.35	-0.57	5	316.80	+5.60	3
500	500	0	7	500	0	4	447.44	-10.51	3	422.40	-15.52	2
HIGH	4000	-	0	2500	-	0	1789.80	-	0	1267.20	-	0
LOW	15.63	-	255	9.77	-	255	6.99	-	255	4.95	-	255

BAUD RATE (Kbps)	Fosc = 4 MHz		SPBRG value (decimal)	3.579545 MHz		SPBRG value (decimal)	1 MHz		SPBRG value (decimal)	32.768 kHz		SPBRG value (decimal)
	KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR	
0.3	NA	-	-	NA	-	-	NA	-	-	0.30	+1.14	26
1.2	NA	-	-	NA	-	-	1.20	+0.16	207	1.17	-2.48	6
2.4	NA	-	-	NA	-	-	2.40	+0.16	103	2.73	+13.78	2
9.6	9.62	+0.16	103	9.62	+0.23	92	9.62	+0.16	25	8.20	-14.67	0
19.2	19.23	+0.16	51	19.04	-0.83	46	19.23	+0.16	12	NA	-	-
76.8	76.92	+0.16	12	74.57	-2.90	11	83.33	+8.51	2	NA	-	-
96	1000	+4.17	9	99.43	+3.57	8	83.33	-13.19	2	NA	-	-
300	333.33	+11.11	2	298.30	-0.57	2	250	-16.67	0	NA	-	-
500	500	0	1	447.44	-10.51	1	NA	-	-	NA	-	-
HIGH	1000	-	0	894.89	-	0	250	-	0	8.20	-	0
LOW	3.91	-	255	3.50	-	255	0.98	-	255	0.03	-	255

TABLE 18-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

BAUD RATE (Kbps)	Fosc = 40 MHz		SPBRG value (decimal)	33 MHz		SPBRG value (decimal)	25 MHz		SPBRG value (decimal)	20 MHz		SPBRG value (decimal)
	KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR	
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	2.40	-0.07	214	2.40	-0.15	162	2.40	+0.16	129
9.6	9.62	+0.16	64	9.55	-0.54	53	9.53	-0.76	40	9.47	-1.36	32
19.2	18.94	-1.36	32	19.10	-0.54	26	19.53	+1.73	19	19.53	+1.73	15
76.8	78.13	+1.73	7	73.66	-4.09	6	78.13	+1.73	4	78.13	+1.73	3
96	89.29	-6.99	6	103.13	+7.42	4	97.66	+1.73	3	104.17	+8.51	2
300	312.50	+4.17	1	257.81	-14.06	1	NA	-	-	312.50	+4.17	0
500	625	+25.00	0	NA	-	-	NA	-	-	NA	-	-
HIGH	625	-	0	515.63	-	0	390.63	-	0	312.50	-	0
LOW	2.44	-	255	2.01	-	255	1.53	-	255	1.22	-	255

BAUD RATE (Kbps)	Fosc = 16 MHz		SPBRG value (decimal)	10 MHz		SPBRG value (decimal)	7.15909 MHz		SPBRG value (decimal)	5.0688 MHz		SPBRG value (decimal)
	KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR	
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	1.20	+0.16	207	1.20	+0.16	129	1.20	+0.23	92	1.20	0	65
2.4	2.40	+0.16	103	2.40	+0.16	64	2.38	-0.83	46	2.40	0	32
9.6	9.62	+0.16	25	9.77	+1.73	15	9.32	-2.90	11	9.90	+3.13	7
19.2	19.23	+0.16	12	19.53	+1.73	7	18.64	-2.90	5	19.80	+3.13	3
76.8	83.33	+8.51	2	78.13	+1.73	1	111.86	+45.65	0	79.20	+3.13	0
96	83.33	-13.19	2	78.13	-18.62	1	NA	-	-	NA	-	-
300	250	-16.67	0	156.25	-47.92	0	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	250	-	0	156.25	-	0	111.86	-	0	79.20	-	0
LOW	0.98	-	255	0.61	-	255	0.44	-	255	0.31	-	255

BAUD RATE (Kbps)	Fosc = 4 MHz		SPBRG value (decimal)	3.579545 MHz		SPBRG value (decimal)	1 MHz		SPBRG value (decimal)	32.768 kHz		SPBRG value (decimal)
	KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR	
0.3	0.30	-0.16	207	0.30	+0.23	185	0.30	+0.16	51	0.26	-14.67	1
1.2	1.20	+1.67	51	1.19	-0.83	46	1.20	+0.16	12	NA	-	-
2.4	2.40	+1.67	25	2.43	+1.32	22	2.23	-6.99	6	NA	-	-
9.6	8.93	-6.99	6	9.32	-2.90	5	7.81	-18.62	1	NA	-	-
19.2	20.83	+8.51	2	18.64	-2.90	2	15.63	-18.62	0	NA	-	-
76.8	62.50	-18.62	0	55.93	-27.17	0	NA	-	-	NA	-	-
96	NA	-	-	NA	-	-	NA	-	-	NA	-	-
300	NA	-	-	NA	-	-	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	62.50	-	0	55.93	-	0	15.63	-	0	0.51	-	0
LOW	0.24	-	255	0.22	-	255	0.06	-	255	0.002	-	255

TABLE 18-5: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

BAUD RATE (Kbps)	Fosc = 40 MHz		SPBRG value (decimal)	33 MHz		SPBRG value (decimal)	25 MHz		SPBRG value (decimal)	20 MHz		SPBRG value (decimal)
	KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR	
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	NA	-	-	NA	-	-
9.6	NA	-	-	9.60	-0.07	214	9.59	-0.15	162	9.62	+0.16	129
19.2	19.23	+0.16	129	19.28	+0.39	106	19.30	+0.47	80	19.23	+0.16	64
76.8	75.76	-1.36	32	76.39	-0.54	26	78.13	+1.73	19	78.13	+1.73	15
96	96.15	+0.16	25	98.21	+2.31	20	97.66	+1.73	15	96.15	+0.16	12
300	312.50	+4.17	7	294.64	-1.79	6	312.50	+4.17	4	312.50	+4.17	3
500	500	0	4	515.63	+3.13	3	520.83	+4.17	2	416.67	-16.67	2
HIGH	2500	-	0	2062.50	-	0	1562.50	-	0	1250	-	0
LOW	9.77	-	255	8,06	-	255	6.10	-	255	4.88	-	255

BAUD RATE (Kbps)	Fosc = 16 MHz		SPBRG value (decimal)	10 MHz		SPBRG value (decimal)	7.15909 MHz		SPBRG value (decimal)	5.0688 MHz		SPBRG value (decimal)
	KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR	
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	2.41	+0.23	185	2.40	0	131
9.6	9.62	+0.16	103	9.62	+0.16	64	9.52	-0.83	46	9.60	0	32
19.2	19.23	+0.16	51	18.94	-1.36	32	19.45	+1.32	22	18.64	-2.94	16
76.8	76.92	+0.16	12	78.13	+1.73	7	74.57	-2.90	5	79.20	+3.13	3
96	100	+4.17	9	89.29	-6.99	6	89.49	-6.78	4	105.60	+10.00	2
300	333.33	+11.11	2	312.50	+4.17	1	447.44	+49.15	0	316.80	+5.60	0
500	500	0	1	625	+25.00	0	447.44	-10.51	0	NA	-	-
HIGH	1000	-	0	625	-	0	447.44	-	0	316.80	-	0
LOW	3.91	-	255	2.44	-	255	1.75	-	255	1.24	-	255

BAUD RATE (Kbps)	Fosc = 4 MHz		SPBRG value (decimal)	3.579545 MHz		SPBRG value (decimal)	1 MHz		SPBRG value (decimal)	32.768 kHz		SPBRG value (decimal)
	KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR	
0.3	NA	-	-	NA	-	-	0.30	+0.16	207	0.29	-2.48	6
1.2	1.20	+0.16	207	1.20	+0.23	185	1.20	+0.16	51	1.02	-14.67	1
2.4	2.40	+0.16	103	2.41	+0.23	92	2.40	+0.16	25	2.05	-14.67	0
9.6	9.62	+0.16	25	9.73	+1.32	22	8.93	-6.99	6	NA	-	-
19.2	19.23	+0.16	12	18.64	-2.90	11	20.83	+8.51	2	NA	-	-
76.8	NA	-	-	74.57	-2.90	2	62.50	-18.62	0	NA	-	-
96	NA	-	-	111.86	+16.52	1	NA	-	-	NA	-	-
300	NA	-	-	223.72	-25.43	0	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	250	-	0	55.93	-	0	62.50	-	0	2.05	-	0
LOW	0.98	-	255	0.22	-	255	0.24	-	255	0.008	-	255

Anexo F. Código Fuente

Imágenes del Control Remoto

```
module Imagen

const Inicio as
byte[1024] = (
  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,
0,  0,  0,
0,248,248,248,248,
0,224,
 224,224,224,  0,
0,248,248,248,248,
0,  0,
0,192,240,240,112,
 112,
0,192,224,240,240,
240,240,224,224,1
28,  0,  0,
0,240,248,
 252,126, 62,
62,126,252,248,24
0,  0,  0,  0,  0,  0,
0,  0,  0,
0,254,254,254,254,
254,254,254,254,2
54,254,254,254,
 254,254,254,  0,
0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,
0,  0,  0,
0,255,255,255,255,
255,255,

255,255,255,255,2
55,255,255,255,25
5,  0,  0,  0,
63,255,255,254,
 254,255,255,255,
63,143,199,255,25
5,255,255,  0,  0,
0,255,255,
31,127,255,255,25
5,254,252,248,252,
254,255,255,255,1
27,
  31,  0,  0,
  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,  0,  0,  1,
  1,  1,  1,  0,  0,  1,
1,  1,  1,  1,  0,  0,
0,  0,  3,  3,
  3,  3,  3,  3,  3,  3,
3,  3,  3,  3,  3,  3,
3,  0,  0,  0,
  0,  0,  0,  1,  1,  3,
3,  3,  3,  3,  1,  1,
0,  0,  0,  0,
  0,  0,  0,  0,  0,  1,
1,  3,  3,  3,  3,  3,
1,  1,  0,  0,
  0,  0,  0,  0,  7,  7,
7,  7,  7,  7,  7,  7,
7,  7,  7,  7,
  7,  7,  7,  0,  0,  0,
0,128,128,192,192,
192,192,128,128,
 31, 31, 31, 31, 31,
31, 31, 31, 31, 31,
31,
 31, 31,  0,  0,  0,
31, 31, 31, 31, 31,
31, 31, 31, 31, 31,
31,
 31, 31, 31, 31,  0,
0,
0,127,127,127,127,
126,126,127,127,1
27,

127,127,127,127,1
27,127,  0,  0,  0,
3, 15, 15, 31, 31,
31, 31,
 15,  3, 24, 28, 31,
31, 31, 15,  0,  0,
0, 31, 31, 31, 63,
63,

63,127,127,127,12
7,255,255,255,254,
254,  0,  0,  0,  0,

  0,  0,  0,  0,  0,  0,
0,  0,  1,  1,  0,  0,
0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,  0,  0,  0
)

const Botones as
byte[1024] = (
  0,  0,  0,  0,  0,
0,240,240,240,240,
240,112,112,112,1
12,112,

112,112,112,112,1
12,240,240,112,11
2,112,112,112,112,
112,112,112,

112,112,112,112,2
40,240,240,240,24
0,240,240,240,  0,
0,  0,  0,

0,240,240,240,240,
240,240,240,240,
48, 48, 48, 48,
48,240,  0,
  0,  0,  0,  0,  0,
0,224,224,224,224,
224,224,224,224,2
24,224,
  96,
96,224,224,224,22
4,224,224,224,224,
128,128,128,128,2
55,255,255,255,25
5,255,255,255,  0,
0,  0,  0,

0,255,255,255,255,
255,255,255,255,
0,  0,  0,  0,  0,255,
0,
 128,128,128,  0,
0,
0,255,255,255,255,
255,255,255,255,2
55,255,
  0,
0,255,255,255,255,
255,255,255,255,2
55,255,255,255,  0,
0,
  0,  0,  0,  0,  0,  0,
0,224,224,224,224,
224,224,224,224,2
24,

224,224,224,224,2
24,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,
0,255,255,255,255,
255,  1,  1,  1,  1,
1,
  1,  1,  1,  1,
1,255,255,  1,  1,
1,  1,  1,  1,  1,  1,
1,
,255,255,255,255,
4,  4,  4,  4,  4,255,
0,
 31, 31, 31,  0,  0,
127,127,255,255,2
55,

255,255,255,255,2
55,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,
7,  7,  7,  7,  7,  6,
6,  6,  6,  6,
  6,  6,  6,  6,  6,  7,
7,  6,  6,  6,  6,  6,
6,  6,  6,  6,
  6,  6,  6,  6,  7,  7,
7,  7,  7,  7,  7,  7,
0,  0,  0,  0,

0,255,255,255,255,
255,255,255,255,
0,  0,  0,  0,  0,255,
0,
  0,  0,  0,  0,  0,
128,128,249,249,2
49,249,255,255,25
5,255,255,255,255,
255,  0,  0,
  0,  0,  0,  0,  0,  0,
0,255,255,255,255,
64,103,229,224,22
9,
  0,  1,  1,  1,  1,  1,
1,  1, 49, 49, 49,  1,
1,  1,  1, 48,
 48, 48, 48, 48, 48,
48, 48, 48, 48,  0,
0,  0, 48, 48, 48,
48,
 48, 48, 48, 48, 48,
48, 48, 48, 48, 48,
48, 48, 48, 48,  0,
0,
  0,  0,  0,  0,  0,  0,
0,255,255,255,255,
192,231,101,
96,101,

103,195,199,255,2
55,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,254,
1,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,
1,113,113,114,
 254,  1,  1,  3,253,
1,  3,253,  2,
6,252,  0,  0,  0,  0,
0,
  0,  0,  0,  0,  0,  0,
0,  0, 51, 51, 51,  0,
0,  0,  0, 51,
 51, 51, 51, 51, 51,
51, 51, 51, 51,  0,
```

```
0,  0, 51, 51, 51,
51,
  51, 51, 51, 51, 51,
51, 51, 51, 51, 51,
51, 51, 51, 51,  0,
0,
   0,  0,  0,  0,  0,  0,
0,255,255,255,255,
192,136,140,140,1
40,
0,  0,  0,  0,  0,  0,
0,  0
)

const Luz as
byte[1024] = (
   0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,252,252,252,

252,252,252,252,2
52,252,252,
12,252,252,252,25
2,252,252,252,252,
 252,252,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,

0,252,252,252,252,
252,252,252,252,2
52,252,
12,252,252,252,25
2,

252,252,252,252,2
52,252,  0,  0,  0,
0,  0,  0,
0,252,252,252,

252,252,252,252,2
52,252,252,
12,252,252,252,25
2,252,252,252,252,
 252,252,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,255,255,255,

255,249,249,249,2
49,249,249,
0,249,249,249,249,
249,249,255,255,
 255,255,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,

0,255,255,255,255,
243,243,243,243,2
43,243,
0,243,243,243,243,
 243,243,255,255,2
64,192,
 192,128,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,
   0,  0,128,192,
64,192,128,  0,
0,128,128,240,240,
128,  0,  0,
const ventana as
byte[1024] = (
   0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,252,252,252,

252,252,252,252,2
52,252,252,
28,252,252,252,25

0,255,255,255,255,
243,243,243,243,2
43,243,
0,243,243,243,243,

243,243,255,255,2
55,255,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,
   0,  0,
0,255,255,255,255,
243,243,243,243,2
43,243,  0,243,243,

239,239,239,239,2
39,239,239,238,23
9,239,239,239,239,
239,239,239,
 239,239,  0,  0,  0,
0,120,252,204, 28,
48, 64,252,252,  0,
0,
0,255,255,255,255,
255,255,255,255,
 255,255,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,

0,255,255,255,255,
255,255,255,255,2
55,255,
0,255,255,255,255,
 24, 24, 24, 24, 24,
24,248,248,248,24
8,248,248,248,248,
248,248,

248,248,248,248,2
48,248,  0,  0,  0,
0,248,248, 24, 24,
24, 24,
  24, 24, 24, 24, 24,
24, 24,
24,248,248,248,24
8,248,248,248,248,

248,248,248,248,2
48,248,248,248,  0,
0,  0, 15, 15, 15,
15, 15,
  15, 15, 15, 15, 15,
15, 15, 15, 15, 15,
0,  0,  0,  0,  0,  0,
   0,  0,  0,
0,255,255,255,255,

0,  0,  0,  0,  0,
0,224,224,
 128,128,239,239,2
0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0
)

const Persiana as
byte[1024] = (
   0,  0,  0,  0,  0,  0,
0,
0,248,248,248,248,
248,248,248,248,
 248,
56,248,248,248,24
8,248,248,248,248,
248,  0,  0,  0,  0,
0,
   0,
0,248,248,248,248,
248,248,248,248,2
48,
56,248,248,248,24
8,

248,248,248,248,2
48,  0,  0,  0,  0,  0,
0,  0,
0,248,248,248,

248,248,248,248,2
48,248,
56,248,248,248,24
8,248,248,248,248,
248,
0,255,255,255,223,
223,223,223,223,
 223,
0,223,223,223,223,

223,223,255,255,2
55,  0,  0,  0,  0,  0,
   0,
0,255,255,255,223,
223,223,223,223,2
23,
0,223,223,223,223,

223,223,255,255,2
55,  0,  0,  0,  0,  0,
0,  0,
0,255,255,255,

223,223,223,223,2
23,223,
0,223,223,223,223,
223,223,255,255,2
55,
   0,  0,  0,  0,  0,  0,
0,255,255,255,223,
223,223,223,223,2
23,

0,223,223,223,223,
223,223,255,255,
0,  0,  0,  0,  0,  0,
0,
,255,255,  0,  0,  0,
0,
   0,  0,  0,  0,  0,  0,
0,255,255,255,255,
255,255,255,255,
 255,
0,255,255,255,255,
255,255,255,255,2
55,  0,  0,  0,  0,  0,
255,255,255,255,2
55,  0,  0,  0,  0,  0,
0,  0,
0,255,255,255,

255,255,255,255,2
55,255,
0,255,255,255,255,
255,255,255,255,2
55,
   0,  0,  0,  0,  0,  0,
0,255,255,255,255,
255,255,255,255,2
55,

0,255,255,255,255,
255,255,255,255,
0,  0,  0,
0,255,255,255,
 255,159, 31,  0,  0,
0,159,223,255,255,
255,255,  0,  0,  0,
0,
   0,
,255,255,
0,255,127,127,127,

 127,  3,
3,127,127,127,127,
255,255,223,207,2
07,199,  3,  3,199,
   0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,
0,  0,  0,  0
)
implements
end.
```

Imágenes del Control Remoto

```
program glcd
include "Imagen"
dim X as word
dim Y as word
dim AUX as word

dim XW as string[5]
dim YW as string[5]

Sub procedure
Delay1S
  Delay_ms(1000)
End sub

main:
 Usart_Init(9600)
  ADCON0.7=1
  ADCON1 =
00000100
' Configuracion
analogica de
PORTA
  TRISC.0=0
TRISC.1=0
' PORTC.1 es
Salida
  TRISA.0=1
' PORTA.1 es
entrada
  TRISA.1=1
' PORTA.3 es
entrada  (ADC3)
  TRISC.2=0
  TRISC.3=0
  'Inicializacion del
grafico
  Glcd_Init(PORTB,
0, 1, 2, 3, 5, 4,
PORTD)
  Glcd_Fill(0x00)

Glcd_Image(Inicio)
  PORTC.2=1
  Delay1S
  Glcd_Fill(0x00)

Glcd_Image(Boton
es)

Glcd_Set_Font(@fo
nt5x7, 5, 7, 32)
'Choose font, see
__Lib_GLCDFonts.
c in Uses folder
  AUX="Z"
  PORTC.2=0
  while TRUE
    Gosub GetX
' Leemos la
pulsacion
    Gosub GetY
     If
((X>590)and(X<900
)and(Y>520)and(Y<
870)) then
        while
((X>590)and(X<900
)and(Y>520)and(Y<
870))
       Gosub GetX
       Gosub GetY
 wend

Glcd_Fill(0x00)

Glcd_Image(luz)
        gosub
LUCESAP

Glcd_Fill(0x00)

Glcd_Image(Boton
es)
    ELSE
                Gosub
GetX
                Gosub
GetY
            wend
           PORTC.2=1

Glcd_Fill(0x00)

Glcd_Image(ventan
a)
          gosub
VENTANASAC

Glcd_Fill(0x00)

Glcd_Image(Boton
es)
        ELSE
           If
((X>270)and(X<590
)and(Y>520)and(Y<
870)) then
              while
((X>270)and(X<590
)and(Y>520)and(Y<
870))
Gosub GetX
Gosub GetY
      wend

PORTC.2=1

Glcd_Fill(0x00)

Glcd_Image(Persia
na)
              gosub
PERSIANASAC
Glcd_Fill(0x00)

Glcd_Image(Boton
es)
          ELSE
((X>270)and(X<440
)and(Y>340)and(Y<
520)) then
'+CARTEL
PORTC.2=1
                END IF
AUX="W"
             ELSE
                If
((X>270)and(X<470
)and(Y>140)and(Y<
340)) then  '-
CARTEL
PORTC.2=1
                   IF
(AUX<>"X")THEN
Usart_write("X")
                   END
IF

AUX="X"
         ELSE
          If
((X>56)and(X<265)
and(Y>225)and(Y<
800))then 'OFF

while
((X>56)and(X<265)
and(Y>225)and(Y<
800))

Gosub GetX
Gosub GetY
wend

PORTC.2=1
      IF
(AUX<>"Y")THEN
Usart_write("Y")

END IF

AUX="Y"

ELSE

PORTC.2=0
 IF
(AUX<>"Z")THEN
Usart_write("Z")
END IF
AUX="Z"
End if
                End if
           End if
   wend

LUCESAP:
   DO
      Gosub GetX
      Gosub GetY
       If
((X>680)and(X<900
)and(Y>140)and(Y<
365)) then
'DISMINUIR 1-
          PORTC.2=1


          IF
(AUX<>"A")THEN

Usart_write("A")
          END IF
          AUX="A"
       ELSE
          If
((X>680)and(X<900
)and(Y>365)and(Y<
590)) then
'AUMENTAR 1+
          PORTC.2=1
          IF
(AUX<>"B")THEN

Usart_write("B")
          END IF
          AUX="B"
          ELSE
             If
((X>440)and(X<660
)and(Y>140)and(Y<
365)) then
'DISMINUIR 2-

PORTC.2=1
                IF
(AUX<>"C")THEN

Usart_write("C")
                END IF

AUX="C"
    ELSE
      If
((X>440)and(X<660
)and(Y>365)and(Y<
590)) then
'AUMENTAR 2+

PORTC.2=1
                   IF
(AUX<>"D")THEN

Usart_write("D")
                   END
IF
```

```
  AUX="D"
                ELSE
                  If
((X>205)and(X<410
)and(Y>120)and(Y<
365)) then
'APAGAR -

PORTC.2=1
                    IF
(AUX<>"F")THEN

Usart_write("F")

END IF
AUX="F"
ELSE
    If
((X>205)and(X<410
)and(Y>365)and(Y<
590)) then
'ENCENDER +
PORTC.2=1
IF
(AUX<>"E")THEN

Usart_write("E")

END IF

AUX="E"

ELSE

PORTC.2=0
Usart_write("Z")
END IF
AUX="Z"
END IF
IF
              END IF
            END IF
          END IF
        END IF
    LOOP
UNTIL((X>60)and(
X<200)and(Y>270)
and(Y<680))
    while
((X>60)and(X<200)
and(Y>270)and(Y<
680))
       Gosub GetX
       Gosub GetY
    wend
RETURN

VENTANASAC:
    DO
      Gosub GetX
      Gosub GetY
       If
((X>680)and(X<900
)and(Y>140)and(Y<
410)) then
'DISMINUIR 1-
          PORTC.2=1
          IF
(AUX<>"J")THEN

Usart_write("J")
          END IF
          AUX="J"
        ELSE
          If
((X>680)and(X<900
)and(Y>410)and(Y<
660)) then
'AUMENTAR 1+
          PORTC.2=1
          IF
(AUX<>"I")THEN

Usart_write("I")
          END IF

((X>440)and(X<660
)and(Y>140)and(Y<
410)) then
'DISMINUIR 2-

PORTC.2=1
                IF
(AUX<>"L")THEN

Usart_write("L")
              END IF
AUX="L"
            ELSE
              If
((X>440)and(X<660
)and(Y>410)and(Y<
660)) then
'AUMENTAR 2+
PORTC.2=1
                  IF
(AUX<>"K")THEN

Usart_write("K")
                    END
IF
AUX="K"
                ELSE
                  If
((X>195)and(X<420
)and(Y>140)and(Y<
410)) then
'DISMINUIR 3-
PORTC.2=1
                      IF
(AUX<>"N")THEN
Usart_write("N")
END IF
AUX="N"
ELSE
                       If
((X>195)and(X<420
)and(Y>410)and(Y<
660)) then
'AUMENTAR 3+
PORTC.2=1
IF
(AUX<>"M")THEN
Usart_write("M")
END IF
AUX="E"
ELSE
PORTC.2=0
IF
(AUX<>"Z")THEN

Usart_write("Z")
END IF
AUX="Z"
END IF
                    END
IF
              END IF
            END IF
          END IF
        END IF
    LOOP
UNTIL((X>60)and(
X<194)and(Y>270)
and(Y<680))
    while
((X>60)and(X<194)
and(Y>270)and(Y<
680))
       Gosub GetX
       Gosub GetY
    wend
RETURN
PERSIANASAC:
    DO
      Gosub GetX
      Gosub GetY
       If
((X>720)and(X<900
)and(Y>140)and(Y<
395)) then
(AUX<>"P")THEN

Usart_write("P")
          END IF
          AUX="P"
        ELSE
          If
((X>720)and(X<900
)and(Y>395)and(Y<
670)) then
'AUMENTAR
HORIZONTAL 1+
PORTC.2=1
            IF
(AUX<>"O")THEN
Usart_write("O")
            END IF
            AUX="O"
          ELSE
            If
((X>540)and(X<720
)and(Y>140)and(Y<
395)) then
'DISMINUIR
HORIZONTAL 2-
PORTC.2=1
              IF
(AUX<>"R")THEN

Usart_write("R")
            END IF
AUX="R"
          ELSE
            If
((X>540)and(X<720
)and(Y>395)and(Y<
670)) then
'AUMENTAR
HORIZONTAL 2+

PORTC.2=1
                IF
(AUX<>"Q")THEN

Usart_write("Q")
              END
IF
AUX="Q"
            ELSE
              If
((X>360)and(X<540
)and(Y>140)and(Y<
395)) then
'DISMINUIR
VERTICAL 3-
PORTC.2=1
                  IF
(AUX<>"T")THEN
Usart_write("T")
  END IF
AUX="T"
      ELSE
      If
((X>360)and(X<540
)and(Y>395)and(Y<
670)) then
'AUMENTAR
VERTICAL 3+
PORTC.2=1
IF
(AUX<>"S")THEN
Usart_write("S")
END IF

AUX="S"
ELSE
If
((X>180)and(X<360
)and(Y>140)and(Y<
395)) then
'AUMENTAR
VERTICAL 4+
PORTC.2=1
IF (AUX<>"V")THE
Usart_write("V")
```

```
 END IF
AUX="V"
ELSE
If
((X>180)and(X<360
)and(Y>395)and(Y<
670)) then
'AUMENTAR
VERTICAL 4+
PORTC.2=1
IF
(AUX<>"U")THEN
Usart_write("U")
END IF
AUX="U"
Usart_write("Z")
END IF
AUX="Z"
END IF
END IF
 END IF
                END IF
              END IF
            END IF
          END IF
        LOOP
UNTIL((X>60)and(
X<175)and(Y>290)
and(Y<770))
    while
((X>60)and(X<175)
and(Y>290)and(Y<
770))
        Gosub GetX
        Gosub GetY
    wend
RETURN

GetX:
'reading X
  TRISC.0=0
  TRISC.1=0
  PORTC.0=1
  PORTC.1=0
  Delay_ms(5)
  X = Adc_Read(0)
  'WordToStr(X,
XW)

'Glcd_Write_Text(X
W,10,4,1)
return

GetY:
'reading Y
  TRISC.0=0
  TRISC.1=0
  PORTC.0=0
  PORTC.1=1
  Delay_ms(5)
  Y = Adc_Read(1)
  'WordToStr(Y,
YW)

'Glcd_Write_Text(Y
W,10,6,1)
return

end.
```

Control de Motores

```
@ DEVICE
HS_OSC
@ DEVICE
BOD_OFF

DEFINE OSC 20
DEFINE
HSER_RCSTA 90h
DEFINE
HSER_TXSTA 24h
DEFINE
HSER_BAUD 9600
DEFINE
HSER_SPBRG 129
ADCON1=7
DATO VAR BYTE
TRISA=%0000110
0
TRISB=%1001100
0
TRISC=%1011000
1
TRISD=%1100110
0
TRISE=%0000011
0

'MOTOR1
MOTI1 VAR
PORTB.1
MOTD1 VAR
PORTB.2

'MOTOR2
MOTI2 VAR
PORTB.5
MOTD2 VAR
PORTB.6

'MOTOR3
MOTI3 VAR
PORTD.0
MOTD3 VAR
PORTD.1

'MOTOR4
MOTI4 VAR
PORTA.0
MOTD4 VAR
PORTA.1

'MOTOR5
MOTI5 VAR
PORTD.4
MOTD5 VAR
PORTD.5

'MOTOR6
MOTI6 VAR
PORTC.1
MOTD6 VAR
PORTC.2

'MOTOR7
MOTI7 VAR
PORTA.5
MOTD7 VAR
PORTE.0
CONTA1 VAR word
CONTA2 VAR word
CONTA3 VAR word
CONTA4 VAR word
CONTA5 VAR word
CONTA6 VAR word
CONTA7 VAR word
LED VAR PORTB.0
gosub parar
ON INTERRUPT
GoTo REGULAR
PIE1.5=1
INTCON=%110000
00
OPTION_REG.6=1

INICIO:
   IF(DATO="I" OR
DATO="J")THEN
      LED=1
      GOSUB
CONTMOTOR1
   ELSE
   IF(DATO="K" OR
DATO="L")THEN
      LED=1
      GOSUB
CONTMOTOR2
   ELSE
   IF(DATO="M"
OR
DATO="N")THEN
      LED=1
      GOSUB
CONTMOTOR3
   ELSE
   IF(DATO="O"
OR
DATO="P")THEN
      LED=1
      GOSUB
CONTMOTOR4
   ELSE
   IF(DATO="Q"
OR
DATO="R")THEN
      LED=1
      GOSUB
CONTMOTOR5
   ELSE
   IF(DATO="S" OR
DATO="T")THEN
      LED=1
      GOSUB
CONTMOTOR6
   ELSE
   IF(DATO="U"
OR
DATO="V")THEN
      LED=1
      GOSUB
CONTMOTOR7
   ELSE
IF(DATO="Y")THE
N
      LED=1
      GOSUB
CERRARTODO
   else
      LED=0
      GOSUB
PARAR
   endif
   ENDIF
   ENDIF
   ENDIF
   ENDIF
   ENDIF
   ENDIF
   ENDIF
GOTO INICIO
CONTMOTOR1:
   IF (dato="J" and
Conta1<500)THEN
         GOSUB
ADELANTE1
         PAUSE 5

conta1=conta1+1
   else
      IF (dato="I"
AND
Conta1>0)THEN
         GOSUB
Atras1
         PAUSE 5
conta1=conta1-1
      else
         GOSUB
PARAR
      endif
   endif
RETURN
CONTMOTOR2:
   IF (dato="L" AND
Conta2<500)THEN
```

```
      GOSUB
ADELANTE2
      PAUSE 5

conta2=conta2+1
   else
      IF (dato="K"
AND
Conta2>0)THEN
         GOSUB
Atras2
         PAUSE 5

conta2=conta2-1
      else
         GOSUB
PARAR
      endif
   endif
RETURN

CONTMOTOR3:
   IF (dato="N"
AND
    GOSUB
ADELANTE3
      PAUSE 5

conta3=conta3+1
   else
      IF (dato="M"
AND
Conta3>0)THEN
         GOSUB
Atras3
         PAUSE 5

conta3=conta3-1
      else
         GOSUB
PARAR
      endif
   endif
RETURN
CONTMOTOR4:
   IF (dato="P"
AND
Conta4<600)THEN
      GOSUB
ADELANTE4
      PAUSE 10
conta4=conta4+1
   else
      IF (dato="O"
AND
Conta4>0)THEN
         GOSUB
Atras4
         PAUSE 10

conta4=conta4-1
      else
         GOSUB
PARAR
      endif
   endif
RETURN
CONTMOTOR5:
   IF (dato="R"
AND
Conta5<600)THEN
      GOSUB
ADELANTE5
      PAUSE 10

conta5=conta5+1
   else
      IF (dato="Q"
AND
Conta5>0)THEN
         GOSUB
Atras5
         PAUSE 10

conta5=conta5-1
      else
         GOSUB
PARAR
      endif
   endif
RETURN
CONTMOTOR6:
   IF (dato="T" AND
Conta6<600)THEN
IF(CONTA4>=50)T
HEN
      GOSUB
ADELANTE6
      PAUSE 5
conta6=contA6+1
    ENDIF
   else
      IF (dato="S"
AND
Conta6>0)THEN
         GOSUB
Atras6
         PAUSE 5

conta6=conta6-1
      else
         GOSUB
PARAR
      endif
   endif
RETURN

CONTMOTOR7:
   IF (dato="V"
AND
Conta7<600)THEN

IF(CONTA5>=50)T
HEN
      GOSUB
ADELANTE7
      PAUSE 5
conta7=conta7+1
    ENDIF
   else
      IF (dato="U"
AND
Conta7>0)THEN
         GOSUB
Atras7
         PAUSE 5

conta7=conta7-1
      else
         GOSUB
PARAR
      endif
   endif
RETURN

CERRARTODO:
   GOSUB PARAR
   WHILE
(Conta1>0)
      GOSUB
Atras1
      PAUSE 5
conta1=conta1-1
   WEND
   GOSUB PARAR
   WHILE
(Conta2>0)
      GOSUB
Atras2
      PAUSE 5
conta2=conta2-1
   WEND
   GOSUB PARAR
   WHILE
(Conta3>0)
      GOSUB
Atras3
      PAUSE 5

conta3=conta3-1
   WEND
   GOSUB PARAR
   WHILE
(Conta4>0)
      GOSUB
Atras4
      PAUSE 10
conta4=conta4-1
   WEND
   GOSUB PARAR
   WHILE
(Conta5>0)
      GOSUB
Atras5
      PAUSE 10
   conta5=conta5-1
   WEND
   GOSUB PARAR
   WHILE
(Conta6>0)
      GOSUB
Atras6
      PAUSE 5
   conta6=conta6-1
   WEND
   GOSUB PARAR
   WHILE
(Conta7>0)
      GOSUB
Atras7
      PAUSE 5
conta7=conta7-1
   WEND
      GOSUB
PARAR
RETURN
PARAR:
RETURN
'MOTOR1
ADELANTE1:
   MOTI1=0
   MOTD1=1
RETURN
ATRAS1:
   motI1=1
   motD1=0
RETURN
'MOTOR2
ADELANTE2:
   MOTI2=0
   MOTD2=1
RETURN
ATRAS2:
   motI2=1
   motD2=0
RETURN
'MOTOR3
ADELANTE3:
   MOTI3=0
   MOTD3=1
RETURN
ATRAS3:
   motI3=1
   motD3=0
RETURN
'MOTOR4
ADELANTE4:
   MOTI4=0
   MOTD4=1
RETURN
ATRAS4:
   motI4=1
   motD4=0
RETURN
'MOTOR5
ADELANTE5:
   MOTI5=0
   MOTD5=1
RETURN
ATRAS5:
   motI5=1
   motD5=0
RETURN
'MOTOR6
ADELANTE6:
   MOTI6=0
```

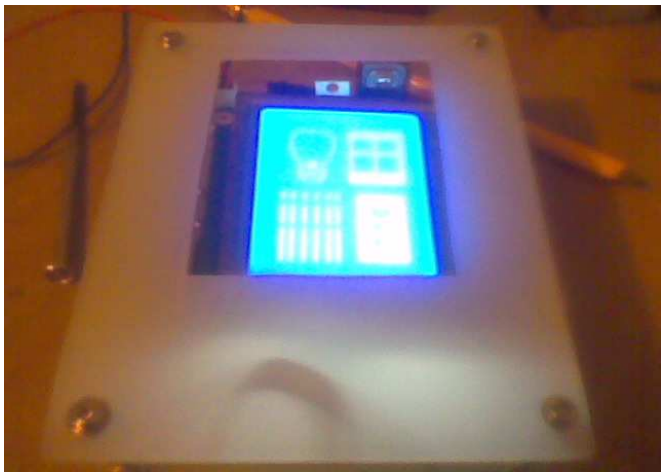
```
   MOTD6=1
RETURN
ATRAS6:
   motI6=1
   MOTD6=0
RETURN

'MOTOR7
ADELANTE7:
   MOTI7=0
   MOTD7=1
RETURN
ATRAS7:
   MOTI7=1
   MOTD7=0
RETURN
Disable
REGULAR:
  IF PIR1.5=1 Then
'RCIF=1
    HSerin[DATO]
  EndIF
   PIE1.5=1
INTCON=%11000000
  Resume
Enable
END
```

Control de Luces

```
@ DEVICE
HS_OSC
@ DEVICE
BOD_OFF
DEFINE OSC 20
DEFINE
HSER_RCSTA 90h
DEFINE
HSER_TXSTA 24h
DEFINE
HSER_BAUD 9600
DEFINE
HSER_SPBRG 129
ADCON1=7
TRISA=%00001111
TRISB=%00000001
TRISD=0
DATO VAR BYTE
LOW PORTD.0
VELOCIDAD1 VAR
WORD
ENCEN_APAGAR1
VAR PORTB.3
VELOCIDAD1=8200
LED1 var PORTB.1
VELOCIDAD2 VAR
WORD
ENCEN_APAGAR2
VAR PORTB.4
VELOCIDAD2=8200
LED2 var PORTB.5
BAND VAR WORD
VAL VAR WORD
BAND=0
VAL=0
FOCO VAR
PORTD.0
LED3 var PORTD.1
LED var PORTD.2
ON INTERRUPT
GoTo REGULAR
PIE1.5=1
INTCON=%11010000
OPTION_REG.6=1
INICIO:
   if (dato="A" or
dato="B" or
dato="C" or
dato="D" or
dato="E" or
dato="F" or
dato="Y")then
     led=1
   else
     led=0
     led1=0
     led2=0
     led3=0
   endif
'DIMMER1
   IF dato="B" Then
'SUBE
      IF
velocidad1<8200
Then
velocidad1=velocidad1+2
        led1=1
        led2=0
        led3=0
      EndIF
   EndIF
   IF dato="A" Then
'BAJA
     IF
(velocidad1>0)
Then
velocidad1=velocidad1-2
        led1=1
        led2=0
        led3=0
     EndIF
  EndIF

'DIMMER2
    IF dato="D"
Then   'SUBE
      IF
velocidad2<8200
Then

velocidad2=velocidad2+2
        LED2=1
        led1=0
        led3=0
      EndIF
   EndIF
   IF dato="C" Then
'BAJA
     IF
(velocidad2>0)
Then
velocidad2=velocidad2-2
        led2=1
        led1=0
        led3=0
     EndIF
  EndIF

'FOCO
  IF dato="F" Then
'ON
     foco=1
     led3=1
     led1=0
     led2=0
  endif
  IF dato="E" Then
'OFF
      FOco=0
     led3=1
     led1=0
     led2=0
  endif
'APAGAR TODO
 IF dato="Y" Then
'OFF
      foco=0

velocidad1=8200
velocidad2=8200
      led1=1
      led2=1
      led3=1
 endif
GoTo inicio
Disable
REGULAR:
  IF PIR1.5=1 Then
'RCIF=1
    HSerin[DATO]
  EndIF
  IF
(VELOCIDAD1=VELOCIDAD2)THEN
    BAND=0
    VAL=0
  ELSE
  IF
(VELOCIDAD1>VELOCIDAD2)THEN
    BAND=1
VAL=VELOCIDAD1-VELOCIDAD2
  ELSE
    BAND=2
VAL=VELOCIDAD2-VELOCIDAD1
  ENDIF
  ENDIF
IF(BAND=0)THEN
      Low
ENCEN_APAGAR1
      Low
ENCEN_APAGAR2
      PauseUs
VELOCIDAD1
      High
ENCEN_APAGAR1
      High
ENCEN_APAGAR2
   ELSE
IF(BAND=1)THEN
      Low
ENCEN_APAGAR1
      Low
ENCEN_APAGAR2
      PauseUs
VELOCIDAD2
      High
ENCEN_APAGAR2
      PauseUs
VAL
  High
ENCEN_APAGAR1
      ELSE
      Low
ENCEN_APAGAR1
      Low
ENCEN_APAGAR2
      PauseUs
VELOCIDAD1
      High
ENCEN_APAGAR1
      PauseUs
VAL
      High
ENCEN_APAGAR2
      ENDIF
   ENDIF
  PIE1.5=1
INTCON=%11010000
   Resume
Enable
End
```

Anexo G.

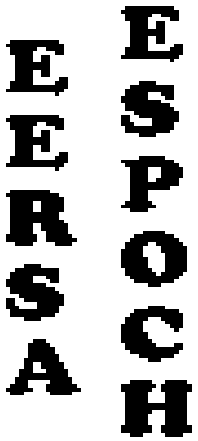
MANUAL DE USUARIO DE CONTROL REMOTO.



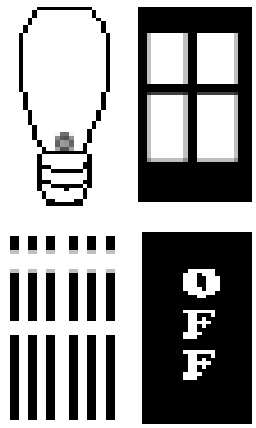
Control Remoto de Sala Inteligente

Este manual esta diseñado para el correcto manejo del control remoto, el cual posee una pantalla táctil, este control funciona con una batería de 9V la que es fácil de cambiar.

1. Encender el control remoto por medio del switch que se encuentra en la parte superior.
2. Aparecerá el siguiente logotipo que durara unos segundos:



3. Automáticamente aparecerá el siguiente menú principal:



 Menú Luces

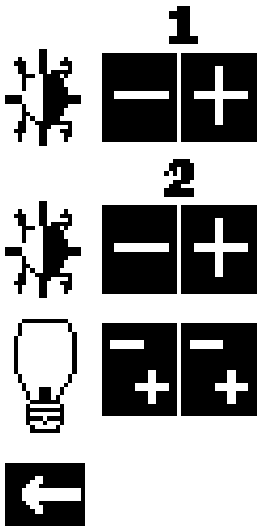
 Menú Ventanas

 Menú Persianas

 Apagar todo el Sistema

4. Presionar sobre la opción del sistema que desea variar.

5. Si presiono sobre el menú luces:



Donde:

 Dimmers

 Luz Fija

 Incrementar la luminosidad

 Disminuir la luminosidad

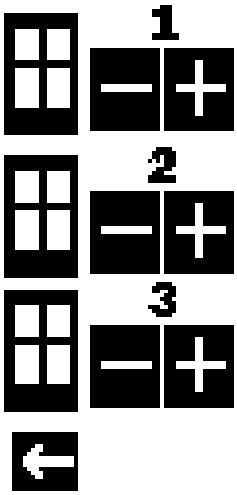
 Encender o apagar la Luz

 Regresar al Menú Principal

Como se puede observar al lado derecho de los dimmers están ubicados los signos "+" y "-", donde en su lado superior está el número que indica que dimmer va ha variar. Al estar presionando sobre el signo "+" de la opción dimmer ira incrementando la luminosidad de los focos, y si tiene presionado el signo "-" ira disminuyendo la luminosidad de los focos que están ubicados al rededor de la sala.

Al lado de la luz fija se puede ver dos botones que tienen el mismo símbolo, como la luz se puede encender o apagar desde la pared o desde el control remoto, para encenderla o apagarla se debe presionar uno de los dos botones, en caso de no ocurrir nada se debe presionar el otro botón.

6. Si presiono sobre el menú Ventanas:



Donde:

 Abrir Ventana

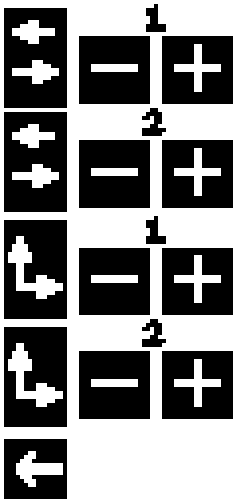
 Cerrar ventana

 Regresar al Menú Principal

Como se puede observar sobre los juegos de signos esta ubicado el numero de la ventana que va ha manipular.

Si presiona el signo “+” se ira abriendo la ventana escogida y el signo “-” la ira cerrando.

7. Si presiono sobre el menú Persianas:



Donde:

 Recorrer Persianas

 Giro de 90º de Persianas

 Abrir

 Cerrar

 Regresar al Menú Principal

Como se puede observar sobre los juegos de signos esta ubicado el numero de la persiana que va ha manipular.

Si presiona el signo “+” se ira abriendo la persiana escogida y el signo “-” la ira cerrando.

Si escogió recorrer abrirá o cerrara.

Y si escogió Giro de 90º abrirá o cerrara los 90º progresivamente.

Nota: Se recomienda manipular el control con un pequeño esfuerzo, ya que la pantalla es sensible al tacto, al momento que usted activa la opción deseada, se encenderá una pequeña luz al lado de la pantalla.

BIBLIOGRAFÍA

LIBROS

- ANGULO, J.M. Robótica Práctica: tecnología y aplicaciones. 4ª ed. Madrid: Paraninfo, 1996.
- ANGULO, J.M.; EUGENIO, M. y ANGULO I. Microcontroladores Pic: la solución en un chip. Madrid: Paraninfo, 1997.
- ANGULO, J.M.; ROMERO, S. y ANGULO, I. Microbótica. Madrid: Paraninfo, 1999.
- BLAKE, R. Sistemas Electrónicos de Comunicaciones. 2da ed. Inglaterra: Thomson Learning, 2004.
- TAYLOR, P.M. Control Robótico. España: CEAC, 1992.
- TORRES, F.; POMARES, J. y GIL, P. Robots y Sistemas Sensoriales. New Jersey: Prentice Hall, 2002.

BIBLIOGRAFÍA DE INTERNET

Domotica

- http://www.monografias.com/trabajos35/domotica/domotica.shtml
  2009/10/02
- http://www.energiasdelsur.com/aplicaciones/domotica-inmotica.html
  2009/10/20
- http://www.energiasdelsur.com/aplicaciones/separacion-de-residuos.html
  2009/10/25

Microcontroladores

- http://www.microchip.com
  2009/10/10
- http://www.galeon.com/microchip/
  2009/10/18
- http://www.todopic.com.ar
  2009/10/18
- http://todopic.mforos.com/8826/2007482-pic16f877-y-lcd-grafico-wintek/
  2009/11/02
- http://todopic.mforos.com/6510/3522308-problema-con-el-pic-y-los-voltajes/
  2009/11/02
- http://micros.mforos.com/?cat=351376
  2009/11/02
- http://www.jvmbots.com/viewtopic.php?t=684&postdays=0&postorder=asc&start=0
  2009/11/20
- http://www.freewebs.com/glafebre/proyectos.htm
  2009/11/25

- http://www.unicrom.com/Tut_MotorCC.asp
  2009/11/25

Zigbee

- http://todopic.mforos.com/6510/3522308-problema-con-el-pic-y-los-voltajes/
  2009/10/02
- http://www.blogelectronica.com/zigbee-maxstream-sdk/
  2009/10/20
- http://www.blogelectronica.com/category/tecnologias/zigbee/
  2009/10/25
- http://www.maxstream.net/
  2009/11/05
- http://www.blogelectronica.com/redes-zigbee-i-introduccion/
  2009/11/15
- http://www.jvmbots.com/viewtopic.php?t=684&postdays=0&postorder=asc&start=0
  2009/11/25
- http://www.unicrom.com/Tut_MotorCC.asp
  2009/11/26
- http://www.monografias.com/trabajos/lacomunica/lacomunica.shtml
  2009/11/26