UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

TESIS DE GRADO

"MODELO DE SISTEMA EMBEBIDO EN LA COMUNICACIÓN ENTRE AUTÓMATAS Y TELEFONÍA MÓVIL"

PARA OPTAR AL TITULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: INGENIERIA DE SISTEMAS INFORMATICOS

POSTULANTE: UNIV. PAOLA SDENKA HELGUERO DURÁN

TUTOR: LIC. GERMÁN HUANCA TICONA

REVISOR: LIC. GROVER RODRIGUEZ

LA PAZ – BOLIVIA

2011

DEDICATORIA

Al pilar fundamental de mi vida, de profesión maestra, mi mamá María Nieves Durán, por acompañarme en cada uno de mis locuras que he emprendido, ¡Por ser mi más ferviente hincha!

Especialmente por tus sabios consejos, por tu apoyo, por tu amor, por estar a mi lado con una comprensión a prueba de todo.

Gracias por creer en mí.

AGRADECIMIENTOS

La autora expresa sus agradecimientos:

A: Dios por darme una vida llena de bendiciones, la capacidad y fuerza para asimilar conocimientos desprendidos de las situaciones adversas que se han presentado en el camino.

A: Nuestra casa de estudios, Universidad Mayor de San Andrés, Facultad Ciencias Puras y Naturales, carrera de Informática, que me dio la oportunidad de ser parte de ella y permitir que logre mis sueños universitarios.

A: Mi tutor Lic. Germán Huanca Ticona, por su comprensión, dedicación y guía en el desarrollo y conclusión de la presente tesis.

A: Mi revisor Grover Rodríguez Ramirez, por su apoyo, paciencia y dedicación al realizar las debidas observaciones y correcciones en el desarrollo de la presente tesis.

A: Todos los docentes por su enseñanza realizada a lo largo de mi estadía en la carrera.

A: La Sociedad Científica, que me guiaron en la parte del conocimiento de electrónica, que fueron valiosos para este trabajo.

A: Mi familia, por brindarme un apoyo incondicional en cada momento de mi vida, especialmente a mi ch'ulla Ximena, gracias por todo te adoro.

A: Ti, que te puedo decir amor, muchas gracias por caminar a mi lado durante todo este tiempo mostrándome; con una sonrisa que aún se puede.

A: Mis amigos y amigas que en una y otra forma colaboraron en la realización del presente trabajo

A: La persona que en este momento se da la molestia de leer este trabajo. ¡Espero que te sea de gran ayuda!.

RESUMEN

Diseño de un sistema embebido de conectividad inalámbrica utilizando bluetooth que permita un flujo de datos, entre un teléfono móvil, un equipo que actúa como centro de control y un autómata (robot con movimientos básicos)

Se utilizo un equipo de telefonía móvil en el cual se desarrollo una aplicación en J2ME que permite el control del sistema.

El equipo de telefonía móvil se conecta al equipo central mediante un conector bluetooth que recibe el flujo de datos en la PC para ser procesados en una aplicación desarrollada en Visual Basic.

Finalmente el equipo central envía los datos a una placa conectada al puerto serial que controla los movimientos del autómata (robot).

Se obtuvo un sistema con un nivel básico de comunicación inalámbrica entre el equipo de telefonía móvil y un autómata con un tiempo de respuesta despreciable lo que permite realizar funciones de mayor complejidad.

El sistema embebido de conectividad inalámbrica aprovecha la capacidad máxima de un equipo de telefonía móvil a través de la tecnología bluetooth permitiendo un amplio campo de aplicación porque es un sistema embebido muy atractivo para seguir investigando y desarrollando nuevas aplicaciones en base al estudio realizado.

ABSTRACT

Design of an embedded system using Bluetooth wireless connectivity to enable data flow between a mobile phone, a computer acting as a control center and PLC (robot basic movements).

They use a mobile computer in which we developed a J2ME application that allows control of the system.

The mobile computer connects to the central computer via a bluetooth connector that receives the data stream on the PC to be processed in an application developed in Visual Basic.

Finally, the central computer sends data to a plate connected to the serial port that controls the movements of the automaton (robot).

It was a system with a basic level of wireless communication between the mobile computer and a robot with a negligible response time allowing for more complex functions.
The wireless embedded system uses the maximum capacity of a mobile computer via Bluetooth technology allows a wide range of applications because it is an embedded system very attractive for further research and develop new applications based on the study.

ÍNDICE GENERAL

CONTENIDO

INDICE DE ABREVIATURAS

API	Application Programming Interface
CISC	Computador con un conjunto complejo de instrucciones
CLDC	Configuración de dispositivos limitados con conexión
EEPROM	Memoria de solo lectura programable y borrable eléctricamente
FH	Salto de Frecuencia
FLASH	Memoria no volátil de bajo consumo
HSERROUT	Salida serial de hardware asíncrono
J2ME	Java 2 Platform, Micro Edition
MIDP	Mobile Information Device Profile
PAN	Red de area personal
PIC	Periphetical Interface Controller
RAM	Memoria de acceso aleatorio
RFCOMM	Protocolo Provee servicio de emulador del RS232
RISC	Computadores de juego de instrucciones reducido
SISC	Computadores de conjunto de instrucciones especifico
SPP	Perfil puerto serie
UCP	Unidad central de proceso
USART	Transmisor/ recpetor asíncrono universal
UART	Adaptador de comunicación serie asíncrona
UUID	Indetificador único universal dispositivo

CAPITULO I

Resumen
En el presente capitulo se explica el contexto de la tesis en curso; se menciona la información necesaria para entender el objetivo final y el medio por el cual se desarrolla la presente tesis.

1.1. INTRODUCCIÓN

La posibilidad de conexión inalámbrica abre la puerta a la creación de nuevos servicios donde la movilidad es el factor clave. También los enlaces inalámbricos facilitan la conexión de muchos periféricos y reducen la cantidad de cables haciendo el montaje de equipos más sencillo y el ambiente de trabajo menos limitante.
La temática de esta investigación aborda la conectividad entre varios dispositivos y la forma de interactuar entre ellos.

La idea es demostrar que no importa qué tipo de dispositivo se use con tal de que posea interfaces que nos permitan un flujo de datos, un método de programación y un receptor. Esto hace posible que podamos enviar o recibir datos en cualquier dirección y así sacarle provecho a esta mezcla de tecnologías.

En este caso trabajaríamos con microcontroladores y receptores inalámbricos, con adaptadores de USB a Bluetooth y lo importante con equipos de celulares.

Aporte a la Sociedad

La denominación de Sistemas Embebidos refleja que son parte integral (interna) del sistema, y en general son dispositivos utilizados para controlar o asistir la operación de diversos equipamientos, estos dispositivos pueden controlar equipos, operación de maquinarias o plantas industriales completas.

Lo interesante de que un sistema sea **"embebido"** es que puede estar de tal forma incrustado, puede quedar tan oculto a nuestros ojos, que la presencia de tales "chips" no resulte nada obvia quien lo mira.

En la actualidad debido al crecimiento de las aplicaciones de los equipos de telefonía celular podemos aprovechar la capacidad de un celular creando una interfaz capaz de monitorear y manejar autómatas, en lugares donde la seguridad física de los seres humanos este en peligro, brindando una herramienta de ayuda al hombre.

1.2. ANTECEDENTES.

La tecnología de los microprocesadores y la fabricación de circuitos integrados están cambiando rápidamente. En la actualidad los microcontroladores más complejos contienen unos 10 millones de transistores. Se provee que en el 2000 los microcontroladores avanzados contengan más de 50 millones de transistores y unos 800 en el 2010.

Se entiende por controlador lógico programable o autómata programable, a toda máquina electrónica para controlar en tiempo real y en medio industrial procesos secuenciales.

Esta definición se ha quedado un poco desfasada, ya que han aparecido los micro-plc's, destinados a pequeñas necesidades y al alcance de cualquier persona.

Un autómata programable suele emplearse en procesos industriales que tengan una o varias de las siguientes necesidades. Tal y como dijimos anteriormente, esto se refiere a los autómatas programables industriales, dejando de lado a los pequeños autómatas para uso más personal (que se pueden emplear, para automatizar procesos en el hogar, como la puerta de una cochera o las luces de la casa).

1.2.1. Telefónica Celular

La tecnología de los celulares, tiene una vida de más o menos dos décadas en nuestro país y han tomado gran terreno en el ámbito económico, por lo cual el desarrollar aplicaciones para equipos celulares es algo que se debe asumir de forma correcta por el potencial que representa "negocio".

1.2.2. Redes Inalámbricas

Existe una inmensa diversidad de propuestas de conexión inalámbrica que se disputa en el mercado de protocolos y productos. Esta variedad ha particularizado la demanda a tal punto que cada protocolo se especializa en entornos de conexión por su alcance. Es decir las redes WPAN (Wireless Personal Área Network) se utilizan básicamente en sincronización de equipos o conexión de periféricos que se encuentran a muy corta distancia normalmente superando los 10 metros.

En este ámbito se sitúa Bluetooth en particular se presenta como un posible estándar en las redes inalámbricas de sensores. Estas redes altamente interconectadas tienen dispositivos que deben tener un bajo consumo energético y capacidad de transmitir información. Es importante familiarizarse con los distintos protocolos de interconexión inalámbrica para aprender su configuración e instalación.

1.2.3.Tecnología Inalámbrica

Una de las tecnología más prometedoras y discutidas en esta década es la de poder comunicar computadoras mediante tecnología inalámbrica. La conexión de computadoras mediante Ondas de Radio o Luz Infrarroja, actualmente está siendo ampliamente investigada. Las Redes Inalámbricas facilitan la operación en lugares donde la computadora no puede permanecer en un solo lugar, como en almacenes o en oficinas que se encuentren en varios pisos. Pero la realidad es que esta tecnología está todavía en proceso y se deben resolver varios obstáculos técnicos y de regulación antes de que las redes inalámbricas sean utilizadas de una manera general en los sistemas de cómputo de la actualidad.

En nuestro país Bolivia el tema de Microcontroladores está dando buenos resultados y hay algunas aplicaciones desarrolladas. Están presentes en nuestra casa, nuestro trabajo y nuestra vida diaria. Se pueden encontrar controlando el funcionamiento de los ratones (mouse) y los teclados de las computadoras, en los teléfonos, en los microondas y especialmente en nuestros televisores del hogar.

Cuando hablamos sobre la comunicación inalámbrica entre microcontroladores podemos decir que al parecer aún hay pocas instituciones o universidades que están investigando sobre ello, es por eso que es conveniente realizar un estudio profundo sobre este tema para obtener beneficios tecnológicos y crear un producto de alto interés.

Aportes Similares

- **Manipuladores y Robots móviles** Anibal Ollero Barcelona – Marcombo 2001. Identifica y analiza los diferentes componentes de un robot, especialmente la estructura mecánica y la unidad de control y programación. Conocer las técnicas fundamentales para su modelizado y control. Evaluar sus capacidades en un entorno social.

- **Desarrollo de un interfaz Abierta que facilite la comunicación entre dispositivos Inalámbricos mediante la Tecnología BLUETOOTH** J Fernández y Terán Jardín, Ingeniería Informática UCAB 2002. El siguiente trabajo tiene como objetivo brindar un aporte tecnológico en el área de las comunicaciones inalámbricas. Se logra realizar un prototipo de interfaz de software basado en una arquitectura cliente servidor que permite el manejo e intercambio de archivos entre dispositivos que manejen la tecnología bluetooth.

- **Conectividad Inalámbrica entre microcontroladores y sus aplicaciones prácticas en Base de Datos.** Guido Roberto Bascuñán Leonelli Universidad Católica de Temuco 2005. Concluye que en su País Ecuador el tema de los microcontroladores se está recién conociendo y por ende hay miles de

aplicaciones para estos que aún no se han desarrollado. Por otro lado la comunicación inalámbrica está llegando de poco en poco en las que son las redes 802.11b y 802.11g, que son las que utiliza para demostrar su investigación.

- **Aplicación de GPRS para arquitecturas de control de robots** Nelson David Muñoz Universidad de Antioquia Mayo 2006. Se describe de un sistema que permite teleoperar un robot a través de teléfonos celulares. Se emplea la red de telefonía celular existente en el país, utilizando el Servicio General de Paquetes por Radiofrecuencia GPRS. De esta manera se permite a los usuarios del sistema un rango más amplio de movilidad.

1.3. DEFINICIÓN DEL PROBLEMA

- En nuestro territorio no existen tecnologías móviles de información que hagan énfasis en generar aplicaciones totalmente embebidas que sean capaces de controlar sistemas remotos.
- En nuestro país existe la carencia de aplicaciones que monitoreen autómatas.
- El uso de las redes inalámbricas ha ido incrementando en la actualidad, se ha extendido y globalizado al mundo, las configuraciones de los mismos no siempre son de las mejores.
- Existen localidades en nuestro territorio donde prácticamente es inexistente los medios de comunicación, ya sean cableados fijos, teléfonos móviles y/o servicios de red inalámbrica.
- Carencia en teleoperar máquinas por medios de órdenes dadas por sistemas embebidos o reportar el comportamiento de dichos sistemas a través del tiempo.

- Poco interés de estudiantes que generen aplicaciones en el campo de robótica, domótica, la seguridad, el control industrial y el sector agroindustrial entre otros.
- Falta de herramientas de seguridad industrial a nivel computacional que brinden seguridad física a los seres humanos en situaciones peligrosas.
- Finalmente poca información de sistemas embebidos que ayuden a la comunicación entre autómatas y teléfonos móviles, capaz de aumentar el monitoreo.

Por lo cual formulamos una interrogante que ayudará a definir los objetivos que fundamentará el siguiente trabajo:

¿Es posible desarrollar un modelo de sistema embebido capaz de monitorear desde un teléfono móvil a un autómata utilizando conexión inalámbrica?

1.4. JUSTIFICACIÓNES

1.4.1. JUSTIFICACIÓN ECONÓMICA

Al trabajar con software libre, cualquier desarrollador puede disponer de su código fuente, lo cual implica independencia total en cuestión de licencias y de desarrollo tecnológico, un costo nulo de adquisición, garantizando sostenibilidad técnica y tecnología en el tiempo.

1.4.2. JUSTIFICACIÓN TÉCNICA

Constituye un aporte a las técnicas de uso de los microcontroladores, las computadoras, telefonía todos conectados a una red inalámbrica pero estos recursos no están aprovechados al máximo, pues solo se utilizan como almacén de datos. Viendo este aspecto el modelar el sistema dará un nuevo y buen uso a los componentes tecnológicos mencionados.

1.4.3. JUSTIFICACIÓN SOCIAL

En el desarrollo del modelo, se pretende satisfacer las necesidades de los usuarios, quienes tropiezan con el problema de trabajar con equipamiento en maquinara e insumos de nivel toxico o alto nivel de peligrosidad.

1.5. OBJETIVOS

1.5.1. OBJETIVO GENERAL

Construir un modelo de sistema embebido de control, capaz de monitorear autómatas, utilizando la telefonía móvil y las tecnologías inalámbricas.

1.5.2. OBJETIVOS ESPECÍFICOS

- Diseñar e implementar una interfaz de comunicación inalámbrica entre un teléfono móvil y un autómata, utilizando un servidor de recepción de datos a través de un sistema embebido.
- Diseñar e implementar una interfaz de un sistema básico de recepción y control en el equipo central (servidor) que envié datos codificados al autómata mediante conexión inalámbrica.
- Realizar un modelo de diseño electrónico para el autómata incorporando un sistema embebido de recepción de datos.
- Probar resultados el modelo desarrollado.

1.6. HIPÓTESIS

El modelo de sistema embebido permite el monitoreo entre el teléfono móvil y el autómata utilizando conexión inalámbrica.

La hipótesis planteada corresponde al tipo de: **"Formulación causa y efecto"**.

La variable independiente es: **el uso del teléfono celular y el autómata.**

La variable dependiente es: **la ejecución del modelo de sistema embebido mediante redes inalámbricas.**

1.7. ALCANCES Y APORTES.

1.7.1. ALCANCES Y LIMITES

Entre los alcances y límites que se tiene en el presente trabajo de tesis tenemos:

- Se presentará el modelo de construcción de un sistema embebido basada en mecanismos propuestos en el presente trabajo de tesis para la comunicación entre autómatas y teléfonos celulares a través de una conexión inalámbrica.
- Para el modelado de la aplicación, se partirá del supuesto que no se cuenta con un mecanismo interfaz entre los autómatas y los teléfonos celulares.
- Además es importante hacer notar que el uso de conexiones inalámbricas solo se alcanza en los casos más ideales, es decir, que no exista degradación (disminución progresiva) de la señal.

1.7.2. APORTES

En cuanto a los aportes que se brindan en el presente trabajo:

- Brindará beneficios en la seguridad física, en el control de autómatas, agilizará los procesos mediante alertas y monitoreo constante.
- Se publicará los procesos de construcción y elaboración del modelo de sistema embebido adjunto a todas las bibliotecas y programas que fueron elaborados, esto como antecedentes para futuros trabajos que se especialicen en el área de conexión inalámbrica entre autómatas y teléfonos celulares.

- Se dejará como antecedente el modelo, para aplicaciones de similar índole o referencias en la construcción de herramientas para interconectividad de las redes inalámbricas, autómatas y teléfonos celulares.

- Se brindaran sugerencias y listados de evaluación, que serán útiles para el estudiante especializado en la conectividad inalámbrica entre teléfonos celulares y autómatas.

1.8. METODOLOGIA

Antes de señalar los métodos o metodologías a utilizar es necesario conocer la diferencia entre estos dos conceptos:

El MÉTODO es el procedimiento para lograr los objetivos. METODOLOGIA es el estudio del método" (Sirpa, 07)

El presente trabajo estará enmarcado en el desarrollo de una investigación científica, la metodología es el método científico, que es camino de la observación, la interpretación y la comparación que sigue la ciencia para encontrarse a sí misma o mejorarse

Así como en la aplicación de los conocimientos teóricos y el conocimiento empírico sobre el tema en cuestión y además se utilizará métodos y técnicas para el desarrollo del modelo, revisiones bibliográficas y análisis de las observaciones, interpretación de las observaciones y citas de referencias de internet.

Como método a utilizar y general se utilizará el deductivo – inductivo, lo deductivo está ligado más al razonamiento que es lo abstracto, lo inductivo es más empírico – observacional, tomando en cuenta etapas de observación experimentación y emisión de observaciones, además de técnicas de verificación de resultados, discusión de resultados.

Para su planificación y desarrollo se seguirá los siguientes pasos metodológicos:

- Recolección y procesamiento de la información requerida
- Evaluación de las características de la información requerida
- Diseño de los elementos propios del sistema embebido
- Formulación del programa de computadora y dispositivos
- Diseño y experimentos del modelo propuesto
- Prueba y resultados del sistema.

El modelo de sistema embebido involucra el diseño para la comunicación inalámbrica y su consecuente implementación en un microcontrolador pic.

Entre las herramientas a emplear como base del modelo en el aspecto tecnológico son:

a) Software de aplicación OPEN.

b) Uso de protocolos RFCOMM

c) Servidor SPP – Equipo central de control

d) Microcontroladores PIC

CAPITULO II

Resumen
En el presente capítulo brinda el marco de referencia y los conceptos fundamentales para el desarrollo de la siguiente tesis.

2.1 COMUNICACIÓN INALÁMBRICA

La conexión inalámbrica de computadoras mediante Ondas de Radio o Luz Infrarroja, actualmente está siendo ampliamente investigada. Las Redes Inalámbricas facilitan la operación en lugares donde las computadoras no pueden permanecer en un solo lugar, como en universidades o en oficinas [Gonzales, 04].

Actualmente el uso de equipos inalámbricos va aumentando enormemente, ya sea el uso de teléfonos celulares, controles a distancia, alarmas, agendas electrónicas, periféricos para computadores como teclados, ratones, joystick, etc. [Gonzales, 04]

Sin embargo se pueden mezclar las redes cableadas y las inalámbricas, y de esta manea generar una "Red Híbrida[1]" y poder resolver los últimos metros hacia la estación. Se puede considerar que el sistema cableado sea parte principal y la inalámbrica le proporcione movilidad adicional al equipo y el operador se pueda desplazar con facilidad dentro de un almacén o una oficina [Aguirre, 08].

2.1.1 DISPOSITIVOS DE COMUNICACIÓN INALÁMBRICA

Los componentes inalámbricos se utilizan para la conexión a redes en distancias que hacen que el uso de adaptadores de red y opciones de cableado estándares

[1] Internetwork compuesta por más de un tipo de tecnología de red, incluyendo LAN y WAN

sea técnica o económicamente imposible. Las redes inalámbricas están formadas por componentes inalámbricos que se comunican con LAN's[2]. [ILUS, 11].

Excepto por el hecho de que no es un cable quien conecta los equipos, una red inalámbrica típica funciona casi igual que una red con cables: se instala en cada equipo un adaptador de red inalámbrico con un transceptor[3]. Los usuarios se comunican con la red igual que si estuvieran utilizando un equipo de cables. [ILUS, 11]

Existen técnicas habituales para comunicación inalámbrica:

2.1.1.1 COMUNICACIÓN POR INFRARROJOS

Funciona utilizando un haz de luz infrarroja que transporta los datos entre dispositivos. Debe existir visibilidad directa entre los dispositivos que transmiten y los que reciben; si hay algo que bloquee la señal infrarroja, puede impedir la comunicación. Estos sistemas deben generar señales muy potentes, ya que las señales de transmisión débiles son susceptibles de recibir interferencias de fuentes de luz, como ventanas. [ILUS, 11].

2.1.1.2 COMUNICACIÓN DE VÍA RADIO BANDA ESTRECHA

El usuario sintoniza el transmisor y el receptor a una determinada frecuencia. La radio en banda estrecha no requiere visibilidad directa porque utiliza ondas de radio. Sin embargo la transmisión vía radio en banda estrecha está sujeta a interferencias de paredes de acero e influencias de carga. La radio en banda estrecha utiliza un servicio de suscripción. Los usuarios pagan una cuota por la transmisión de radio [ILUS, 11].

2.1.1.3 COMUNICACIÓN DE RADIOFRECUENCIA.

[2] Es un conjunto de equipos que pertenecen a la misma organización y están conectados dentro de un área geográfica pequeña mediante una red, generalmente con la misma tecnología

[3] Dispositivo que trasmite y recibe señales analógicas y digitales.

Se le llama así a las ondas aéreas electromagnéticas las que sirven para comunicar datos desde un punto a otro. Son portadoras de radio porque desempeñan la función de entregar energía al receptor. Los datos que se trasmiten son sobrepuestos sobre la portadora de radio para que pueda extraer de manera precisa por el receptor. Es a lo que se conoce como la modulación de la portadora por la información que se transmite. Después de que los datos son sobrepuestos (modulados) en el transportador de radio, la señal de radio ocupa más de una sola frecuencia, donde la frecuencia de la información modulada se agrega a la portadora. Múltiples portadoras de radio pueden coexistir en el mismo espacio a la vez, sin que haya interferencia, así las ondas de radio se transmiten sobre radiofrecuencias diferentes [Fort, 11].

Figura II, 1 Transmisor y Receptor de Radio
Fuente: [Fort, 11]

2.1.1.3.1 MODULACIÓN DE FRECUENCIA

La modulación en frecuencia (FM) es el proceso de combinar una señal de AF (Audio Frecuencia) con otra de RF (Radio Frecuencia) en el rango de frecuencias entre 88Mhz y 108MHz, tal que la amplitud de la AF varíe la frecuencia de la RF. [Aguirre, 08]

Si la señal de modulación varía en frecuencia, no tiene efecto en las excursiones máxima y mínima de la frecuencia de portadora, sino que solo determinada la rapidez o lentitud con que ocurren las variaciones en frecuencia. Es decir, que

una frecuencia más baja de la modulación hace que ocurran a una tasa más rápida. Sin embargo, las variaciones en amplitud de la señal de modulación si afectan las excursiones máxima y mínima de la frecuencia portadora. Una señal de mayor amplitud provoca un mayor cambio en la frecuencia y una señal más pequeña provoca un cambio menor en la frecuencia. [Aguirre, 08]

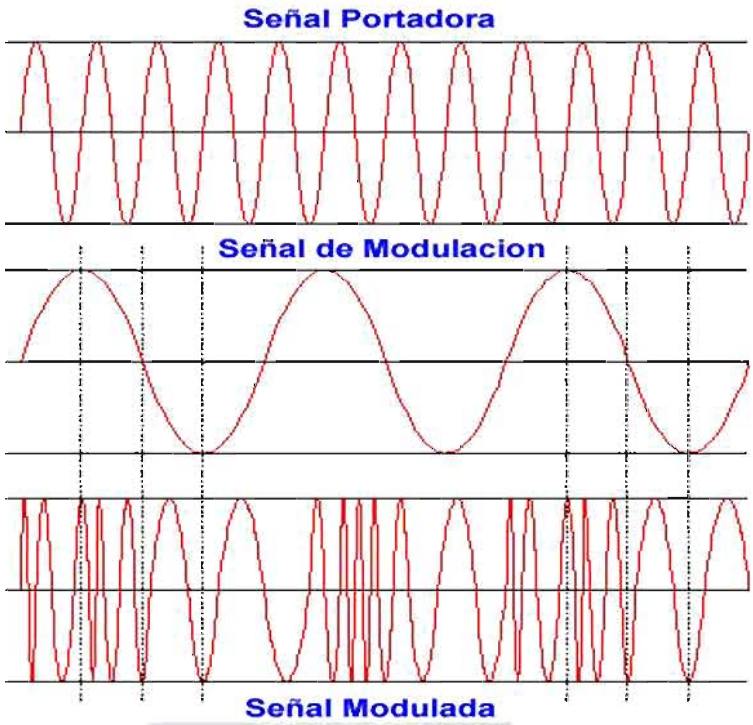


Figura II, 2 Ejemplos de Señal Modulada
Fuente: [Natur, 11]

2.1.1.3.1.1 MODULACIÓN DE FRECUENCIAS MULTI-PORTADORAS (MC FM)

Es un instrumento en el que un oscilador modula simultáneamente dos o más osciladores portadores. Las amplitudes de las portadoras son independientes, y cuando las amplitudes de las portadoras 2 y 3 son una fracción de la portadora 1, el instrumento genera picos alrededor de las frecuencias de las portadoras 2 y 3 así,

un instrumento con frecuencias multi-portadoras suena más realista, ya que cada portador produce frecuencias en diferentes partes del espectro. Por ejemplo, con modulación de frecuencias multi-portadoras se puede imitar el sonido de trompetas o de vocales [Roads, 11].

2.1.1.3.1.2 MODULACIÓN DE FRECUENCIAS MULTI-MODULADORAS (MM FM)

En la MM FM más de un oscilador modula a un oscilador portador y son posibles dos configuraciones: Paralela y en serie.

La MM FM paralela consiste en que dos ondas sinusoidales modulan simultáneamente a una onda sinusoidal portadora, mientras que la MM FM en serie consiste en que la onda sinusoidal moduladora M1 es modulada por otra onda moduladora M2. Mientras la frecuencia portadora aumenta, el índice de modulación disminuye [Wiki, 11].

2.1.1.3.1.3 MODULACIÓN DE FRECUENCIAS POR RETROALIMENTACIÓN (FEEDBACK)

En la FM por retroalimentación se soluciona un problema de la FM sencilla, el sonido electrónico. Con la FM por retroalimentación se logran sonidos más naturales. Existe FM por retroalimentación con uno, dos o tres osciladores. [Dodge, 11]

En la FM por Retroalimentación con un oscilador, el oscilador alimenta su output al input de la frecuencia a través de un multiplicador y un sumador. Lo que hace la retroalimentación es prolongar el decrecimiento de la amplitud mientras que la frecuencia aumenta. En la FM por Retroalimentación con dos osciladores se toma el output del oscilador por retroalimentación y se utiliza para modular a otro oscilador. En este tipo de retroalimentación se prolonga aún más el decrecimiento

de la amplitud mientras que la frecuencia aumenta generando un sonido más estridente [Dodge, 11].

Finalmente, en la FM por Retroalimentación con tres osciladores o retroalimentación indirecta, lo que se hace es generar un efecto de coro, logrando un amplio espectro, y cuando la retroalimentación es aumentada la energía tiende a centrarse en el punto alto del espectro [Dodge, 11].

2.1.2 BLUETOOTH

Bluetooth es una tecnología de la comunicación inalámbrica de corto alcance y bajo consumo de potencia, que permite conectividad inalámbrica entre dispositivos remotos. Se diseñó pensando básicamente en tres objetivos: tamaño, mínimo consumo y bajo precio. [Aguirre, 08]

Estos dispositivos pueden ser computadoras de escritorio, PDAs[4], teléfonos móviles y en fin las posibilidades pueden considerarse infinitas. [Aguirre, 08]

Bluetooth opera en la banda libre de radio ISM1[5] a 2.4 GHz Su máxima velocidad de transmisión de datos es de 1 Mbps El rango de alcance bluetooth depende de la potencia empleada en la transmisión. La mayor parte de los dispositivos que usan bluetooth transmiten con una potencia nominal de salida de 0 dBm, lo que permite un alcance de unos 10 metros en una ambiente libre de obstáculos. [Aguirre, 08].

Resumiendo, Bluetooth será una opción interesante para intercambio de datos entre teléfonos móviles, agendas, pasarelas residenciales, centralitas de seguridad/domótica, ordenadores, webcams, equipos de HiFi o reproductores MP3, mandos a distancia universales, etc.[Casadomo, 11].

[4] Del Inglés Personal Digital Assistant (Asistente Digital personal) , es un computador de mano originalmente diseñado como agenda personal

[5] Del Inglés (Industrial, Scientific and Medical) son bandas reservadas internacionalmente para uso no comercial de radiofrecuencia electromagnéticas en aéreas industrial, científica y médica. En la actualidad estas bandas han sido popularizadas por su uso en comunicaciones WLAN o WPAN.

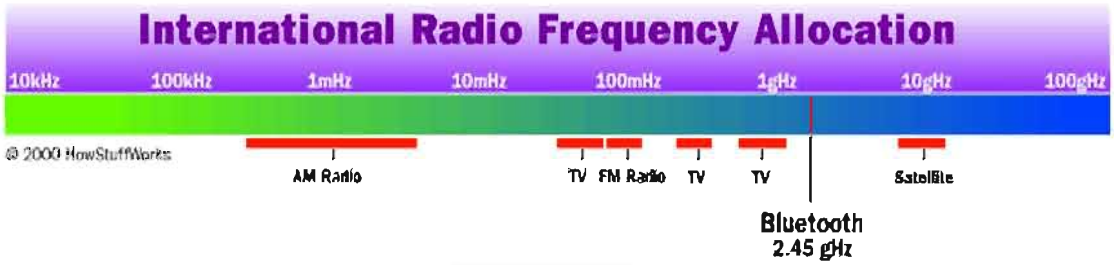


Figura II, 3 Relación de frecuencia de Bluetooth
Fuente: [Mono, 11]

Dos o más dispositivos Bluetooth que usan el mismo canal forman una picored.

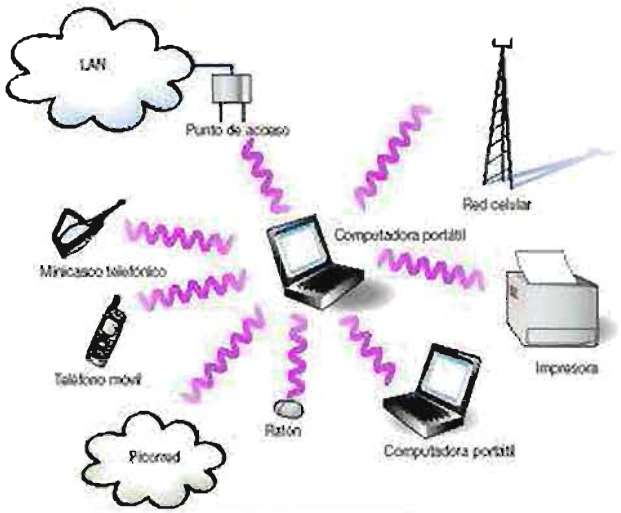


Figura II, 4 Picored
Fuente: [Electrónica, 11]

2.1.2.1 SALTO DE FRECUENCIA.

Debido a que la banda ISM está abierta a cualquiera, el sistema de radio bluetooth deberá estar preparado para evitar las múltiples interferencias que se pudieran producir. Estas pueden ser evitadas utilizando un sistema que busque una parte no utilizada del espectro o un sistema de salto de frecuencia. En los sistemas de radio Bluetooth se suele utilizar el método de salto de frecuencia debido a que está tecnología puede ser integrada en equipos de baja potencia y bajo coste. Este

sistema divide la banda de frecuencia en varios canales de salto, donde los transceptores, durante la conexión van cambiando de uno a otro canal de salto de manera pseudo-aleatoria. Con esto se consigue que el ancho de banda instantáneo sea muy pequeño y también una propagación efectiva sobre el total de ancho de banda. En conclusión, con el sistema FH (Salto de frecuencia), se pueden conseguir transceptores de banda estrecha con una gran inmunidad a las interferencias [Wiki, 11].

2.1.2.2 EL CANAL.

Bluetooth utiliza un sistema FH/TDD[6], en el que el canal queda dividido en intervalos de 625 µs, llamados slots, donde cada salto de frecuencia es ocupado por un slot. Dos o más unidades Bluetooth pueden compartir el mismo canal dentro de una piconet[7], donde una unidad actúa como maestra, controlando el tráfico de datos en la piconet que se genera entre las demás unidades, donde estas actúan como esclava, enviando y recibiendo señales hacia el maestro [Casadomo, 11].

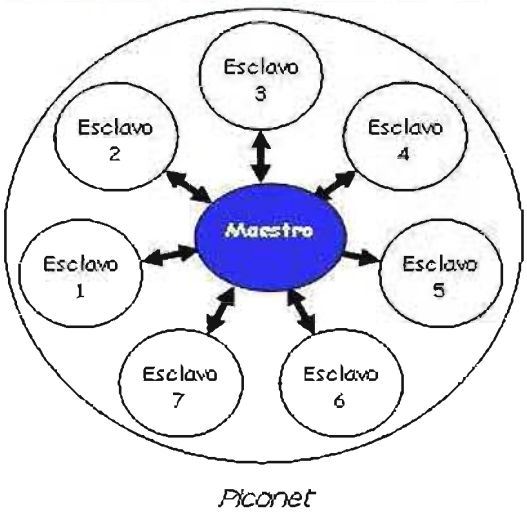


Figura II, 5 Diagrama Red Bluetooth
Fuente: [Hard, 11]

[6] Salto de frecuencia/ División de tiempo Dúplex
[7] Pequeña red que establecen automáticamente las terminales bluetooth para comunicarse entre sí.

El salto de frecuencia del canal está determinado por la secuencia de la señal, es decir, el orden en que llegan los saltos y por la fase de esta secuencia. En bluetooth, la secuencia queda fijada por la identidad de la unidad maestra de la piconet (un código único para cada equipo), y por su frecuencia de reloj [Casadomo, 11].

El maestro es el responsable de la sincronización entre los dispositivos de la piconet, su reloj y saltos de frecuencia controlan al resto de dispositivos. Además el maestro es quien, de manera predeterminada, lleva a cabo el procedimiento de búsqueda y establecimiento de la conexión. Los esclavos simplemente se sincronizan y siguen la secuencia de saltos determinada por el maestro [Electrónica, 11].

2.1.2.3 PICONETS

Debemos considerar si un equipo se encuentra dentro del radio de cobertura de otro, estos pueden establecer conexión entre ellos. Cada dispositivo tiene una dirección única de 48 bits, basada en el estándar IEEE 802.11[8] para WLAN. En principio solo son necesarias un par de unidades con las mismas características de hardware para establecer un enlace. Dos o más unidades Bluetooth que comparten un mismo canal forman una piconet.

Para Regular el tráfico en el canal, una de las unidades participantes se convertirá en maestra, pero por definición, la unidad que establece la piconet asume este papel y todos los demás serán esclavos. Los participantes podrían intercambiar los papeles si una unidad esclava quisiera asumir el papel de maestra. Sin embargo solo puede haber un maestro en la piconet al mismo tiempo. Hasta ocho usuarios o dispositivos pueden formar una piconet y hasta diez piconets pueden coexistir en una misma área de cobertura.

La topología Bluetooth permite la interconexión de varias piconets formando una scatternet. Aunque no existe sincronización entre piconets, un dispositivo puede

[8] Define el uso de niveles inferiores de la arquitectura OSI (Capa física y de enlace de datos), especificando sus normas de funcionamiento en una WLAN. Define la tecnología de redes de área local.

pertenecer a varias de ellas haciendo uso de la multiplexación por división del tiempo (TDD), aunque el dispositivo solo está activo en una piconet a la vez.

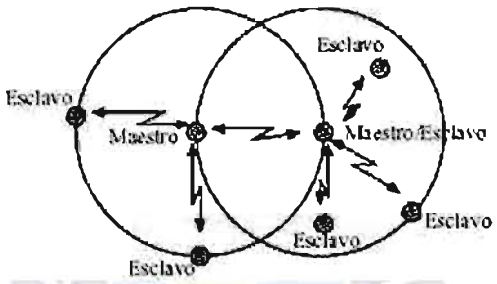


Figura II, 6 Scatternet
Fuente: [Electrónica, 11]

2.1.2.4. ALCANCE

La especificación del producto del protocolo bluetooth define 3 tipos de estados de entrega de energía para transmisores con una salida de 1mW, 2.5mW y 100mW.

La salida de energía define el alcance en metros para este dispositivo, es por ello que se debe considerar el uso que le queremos dar al dispositivo antes de adquirirlo.

2.1.2.4.1 MÁXIMA SALIDA DE PODER

1. Clase1 a 100mW y 100 Metros aprox.
2. Clase2 a 2.5mW y 10 Metros aprox.
3. Clase3 a 1mW y 1 Metros aprox

2.1.2.5 DISPOSITIVOS BASADOS EN BLUETOOTH

Ya están en el mercado muchos dispositivos que usan la tecnología Bluetooth, como son Impresoras, teclados, mouse, scanner, PDA, equipos celulares GPS[9], etc.

2.1.2.6 ADAPTADOR BLUETOOTH PARA PC O LAPTOPS

El dispositivo que va conectado al equipo central o PC es un adaptador USB a Bluetooth, es de solo 3cm de tamaño y nos permite un alcance de 10 metros aprox. Ya que posee un transmisor de 2.5mW de potencia.

Figura II, 7 Adaptador Bluetooth para PC o Laptops
Fuente: [Merc, 11]

2.1.2.7 CONVERSOR BLUETOOTH A SERIAL

Es un modulo Bluetooth que se utiliza para convertir a una señal con 9600bps, con 1 bit de inicio, 8 bits de datos y 1 bits de parada.

9 Del inglés (Global Positioning System) Sistema de Posicionamiento Global, es un sistema de posicionamiento terrestre, la posición la calculan los receptores GPS gracias a la información recibida desde satélites en órbita alrededor de la Tierra.

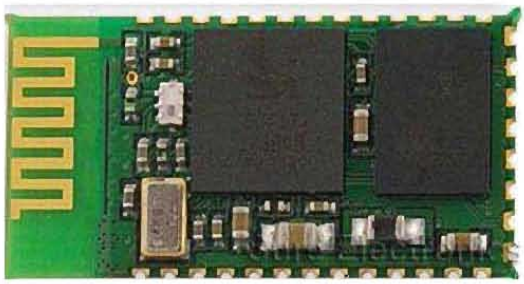


Figura II, 8 Modulo Conversor Bluetooth a Serial
Fuente: [Micro, 11]

El modulo bluetooth puede comunicarse con cualquier unidad máster como laptop; PDA, Computadoras personales y celulares.

Para comunicarse con los demás dispositivos el modulo se detecta como un COM virtual.

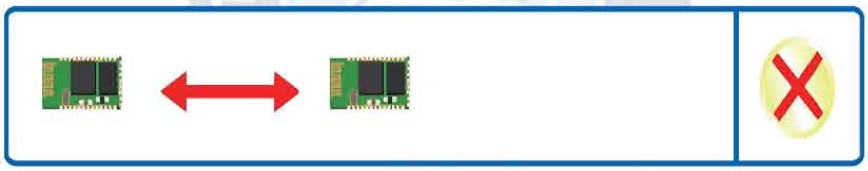


Figura II, 9 Comunicación Modulo Conversor Bluetooth Serial
Fuente: [Micro, 11]

Dos módulos Bluetooth son siempre unidades eslavas por lo tanto no se pueden comunicar entre sí.

Figura II, 10 Comunicación de módulos Bluetooth
Fuente: [Micro, 11]

2.2. MICROCONTROLADOR PIC

Los microcontroladores son circuitos integrados que poseen características de un computador completo; su tamaño en la unidad central de procesamiento, la cantidad de memoria y los periféricos incluidos dependen de la aplicación. Los microcontroladores son computadores digitales integrados en un chip que cuentan con un microprocesador o unidad de procesamiento central (CPU); dentro de su interior incluye tres unidades funcionales de una computadora: unidad central de procesamiento, memoria y unidades de E/S (Entrada / Salida).

A diferencia de los microcontroladores de propósito general, como los que se usan en las computadoras PC, los microcontroladores son unidades autosuficientes y más económicas; debido a que es más fácil convertirla en una computadora en funcionamiento con un mínimo de chips externos de apoyo. La idea es que el chip se coloque en el dispositivo, enganchando a la fuente de energía y de información que necesite. Un microcontrolador tradicional no le permitirá hacer esto, ya que espera todas las tareas sean manejadas por otros chips. Hay que agregarle los módulos de entrada y salida (puertos) y la memoria para almacenamiento de información. [Angulo, 2004]

Este puede escribirse en distintos lenguajes de programación; además, la mayoría de los microcontroladores actuales pueden reprogramarse repetidas veces, en el caso del PIC 16F628A, se puede reagravar 100 veces.

Por las características mencionadas y su alta flexibilidad, los microcontroladores son ampliamente utilizados como cerebro de una gran variedad de sistemas interesados en esta tarea que controlan máquinas, componentes de sistemas complejos, como aplicaciones de automatización y robótica, domótica, equipos médicos e incluso dispositivos de la vida diaria como automóviles. Un ejemplo de ellos es el famoso PIC16F628A de la familia PIC 16F628X.

Frecuentemente se emplea la notación µC o las siglas MCU (por microcontroller unit para referirse a los microcontroladores).

2.2.1. MICROCONTROLADOR PIC16F628A.

El PIC16f628A tiene 18 pines y es uno de los modelos estrella de Microchip, siendo además el sucesor del anterior modelo más importante (y todavía vigente) 16F84.

Figura II.11 Microcontrolador PIC16F628A
Fuente: [Robótica, 11]

Siendo un micro de la gama media tiene varias funcionalidades incorporadas. Es comercializado en 3 versiones que soportan velocidades de reloj diferentes, 4 MHz[10], 10 MHz y 20 MHz

Los PIC26F62X son chips de 18 pines, basados en memoria FLASH, miembros de la versátil familia de chips de alta performance, bajo costo PIC16CXX que tienen entre sus características relevantes utilizar tecnología CMOS[11], ser microcontroladores de 8 bits, soportar interrupciones externas e internas y ser reprogramables.

Estos microcontroladores tienen características especiales que permiten la reducción de componentes externos, y por lo tanto la reducción de costos, reforzando la confiabilidad y reduciendo el consumo eléctrico.

A continuación exponemos las características más significativas:

---

[10] Se utiliza muy frecuentemente como unidad de medida de la frecuencia de trabajo de un dispositivo de Hardware, o bien como medida de ondas electromagnéticas en telecomunicaciones.

[11] Del inglés Complementary Metal Oxide Semiconductor. Es una de las familias lógicas empleadas en la fabricación de circuitos integrados.

- Conjunto reducido de instrucciones (RISC). Solamente 35 instrucciones que aprender a utilizar.
- Oscilador interno de 4MHz
- Las instrucciones se ejecutan en un solo ciclo de máquina excepto los saltos (goto y call), que requieren 2 ciclos. Aquí hay que especificar que un ciclo de máquina se lleva 4 ciclos de reloj, sise utiliza el reloj interno de 4MHz, los ciclos de máquina se realizarán con una frecuencia de 1MHz, es decir que cada instrucción se ejecutará en 1uS (microsegundo)
- Opera con una frecuencia de reloj de hasta 20MHz (ciclo de máquina de 200 ns)
- Memoria EEPROM: 128 bytes (8 bits por registro)
- Stack de 8 niveles
- 16 terminales de I/O que soportan corrientes de hasta 25 mA
- 3 Temporizadores
- Módulos de comunicación serie, comparadores, PWM

2.2.1.1. ASPECTO EXTERNO

El PIC16F628A está fabricado con tecnología CMOS de altas prestaciones y encapsulado en plástico con 18 patillas, con la nomenclatura que se muestra. La misión de cada patilla comentada brevemente es:

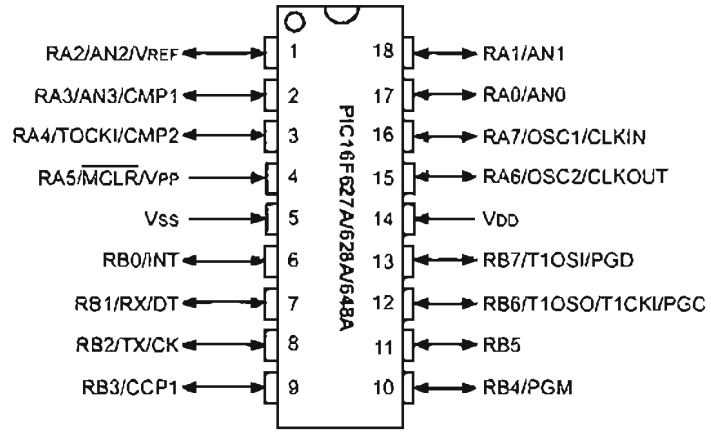


Figura II.12 Distribución de pines PIC16F628A
Fuente: [Robótica, 11]

Veamos la misión de cada patilla:

- VDD: Patilla por la que se aplica la tensión positiva de la alimentación.
- VSS: Patilla conectada a la tierra o negativo de la alimentación.
- OSC1/CLKIN: Patilla por la que se aplica la entrada del circuito oscilador externo que proporciona la frecuencia de trabajo del microcontrolador.
- OSC2/CLKOUT: Patilla auxiliar del circuito oscilador.
- MCLR#: Esta patilla es activa con nivel lógico bajo, lo que se representa con el símbolo #. En pocas palabras con esta reseteas el PIC.
- RA0-RA4: Son las líneas de E/S digitales correspondientes a la puerta A. La línea RA4.
- RB0-RB7: estas 8 patillas corresponden a las 8 líneas de E/S digitales de la puerta B. La línea RB0 multiplexa otra función, que es la de servir como entrada a otra petición externa de interrupción, por eso se le denomina RB0/INT.

2.2.1.2. ARQUITECTURA INTERNA

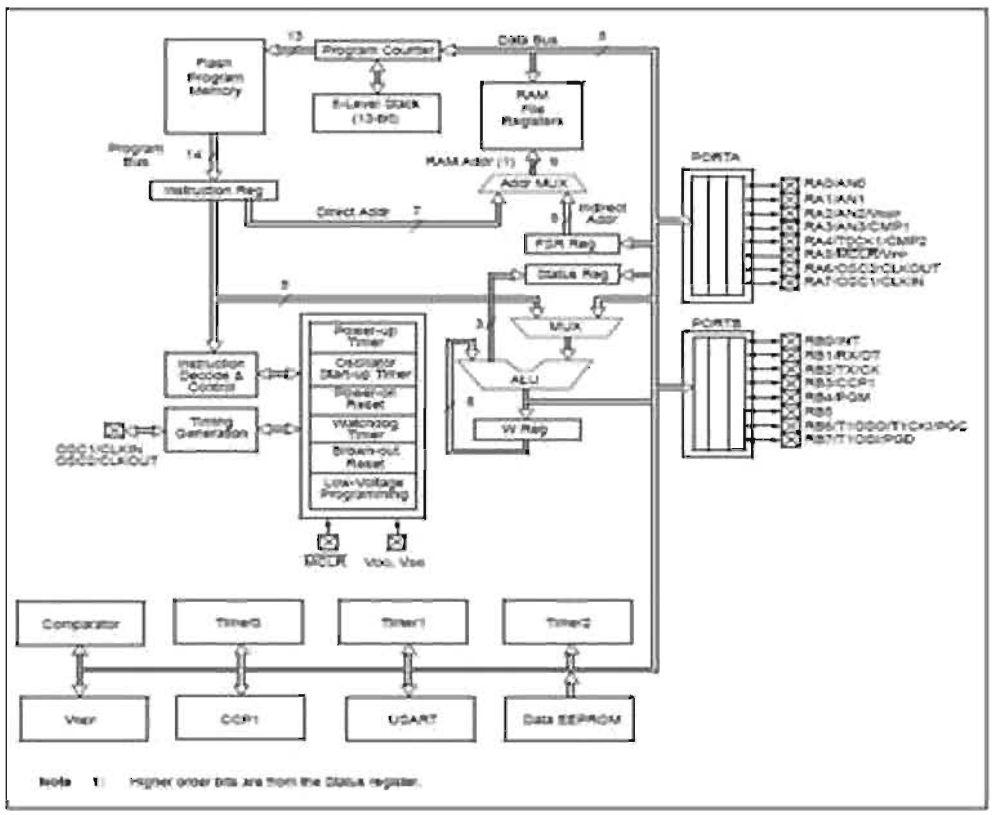


Figura II, 13 Arquitectura Interna PIC 16F628A
Fuente: [Robótica, 11]

2.2.1.3. ORGANIZACIÓN DE MEMORIA

El PIC16F628A posee un controlador de programa de 13 bits, capaz de direccionar un espacio de memoria de 8Kx14. Sin embargo, únicamente los primeros 2Kx14, desde 0000h hasta 07FFh están implementados, figura 2. Los vectores de reset e interrupción están en las direcciones 0000h y 0004H, respectivamente. La pila (stack) es de 8 niveles, lo cual significa que puede soportar hasta 8 direcciones de retorno de subrutina. El PIC16f627 y el PIC 16F84 tienen la misma organización,

excepto que únicamente están implementados los primeros 1Kx14, desde 0000H hasta 03FFh [Pict, 11].

El PIC16F628 y el PIC16F627 poseen un espacio de memoria RAM de datos de 512x8, dividido en 4 bancos de 128 bytes cada uno, figura3. Sin embargo, solo están implementados 330 bytes, correspondiendo 224 al área de los registros de propósito general (GPR) y 36 al área de los registros de función especial (SFR)

Los restantes 70 bytes implementados son espejos de algunos SFR de uso frecuente, así como de los últimos 16 GPR del banco 0. Por ejemplo, las posiciones 0Bh, 8Bh, 10Bh, 18Bh corresponden al registro INTCON, de modo que una operación hecha en cualquiera de ellos, se refleja automáticamente en los otros. Se dice, entonces, que las posiciones 8Bh, 10Bh y 18Bh están mapeadas en la posición 0Bh. Esta característica agiliza el acceso a estos registros, puesto que no siempre es necesario especificar el banco donde se encuentran. La selección del banco de ubicación de un SFR o un GPR particular se hace mediante los bits 6 (RP1) y 5(RP0) del registro STATUS [Pict, 11].

2.2.2. LA GESTIÓN DE LOS PUERTOS

Los microcontroladores PIC tienen terminales de entrada / salida divididos en puertos, que se encuentran nombrados alfabéticamente A,B,C,D,... Cada puerto puede tener hasta 8 terminales que, de forma básica se comportan como una entrada y salida digital. Según las características del PIC, cada puerto puede tener, además asignado un bloque funcional.

La gestión de bus de datos se realiza a través de los registros PORTx (PORTA:05h,PORTB:06h,PORTC:07h,PORTD:08h o PORTE:09h en el BANCO 0 de la memoria RAM).

2.3. COMPILADOR PIC Y SIMULADOR PROTEUS PARA MICROCONTROLADORES PIC

Este proceso corresponde en utilizar un programa en el PC que toma el código ensamblado(.hex, .o, .bin, .coff) para el µC específico, y lo envía mediante algún puerto serial , párelo o USB, a un dispositivo que lo describe en la memoria del µC.

Se acostumbra a denominar programador tanto al software como al hardware que está involucrado para este propósito, lo cual puede prestarse a confusión. El software programador a veces recibe también el nombre de *downloader*, ya que su propósito es descargar o transferir el código ensamblado desde el PC al µC. Es importante mencionar que no deben confundirse los términos desarrollo o programación del software y programación del µC, el primero se refiere a describir el programa, mientras que el segundo se refiere a transferir el código de la maquina a memoria del µC. Los códigos de ensamblador generados por los compiladores son variados, que de igual manera tendrán un ensamblador para generar el código binario de máquina, los más conocidos compiladores es el MPlab, Microcode Plus Studio Basic, PIC C, Niple.

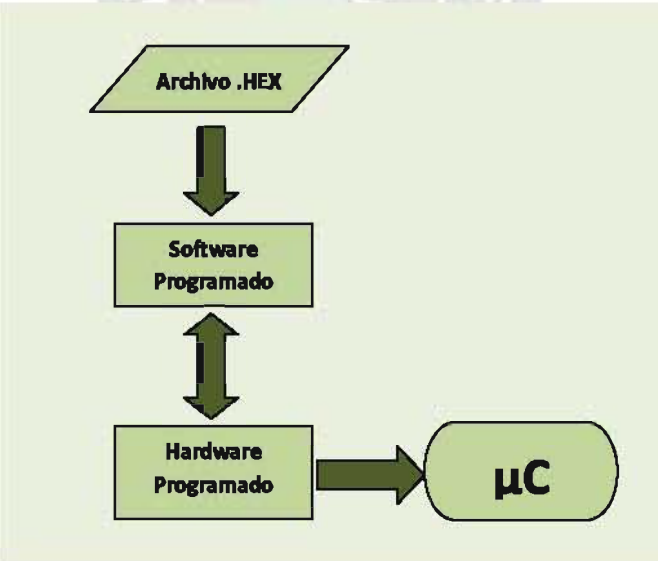


Figura II, 14 Proceso de Grabación al Microcontrolador
Fuente: Elaboración Propia

Una vez programado el µC se puede instalar en el circuito final para comprobar su adecuado funcionamiento. Existen herramientas de software que permiten simular el comportamiento de un µC, muy útiles cuando el programa alcanza cierta complejidad. Para resolver problemas en un circuito real, el instrumento más utilizado es el analizador lógico o el más conocido es el simulador PROTEUS. [Horta, 2006]

2.3.1 MICROCODE STUDIO PLUS

Microcode Studio plus es un software gratuito, el cual está diseñado para que se pueda ejecutar en los sistemas operativos Windows de Microsoft, para la realización de esta investigación se usa como sistema operativo Windows XP, el microcode Estudio plus es u editor de texto como el bloc de notas de Windows, el programa está hecho principalmente para la programación de los microcontroladores, este software puede trabajar con muchos tipos de microcontroladores como PIC16F628A,16F877A, entre otros.

Este software al momento de realizar un programa nos da varios archivos con los cuales vamos a utilizar el archivo con extensión HEX, el mismo que ayudará para poderle abrir y así poder grabar en el PIC.

El Microcode Studio ayuda de una manera sencilla el crear programas y con la ayuda del compilador se puede identificar si existe algún error en el código, cabe mencionar que este programa trabajar en lenguaje Basic.

Como trabaja con lenguaje Basic el Microcode Studio también soporta sentencias en Assembler, si es necesario al momento de programar tuviéramos que poner sentencias en assembler si lo puede recibir pero llamando por medio de subrutinas en el programa que está realizando.

2.3.2 SIMULADOR DE CIRCUITOS ISIS DE PROTEUS

El entorno de diseño electrónico PROTEUS de LABCENTER ELECTRONICS, ofrece la posibilidad de simular código microcontrolador de alto, medio y bajo nivel y

simultáneamente con la simulación en modo mixto de SPICE. Esto permite el diseño tanto a nivel hardware y realizar las simulaciones en el mismo y único entorno. Para ello se suministran tres potentes sub – Entornos como son el ISIS para el grafico, VSM (Virtual System Modelling) para la simulación. [García, 2008]

Posee una muy buena colección de librerías de modelos tanto para dibujar, como simulando o para las placas. Además permite la creación de nuevos componentes, su modelización para su simulación e incluso la posibilidad de solicitar al fabricante (LABCENTER ELECTRONICS), que cree un nuevo modelo.

2.3.3. GRABACION DE MICROCONTROLADORES PIC

Un microcontrolador es un circuito integrado programable que contiene todos los componentes necesarios para controlar el funcionamiento de una tarea determinada. El microcontrolador dispone de una memoria de programa interna donde se almacena el programa que lo controle y lo consista realmente en una serie de números hexadecimales. [Palacios, 2004]

El programa de control se graba en la memoria mediante un equipo físico denominado grabador o programador. El grabador se conecta a un ordenador normalmente a través de un puerto (paralelo, serie, USB) mediante el cable de conexión adecuado. En el ordenador se ejecuta un software que controla la grabación de la memoria de programa del microcontrolador. Este proceso se denomina grabar o programar el microcontrolador.

2.3.4. SOFTWARE DE GRABACION

El software de grabación de microcontroladores PIC, permite programación de muchos dispositivos. Los software vienen en un CD o un dispositivo de almacenamiento, también se puede descargar mediante vía web.

La instalación de este software es muy sencilla basta seguir el procedimiento usual en Windows. Este archivo consta del fichero. exe, que contiene todo el código

necesario para su funcionamiento, con versiones para cualquier sistema operativo Windows.

2.4. PROGRAMAS

2.4.1. VISUAL BASIC

Visual Basic es una herramienta rápida de desarrollo (RAD) de Microsoft utilizada para crear aplicaciones, es decir, utilizando el CLR, la librería de clases, ADO, NET, ASP.NET, etc.

Brinda las herramientas necesarias para crear, distribuir, administrar y dar mantenimiento a las aplicaciones con una gran facilidad, rapidez y bajo costo. Permite la integración y el uso cruzado de los lenguajes de programación.

2.4.2 JAVA 2 MICRO EDITION: SOPORTE BLUETOOTH

Bluetooth es, una tecnología ideal para la conexión de dispositivos de bajas prestaciones (móviles, cámaras de fotos, auriculares manos libres, impresoras....). Uno de los mayores ámbitos de utilización de Bluetooth es sin duda los teléfonos móviles. Cada vez es más común encontrar terminales móviles con soporte Java y Bluetooth y simplemente es un paso natural que surja la necesidad de programar estos dispositivos a través de Java. Desde el JCP se ha desarrollado un JSR que cubre esta necesidad [Mondaca, 07].

Actualmente mobile processing es un contexto para la exploración del emergente espacio conceptual habilitado por los medios electrónicos. El principal objetivo fue crear de manera rápida sketches de aplicaciones, simplificando las tareas que se deben realizar para lograr un prototipo funcional. Mobile Processing es que podemos escribir código de una manera más simple en un entorno más sencillo e intuitivo que los habituales en Java.

2.4.2.1. APIS JAVA PARA BLUETOOTH

Mientras que el hardware Bluetooth había avanzado mucho, hasta hace relativamente poco no había manera de desarrollar aplicaciones java Bluetooth hasta que apareció JSR 82 que estandarizó la forma de desarrollar aplicaciones Bluetooth usando java. Esta esconde la complejidad del protocolo Bluetooth detrás de unos APIs que permiten centrarse en el desarrollo en vez de los detalles de bajo nivel del Bluetooth [Mondaca, 07].

2.5. CONCEPTO DE AUTÓMATA

La palabra autómata significa *robot* y deriva del checo robots que significa trabajador, pero no es eso exactamente lo que se entiende hoy en día como el significado de esta palabra. Actualmente se define como robots a un manipulador multifuncional, reprogramable, diseñado para mover materiales, piezas u otros dispositivos especializados, a través de distintos movimientos para el desempeño de una variedad de tareas. [Todorobot, 2009]

Las tres leyes de la robótica fueron denominadas por Asimov, y son:

- Un autómata o robot no puede actuar contra un ser humano, mediante la inacción, que un ser humano sufra daños.
- Un autómata o robot deben obedecer las órdenes dadas por los seres humanos, salvo que estén en conflictos con la primera ley.
- Un autómata o robot debe proteger su propia existencia, a no ser que esté en conflicto con las dos primeras leyes.

2.5.1. AUTÓMATA PROGRAMABLE

Un autómata es un sistema secuencial, aunque en ocasiones la palabra es utilizada también para referirse a un robot. Puede definirse como un equipo electrónico programable en lenguaje no informático y diseñado para controlar, en tiempo real y

en ambiente industrial, procesos secuenciales. Sin embargo, la rápida evolución de los autómatas hace que esta definición no esté cerrada [Wiki, 11].

2.6. NORMA ETSI

Las normas ETSI son estándares desarrollados por el Instituto de Estándares de Telecomunicaciones (ETSI), una organización independiente y sin ánimo de lucro reconocida oficialmente por la Comisión Europea como organismo de estandarización de las Tecnologías de la información y las comunicaciones (TIC) en Europa.

Posiblemente, la norma ETSI más relevante ha sido el estándar de telefonía móvil digital GSM, extendido por gran parte del mundo. Ente los desarrollos ms importantes en los que participa actualmente, el TISPAN es la organización dependiente de ETSI encargada de normalizar las redes fijas y la convergencia de Internet, y ETSI también en el 3GPP, el principal organismo de estandarización de la telefonía móvil 3G.

2.7. PERFIL DE PUERTO SERIE (SPP)

Es un protocolo fiable basado en comunicaciones con conexión que se encarga de controlar la integridad de los paquetes y confirmar los paquetes recibidos a través de una red [WIKI, 11]

2.8. PROTOCOLO RFCOMM

El protocolo RFCOMM provee múltiples emulaciones de los puertos serie RS – 232 entre dos dispositivos Bluetooth. Las direcciones Bluetooth de los dos puntos terminales definen una sesión RFCOMM. Una sesión puede tener de una conexión, el número de conexiones dependerá de la implementación. Un dispositivo podrá tener más de una sesión RFCOMM en tanto que esté conectado a más de un dispositivo. [Blue, 11].

CAPITULO III

Resumen
En el presente capítulo se realiza el desarrollo de la tesis utilizando los conceptos y los valores mencionados en el anterior capítulo, combinando la teoría con la práctica.

El método que se utiliza en el presente trabajo es el método automático, que comienza desde el análisis hasta el desarrollo, el cual consiste en la implementación de programas que son: J2ME, Visual Basic, Microcode Studio Plus y el diseño del prototipo del autómata incluyéndole el microcontrolador PIC.

3.1. DESCRIPCION INFORMAL DEL MODELO DE SISTEMA

La temática de la tesis aborda la conectividad entre dos dispositivos y la forma de interactuar entre ellos. La idea es demostrar que no importa qué tipo de dispositivo se use, con tal de que se posea interfaces que nos permitan un flujo de datos, un método de programación y un equipo que actué como centro de control de datos. Esto hace posible que podamos enviar datos y así sacarle provecho a esta mezcla de tecnologías. En este caso se trabajo con microcontroladores PIC, transmisores y receptores inalámbricos de 433Mhz. Un PC, con dispositivos Bluetooth y equipos celulares.

La arquitectura general del sistema se ilustra en la figura III, 1; se observan los diferentes elementos que conforman la comunicación entre los dispositivos. A continuación se describen cada uno de ellos en el siguiente diagrama de bloques.

Figura III, 1 Esquema Propuesto – Comunicación entre dispositivos
Fuente: [Elaboración Propia]

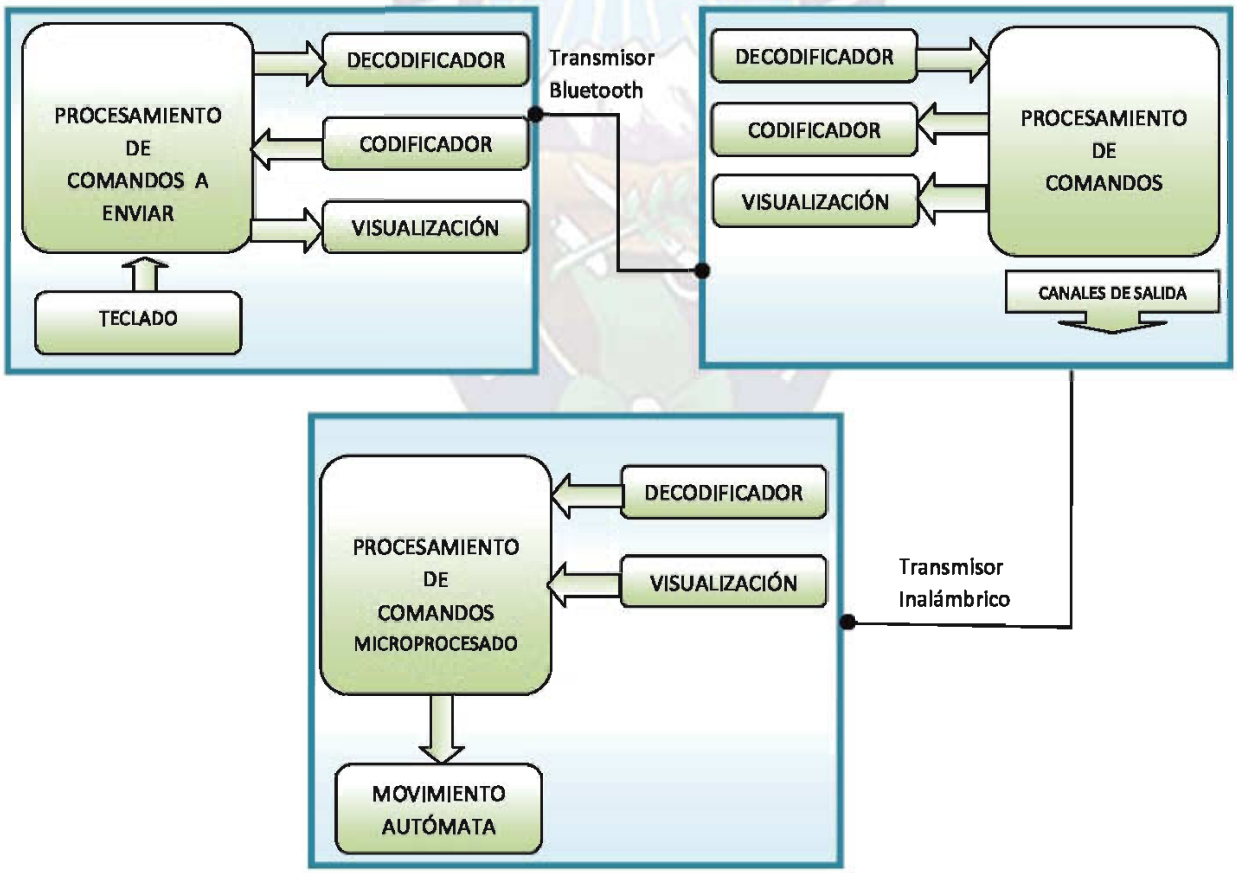


Figura III, 2 Diagrama de bloques – Esquema Propuesto
Fuente: [Elaboración Propia]

3.1.1. EQUIPO CENTRAL O MAESTRO

El equipo central de Control (PC) se conecta con un equipo de telefonía móvil mediante la tecnología inalámbrica Bluetooth. Como la mayoría de los PC que hay en el mercado no traen la tecnología Bluetooth incorporada como estándar, se usa modulo Bluetooth y un cable USB-RS232 para integrar esta habilidad al PC, gracias a esto podemos crear un canal estable de transmisión de datos entre el PC y el celular usando la tecnología inalámbrica bluetooth.

Para transmitir datos entre el PC y el teléfono móvil, se utilizó una programación con soporte para JavaMicroEdition, más conocido como J2ME que permite establecer un enlace de comunicación de datos entre celular y el PC emulando un puerto serial.

La Figura III, 2 muestra el proceso de funcionamiento del equipo Central o Maestro.

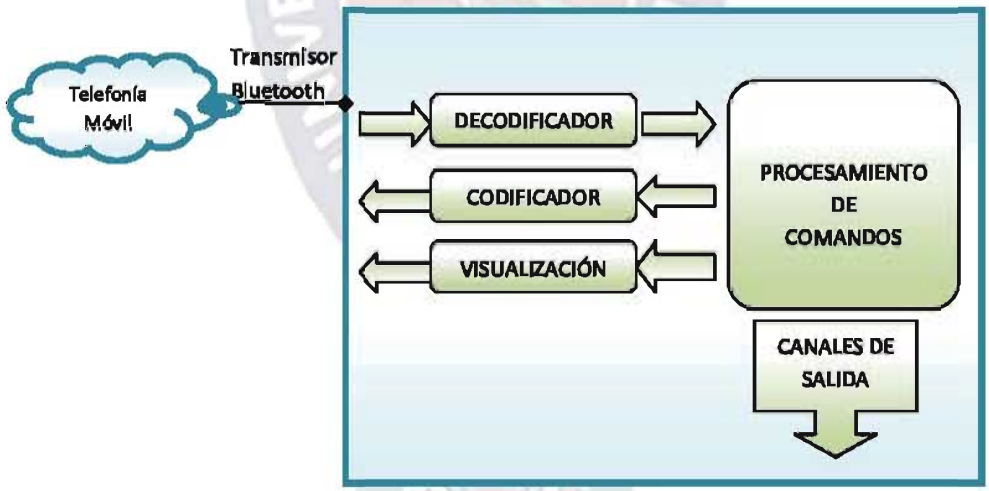


Figura III, 3 Diagrama de bloques del Equipo Central o Maestro
Fuente: [Elaboración Propia]

3.1.2. SISTEMA SOFTWARE – CELULAR

El dispositivo, teléfono móvil procesa comandos y los envía al equipo central o maestro mediante un transmisor Bluetooth cuyo diseño nos permite realizar transferencia de datos desde cualquier lugar donde se detecte la señal Bluetooth.

Este dispositivo se comunicará con la unidad central con la cual se podrá cambiar los estados de salida y hacer la lectura de los estados.

La Figura III, 3 muestra el proceso de funcionamiento del dispositivo teléfono móvil.

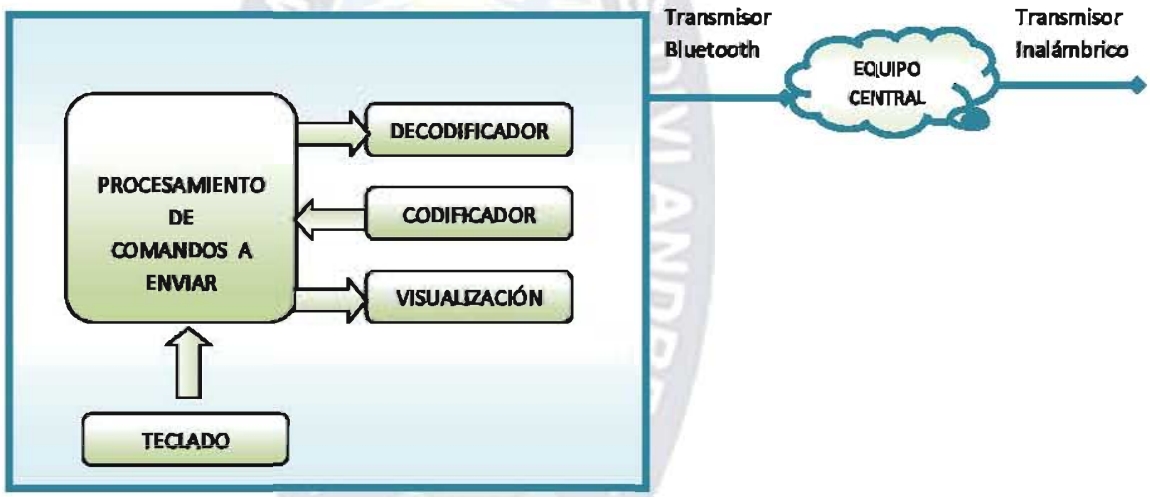


Figura III, 4 Diagrama de Bloques Sistema de Software Celular
Fuente: [Elaboración Propia]

3.1.3. MODULO AUTOMATA

Este dispositivo recibirá comandos remotos de la unidad central y ejecutará las acciones requeridas.

El modulo Autómata es la encargada de recepcionar las señales producidas por la unidad central o maestro que tiene almacenado las instrucciones recibidas de teléfono móvil, y por medio

El circuito más importante de este módulo es el microcontrolador PIC que es el encargado de recepcionar las señales binarias y poner en movimiento el autómata.

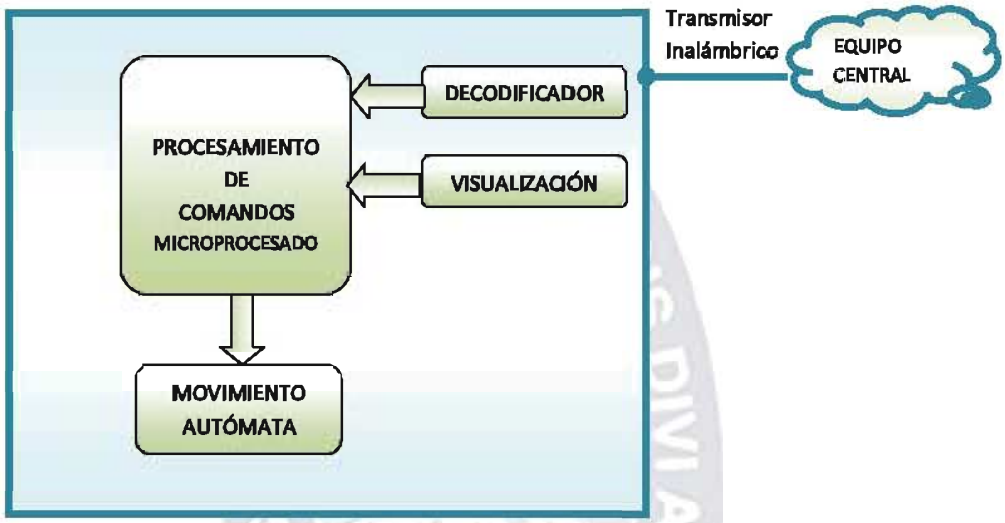


Figura III, 5 Diagrama de Bloques Módulo Autómata
Fuente: [Elaboración Propia]

3.2. DESCRIPCION FORMAL DEL MODELO DE SISTEMA

3.2.1. ADECUADOR DE SEÑAL

Esta etapa está construida en base de transmisores, compuestas lógicas y otros circuitos integrados los cuales confirman un circuito que adecua la señal basado fundamentalmente en el control de movimiento del autómata para que luego dicha señal llegue a los motores paso a paso.

- **Fuente de alimentación**

  La alimentación del sistema de control y potencia se suministra mediante fuentes aisladas inalámbricamente:

Figura III, 8 Sistema de Control
Fuente: [Elaboración Propia]

- **Sistema microcontrolador**

El diseño y análisis del sistema central de control, se basa fundamentalmente en el microcontrolador PIC16F628A que ofrece todas las posibilidades y versatilidad requeridas. El esquema general, se presenta en el diagrama de la figura III, 7 donde se detallan los puertos y líneas de control utilizadas en este dispositivo.

- **Sistema de potencia**

El sistema de potencia toma la información del microcontrolador y la convierte en señales con el nivel de potencia requerido por los motores.

- **Control de movimientos**

El sistema de control de movimientos de cada uno de los motores está implementado con el microcontrolador PIC16F628A. Las características particulares del sistema de los ejes, permiten el movimiento sincronizado básico del autómata.

- **Sistema de protección**

El sistema de protección, el cual evita el sobre voltaje, corto circuitos que pueden dañar el sistema.

3.2.2. GENERACION DE COMUNICACIÓN INALAMBRICA POR SOFTWARE

La especificación de la comunicación inalámbrica requiere de un error de frecuencia de menos del 1% una distorsión armónica total de menos 10%. Adicionalmente, la respuesta en frecuencia de una línea telefónica móvil, atenúa las frecuencias altas, porque la transmisión del grupo de frecuencias altas, requiere de mayor amplitud que el grupo bajas de frecuencias.

En este análisis se utiliza una tabla de seno con dos apuntadores, uno para el tono del celular y el otro del tono de recepción del microcontrolador. Cada apuntador tiene su propio valor de incremento, tal que se puede generar tonos de diferentes frecuencias.

A fin de reducir el contenido armónico al mínimo, la salida debe ser muestreada debe ser nula para filtrar el ruido introducido, por la frecuencia del muestreo. Un periodo de 128us, el cual es un periodo estándar de la industria de las telecomunicaciones.

3.2.3. CALCULANDO EL INCREMENTO PARA LOS APUNTADORES

La forma general de ecuación es:

$$Fsalida = incremento * Fs / Tamaño\ de\ tabla$$

Donde:

$$Fsalida = Frecuencia\ a\ Generar$$

$$Fs = Frecuencia\ de\ muestreo$$

Tamaño de Tabla: Numero de muestra en tabla seno para un ciclo completo

Incremento: valor que debe ser incrementado el apuntador para cada periodo de muestreo.

Dejando incremento tenemos:

$$Incremento = Fsalida * TamañoTabla / Fs$$

Por ejemplo para generar la frecuencia 697Hz, si tabla de seno tiene 256 entradas, el valor de incremento es:

$$Incremento = 697 * 256/7812 = 23$$

Como podemos usar un intervalo en fracción, lo redondeamos a 22. Introduciendo este valor en la ecuación, obtenemos como frecuencia generada:

$$Fsalida = 23 * 7812/256 = 701.86\ Hz$$

Lo cual da un error en frecuencia de 0,7% lo cual es aceptable.

La siguiente tabla, resume los cálculos realizados para una frecuencia de muestreo de 7812Hz y una tabla de 256 entradas.

Periodo de Muestreo	128us (7812.5HZ)			
Tamaño de Tabla	256			
Frecuencia	697	770	852	941
Incremento	23	25	28	31
Frecuencia actual	701.86	763	854	946
%error	0.7	-0.92	0.29	0.53
Frecuencia	1209	1336	1477	1633
Incremento	40	44	48	54
Frecuencia actual	1221	1343	1465	1648
%error	0.96	0.5	-0.83	0.91

Tabla III, 2 Calculo de Tonos Comunicación Inalámbrica
Fuente: [elaboración Propia]

3.2.4. IMPLEMENTANDO PRE – ÉNFASIS DE LOS TONOS ALTOS

A fin de compensar la atenuación de las frecuencias altas, lo cual es característica de la mayoría de las líneas telefónicas móviles, la amplitud de estos tonos deben estar de 1 a 3dB por encima de la amplitud de las frecuencias bajas, eso equivale a multiplicar la amplitud actual por valores de 1.12 a 1.41. Un valor aproximado es de 1.25 ya que 0.25 puede ser obtenido dividiendo la unidad entre 4 que a su vez equivale a realizar desplazamiento a la derecha de 2 bits.

Cuando una frecuencia del grupo alto es leída de la tabla de seno, esta es desplazada a la derecha de 2 bits (dividida por 4), entonces el mismo valor seno es agregado al resultado produciendo la multiplicación por 1.25. El pre –énfasis es:

$$dB = 20 \log (V2/V1) = 20 \log (1.25) = 1.94dB$$

3.2.5. CALCULANDO LOS VALORES DE SENO PARA EVITAR DESBORDAMIENTO

Los valores de la tabla seno deben ser calculados para evitar error por desbordamiento dos muestras están sumadas. Si por ejemplo se usan 16 bits para almacenar los valores de la tabla, el rango a presentar está entre – 32768 y + 32767.

Para evitar desbordamiento, los valores deben se escalados tal que el rango este entre - 16384 y + 16383. Ya que a los tonos se le aplica pre- énfasis, el valor debe ser algo menor que el máximo.

$$MaxSalida = Maxsin * (1+preénfasis)$$

Resolviendo para MaxSin:

$$MaxSin = MaxSalida* (1+preénfasis) = 32767/(1+1.259)= 14563.11$$

De tal manera que los valores en la tabla seno deben estar entre -14563 y +14563.

3.2.5.1. FILTRO PASA BAJOS

Es necesario agregar un filtro pasa bajos que opere como integrador a la salida del microcontrolador.

3.2.5.2. TEORÍA DE LA DECODIFICACIÓN

Existe una variedad de métodos disponibles para decodificar señales, los métodos más básicos involucran el uso de ocho filtros calibrados a cada una de las frecuencias inalámbricas.

Cuando el análisis del espectro de frecuencias se requiere solo para ciertas frecuencias, la transformada de Fourier provee una mejor solución comparada con otros algoritmos.

El método de la transformada discreta de Fourier requiere de un considerable poder computacional. Típicamente los microcontroladores de 8 bits no tienen un

considerable poder computacional, así que la implementación de esta clase de algoritmo ocuparía toda su capacidad.

Uno de los principales objetivos es hallar una alternativa para lograr la conversión al dominio de las frecuencias de las señales inalámbricas sin sacrificar muchos recursos del microcontrolador.

Ante de esto es importante entender lo básico de los métodos tradicionales:

3.2.5.3. ESTUDIO BÁSICO TRANSFORMADA DE FOURIER

Mostro que cualquier señal puede ser construida sumando series de ondas con la apropiada amplitud y fase, el teorema de Fourier asume la sumatoria de ondas de infinita duración.

El espectro de frecuencias tiene dos ejes: frecuencias y energía. La energía es una frecuencia particular está representada por una barra en ese punto. La transformada de Fourier indica detalles de amplitud, fase y frecuencia de señales puras requeridas para crear una señal particular. Debido a que cada señal pura representa una frecuencia en el espectro y su amplitud de detalles de energía, el método de la transformada de Fourier Ecuación 3.1 provee el espectro de la señal.

$$X^{(W)} = \frac{1}{2\pi}\int_{-\infty}^{\infty} x(t) * X^{-wt}\, dt \qquad \textbf{Ecuación 3.1}$$

Donde en el dominio del tiempo se tiene:

x(t) : es complejo, continuo y no periódico

t: va de $-\infty$ a $+\infty$

y en el dominio de la frecuencia:

X(t) : es complejo, continuo y no periódico

w: va de $-\infty$ a $+\infty$

La transformada de Fourier es una ecuación matemática que usa integrales, mientras que la transformada discreta de Fourier que se indica en la ecuación 3.2 equivale usando sumatorias.

$$X^{(k)} = \frac{1}{N} = \sum_{n=0}^{N-1} x[n] * \gamma^{-j2\pi k\frac{n}{N}}$$

Ecuación 3.2

Donde el dominio del tiempo se tiene:

x[n]:	es complejo, discreto y periódico
n	va en un periodo de 0 a N-1

Y en el dominio de la frecuencia:

X(k):	es complejo, discreto y periódico
k	va en un periodo de 0 a N-1

Para entender mejor cómo funciona la transformada discreta de Fourier, esta ecuación puede ser modificada usando la ecuación de Euler y finalmente esta queda como se indica en la ecuación 3.3.

$$X^{(k)} = \frac{1}{N}\sum_{n=0}^{N-1} x[n] * \left(\cos(2\pi k\frac{n}{N}) - j\sin(2\pi k\frac{n}{N})\right)$$

Ecuación 3.3

Con la ecuación simplificada, se nota que si N veces las muestras periódicas se multiplican por ondas, da un número complejo como resultado. La magnitud de este número nos da la energía en el análisis de frecuencias. Por tanto, si se aplica la transformada para una frecuencia fija de análisis, se tendrá el valor de energía de la señal en ese punto de frecuencia.

Los principios hasta aquí revisados se emplearán eventualmente en el desarrollo de esta investigación de tesis.

3.2.6. PRESENTACION DEL MODELO DE CONTROL

3.2.6. PROGRAMA PARA EL MICROCONTRADOR

El programa se realizo en el lenguaje de compilador Microcode Studio Plus, esto es por la razón de que es uno de los lenguajes que acepta el microcontrolador.

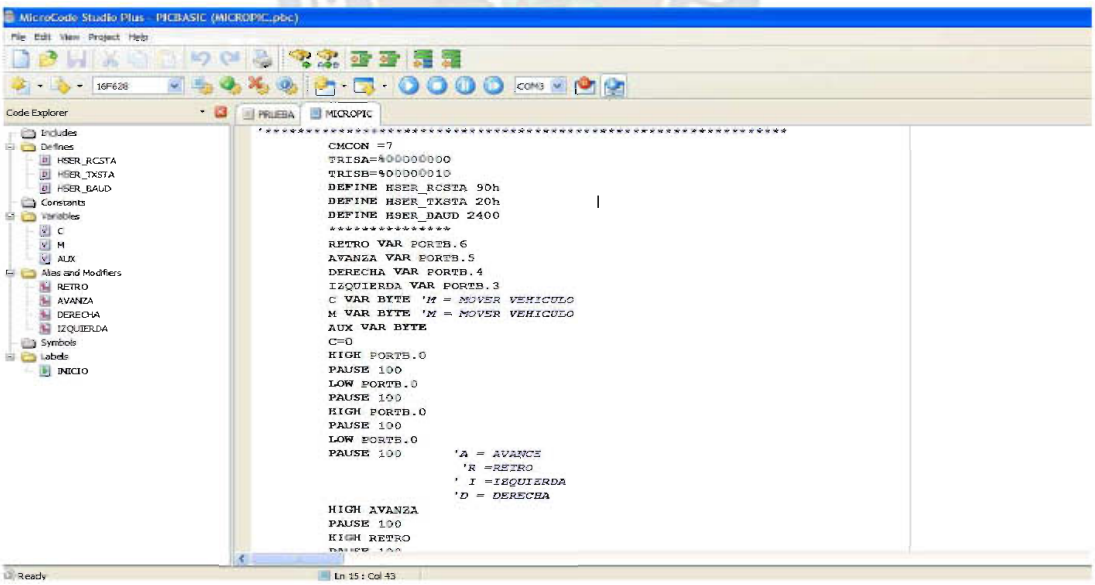


Figura III, 8 Pantalla capturada de microcode studio
Fuente: [Elaboración Propia]

3.2.6.1. PROGRAMA COMPILADOR MICROCODE STUDIO PLUS

Para esta sección del programa se utilizó la función HSEROUT que permite la comunicación serial a través de:

- DEFINA HSER_TXSTA 20h Determinación de la velocidad de transmisión.
- DEFINA HSER_BAUD 2400 Determina la ruta de transmisión.

Además se utilizo la función de selección CASE para dar paso a los distintos movimientos del autómata según la lectura del puerto serial. Ver Código Anexo C

3.2.7. LA GRÁFICA DEL CIRCUITO

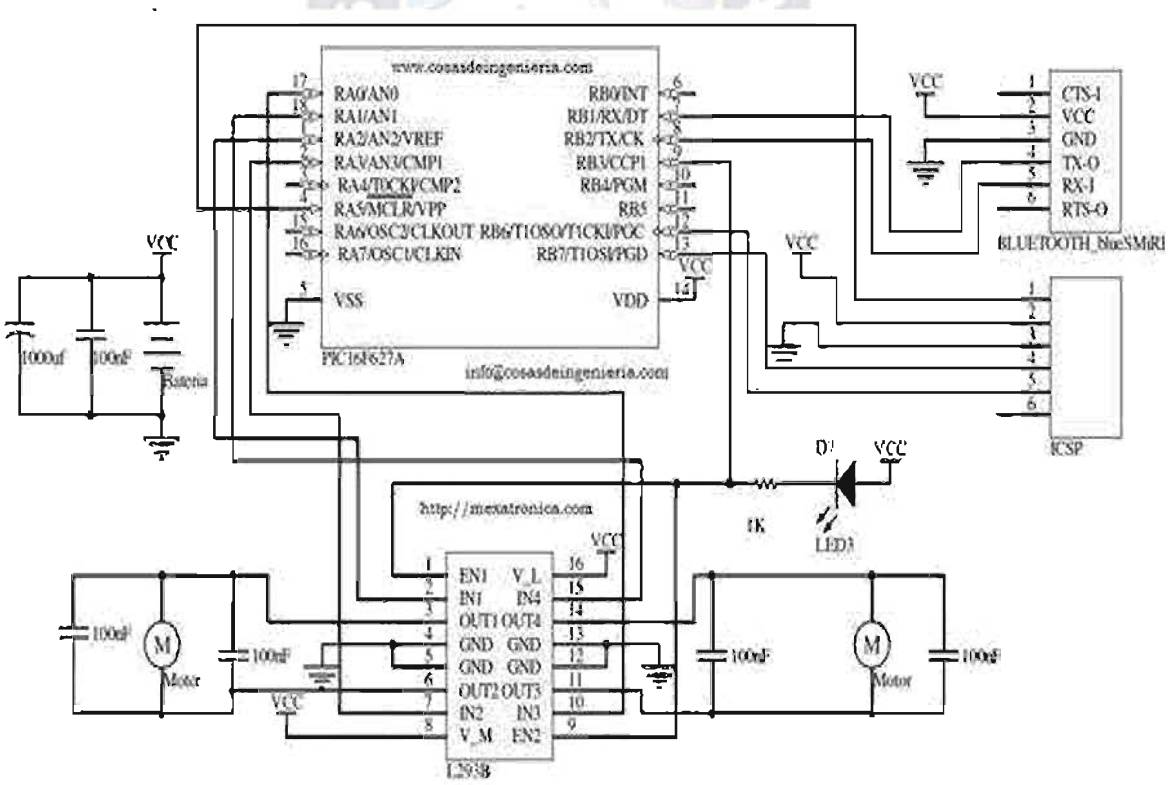


Figura III, 6 Esquema de un microcontrolador con dos motores
Fuente: [Elaboración Propia]

3.2.8. SIMULACIÓN EN PROTEUS

Para realizar la demostración del programa se utiliza el simulador Proteus que sirve para realizar en este caso la simulación de los movimientos básicos del autómata.

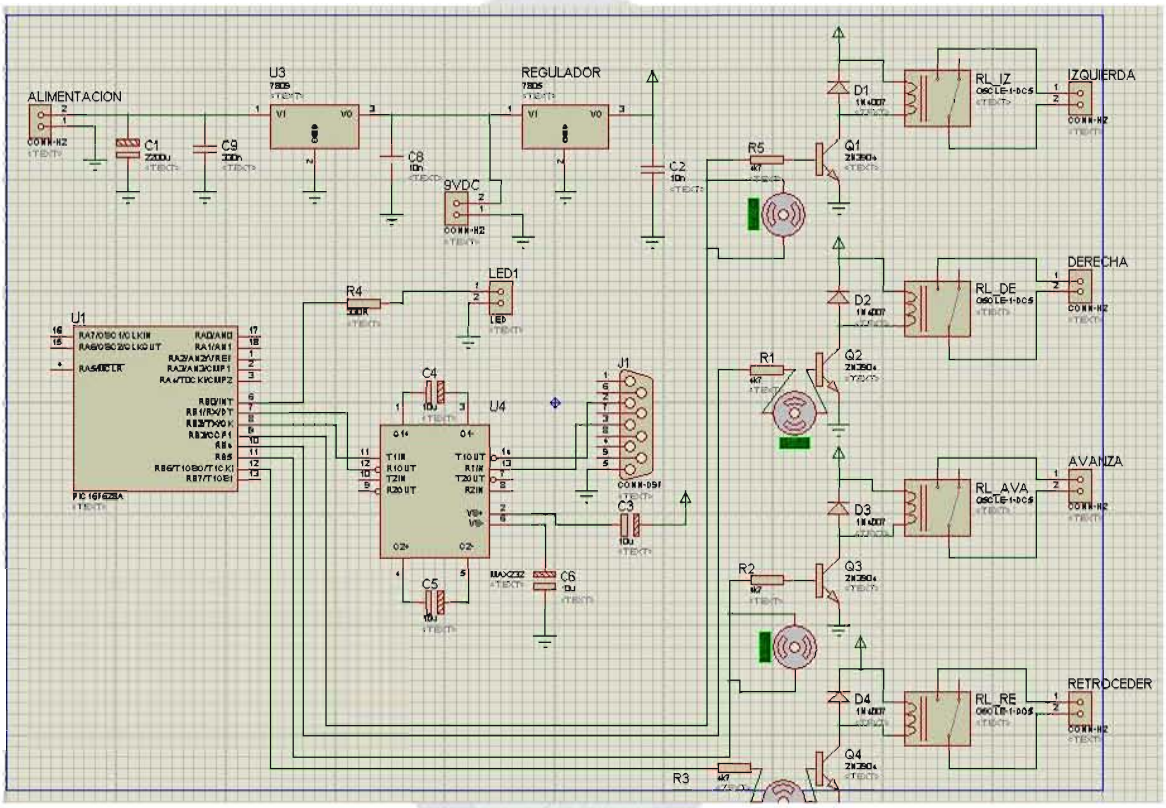


Figura III, 7 Pantalla de ISIS – PROTEUS donde se encuentra el microcontrolador.
Fuente: [Elaboración Propia]

3.2.9. DISEÑO DEL EQUIPO

3.2.9.1. PARTE ELECTRICA Y ELECTRONICA

Al igual que el sistema mecánico, el eléctrico y el electrónico ha sido diseñado localmente y concebido para ser construido con materiales de fácil consecución en medio nuestro.

3.2.9.2. COMPONENTES

Para la experimentación se vio como conveniente una gran variedad de componentes, muchos de los cuales se fueron incluyendo a medida que se avanza el modelo, los componentes utilizados son:

Nº	NOMBRE	NO PARTE	DESCRIPCIÓN
2	Limitadores	7805	Para eliminar el paso del voltaje
1	Microprocesador	16F628A	Para manejar el autómata
4	Condensador Cerámico	100nF	Eliminar el Ruido
1	Cristal	20MHz	Aumentar la velocidad de la comunicación
2	Servomotores		Truncados mover el carro
1	LED emisor de luz		Para indicar el encendido del autómata
3	Resistencia	1K,220	Limitador de corriente que circula hacia la LED

Tabla III, 1 Lista de Componentes
Fuente: [Elaboración Propia]

3.2.10. CONEXIÓN DEL TELÉFONO MÓVIL AL EQUIPO CENTRAL

El software diseñado en J2ME tiene como objetivo transmitir órdenes envía de comunicación bluetooth a la central maestro que son ordenes de movimiento del autómata.

Este programa está diseñado para establecer la comunicación entre el equipo celular y el modulo bluetooth que se encuentra en la placa del circuito, para ellos se detallan a continuación las órdenes que se implementaron:

1. En primer lugar se definen las variables que se utilizaran para controlar la comunicación.

   Service [] servicios;

   Bluetooth bt;

   Cliente c;

2. Se inicializa el objeto bluetooth con el cual se realizará la conexión al puerto serial.

   bt= new Bluetooth (this, Bluetooth.UUID_SERIALPORT);

3. Una vez iniciada la aplicación en el equipo celular se presentara una pantalla de bienvenida.

4. Se procede a buscar el dispositivo bluetooth que en este caso es el modulo Bluetooth Serial Converter.

5. Una vez encontrado visualiza la MAC ADRESS del dispositivo y procede a buscar puerto serial.

6. Cuando se logro la comunicación, ingresa al proceso de conectado y espera la pulsación de las teclas para direccionar el vehículo (robot).

7. El usuario en cualquier momento puede terminar la aplicación seleccionado la tecla salir.

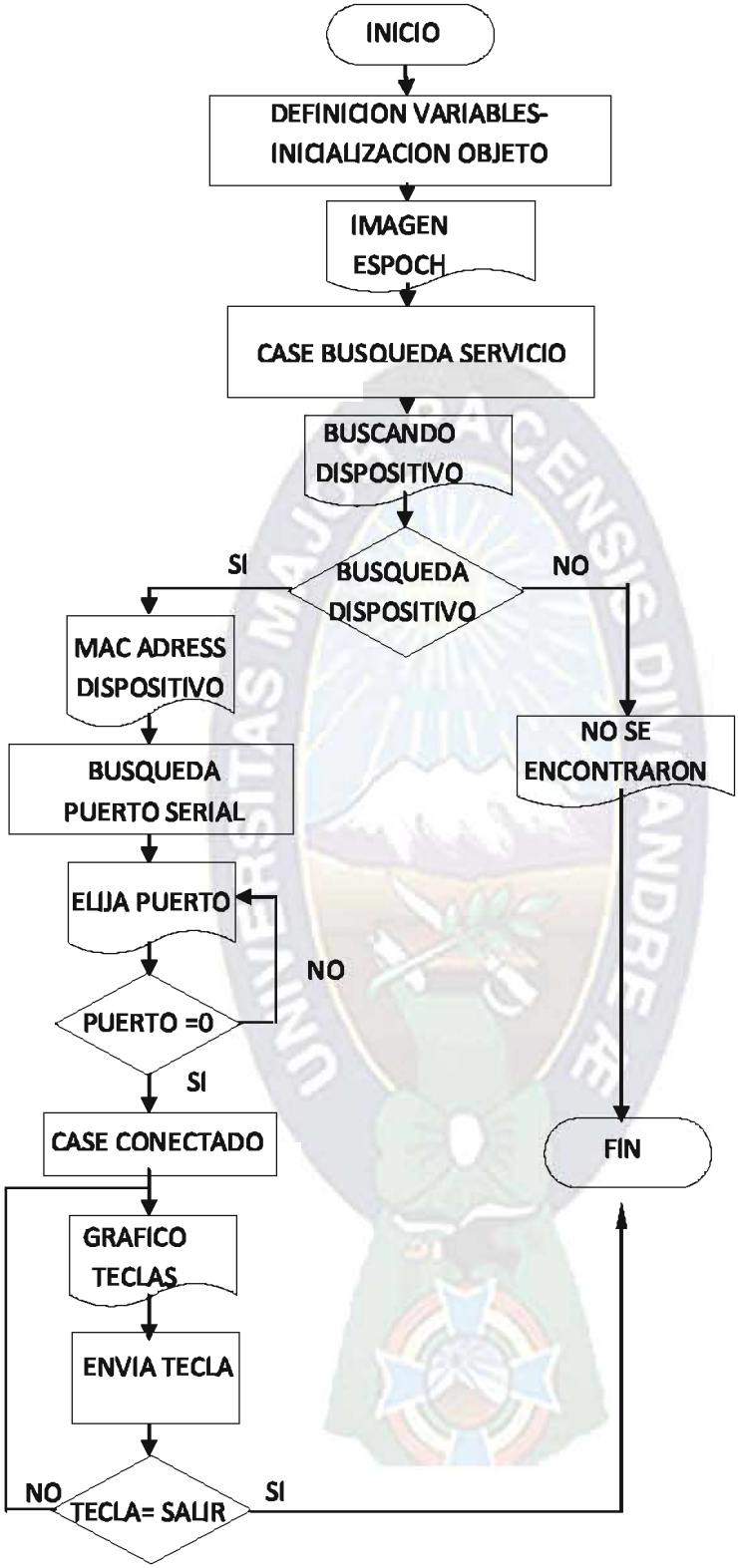


Figura III, 9 Ingreso y Ejecución de Comandos
Fuente: [elaboración Propia]

Los APIs Java para Bluetooth definen dos paquetes:

- Javax.bluetooth
- Javax.obex

En este caso se utilizo javax.bluetooth que nos permitió las búsquedas de dispositivos, búsquedas de servicios a través de:

1. Interface DiscoveryListener con los métodos:
   - deviceDiscovered
   - inquiryCompleted
   - servicesDiscovered
   - service_search_completed
   - service_search_terminated

En cuanto a las librerías se utilizo los packages:

Java.lang.-	Clases e interfaces de la maquina virtual
Java.util.-	Clases y utilidades estándar
Java.microedition.io.-	Clases e interfaces de Conexión Genérica CDLC
Java.io.-	Clases y paquetes estándar de E/S

Ver código Anexo A.

3.2.10.1. REGISTRO DE SERVICIO DE PUERTO SERIE.

Un SPP debe inicializar los servicios que ofrece y registrarlos en el SDDB. Un servicio de puerto serie viene representado por un par de objetos emparentados:

1. Un objeto que implementa el interfaz javax.mocroedition.io.StreamConnectorNotifier

Este objeto escucha conexiones clientes que demanden este servicio.

2. Un objeto que implemente el interfaz javax.bluetooth.ServiceRecord.

   Este objeto describe el servicio y como puede ser accedido por dispositivos remotos.

Para crear estos objetos la aplicación servidora usa el método Connector.open () con un argumento de conexión URL.

Invocando Connector.open() con un argumento conexión URL, este devuelve un StreamConnectionNotifier que representar el servicio SPP. La implementación de Connector.open() además crea un nuevo registro de servicio (service record) que representa el servicio SPP. Una implementación de un SPP debe realizar los siguientes pasos cuando crea el registro de servicio.

1. Crea un identificador de una canal servidor RFCOMM, chanN y asignarlo.
2. chanN es añadido al ProtocolDescritorList en el registro de servicio.
3. El UUID usado en conexión string para describir el tipo de servicio ofrecido es añadido al SrrviceClassIDList.
4. El atributo ServiceName es añadido al registro de servicio con el valor "SPPEx".

En el caso de un servicio run-before-connect, el registro de servicio es añadido a la SDDB la primera vez que la aplicación servidora llama a acceptAndOpen() en el StreamConnectionNotifier asociado. El registro de servicio se hace visible a potenciales clientes SPP cuando es añadida a la SDDB.

3.2.10.2. ESTABLECIMIENTO DE LA CONEXIÓN

Antes de que un cliente SPP pueda establecer una conexión con un servicio SPP, este debe previamente haber descubierto el servicio mediante el servicio discovery. Una conexión URL el cliente incluye la dirección Bluetooth del dispositivo servidor y

el identificador de canal del servidor. El método getConnectionURL() en el interfaz ServiceRecord se usa para obtener la conexión URL del cliente.

Invocando el método Connector.open() con una conexión URL del cliente, devuelve un objeto StreamConnection que representa la conexión SPP del lado del cliente.

StreamConnectión con = (StreamConnection) Connection.open ("btspp://dirección:identificador_canal").

3.2.11. CONEXION EQUIPO CENTRAL O MAESTRO A MODULO AUTÓMATA

Para esta etapa se diseño una aplicación en el Lenguaje Visual Basic en la cual se inserto varios componentes para trabajar con los datos del puerto serial. Para ello detallamos a continuación las instrucciones:

1. Colocamos los componentes MSComm  los cuales nos ayudan a procesar los datos del puerto serial a través de las siguientes instrucciones:

   MSComm1.PortOpen = True // Entrada

   MSComm2.PortOpen = True // Salida

2. Colocamos el componente Timer  que permite ejecutar el programa cada cierto intervalo de tiempo y de esta forma tener el ingreso y salida de los datos del puerto serial.

3. Empezamos a leer los puertos mediante : A=MSComm.Input

4. Procesamos la información de acuerdo a los datos e enviamos al puerto de salida la instrucción necesaria para el direccionamiento del autómata.

   MSComm2.Output ="A" 'Adelante

5. El usuario puede terminar el programa seleccionado el botón Cerrar

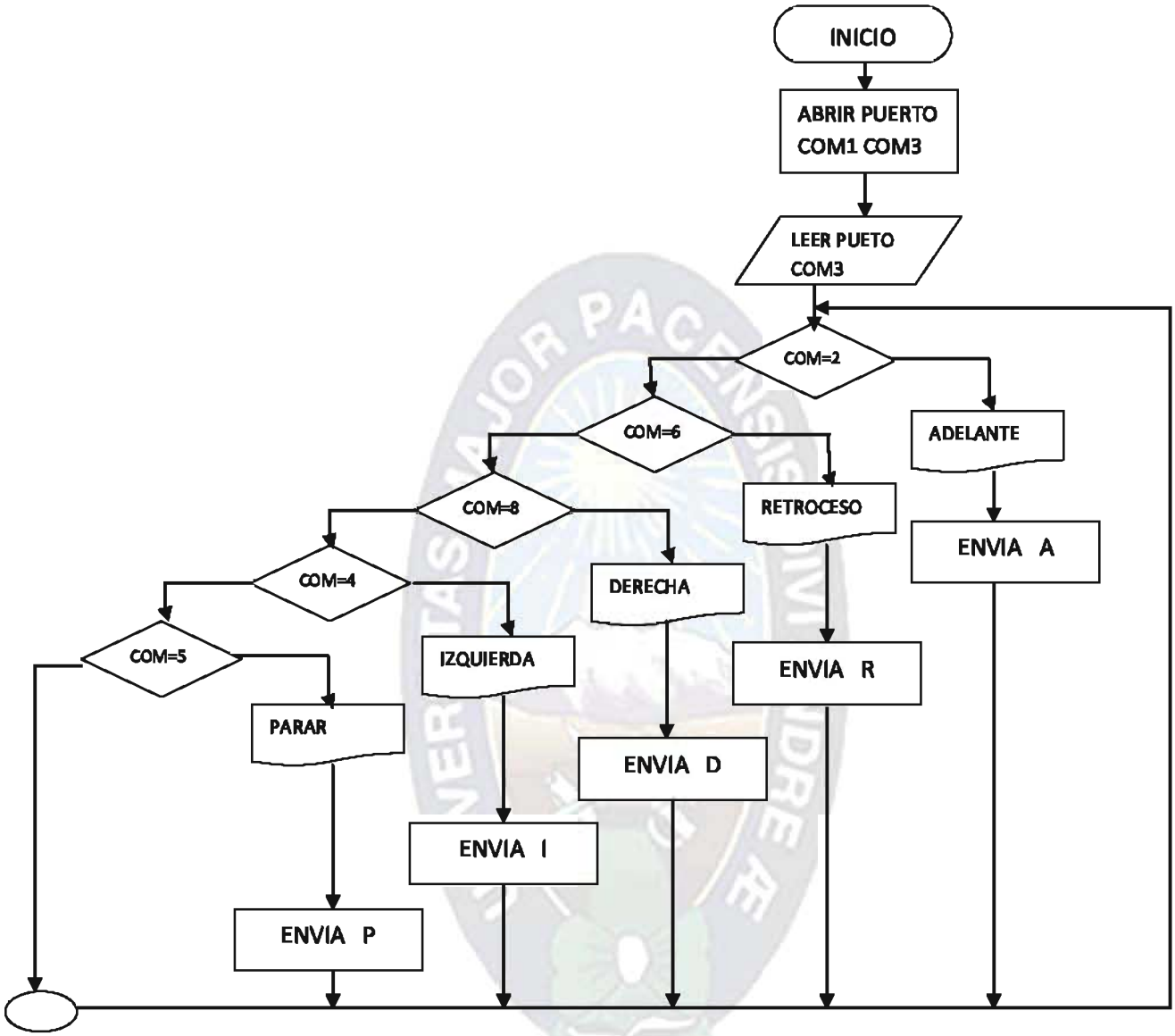


Figura III, 10 Ingreso y ejecución de Comandos – Equipo Central
Fuente: [elaboración Propia]

3.2.12. DISEÑO - CONTROL DEL AUTOMATA

Para el diseño del programa se utilizo Microcode Studio plus en el cual se utilizo la función HSEROUT que permite la comunicación serial a través de:

- DEFINA HSER_TXSTA 20h Determinación de la velocidad de transmisión.
- DEFINA HSER_BAUD 2400 Determina la ruta de transmisión.

Además se utilizo la función de selección CASE para dar paso a los distintos movimientos del autómata según la lectura del puerto serial. Ver Código Anexo B

3.2.13. COMPUTADOR PERSONAL

En este caso es recomendable utilizar una computadora Pentium IV o alguna de mayor capacidad en donde tenga la entrada de puerto paralelo o USB, se ejecutará la aplicación software desarrollada, todo el control del interface hardware se hace utilizando el puerto del computador.

El puerto se utilizaba generalmente para manejar impresoras, mouse. Sin embargo, dado que este puerto tiene un conjunto de entradas y salidas, se puede emplear para hacer prácticas experimentales de lectura de datos y control de dispositivos. También el puerto se puede utilizar como un interface de entrada y salida que funcione de modo subordinado a rutinas de software en este caso "Visual Basic".

CAPITULO IV

Resumen
En este presente capitulo se desarrollará la simulación, prueba y resultados del modelo propuesto.

4.1. PRUEBA DEL SISTEMA CON EL TELÉFONO MÓVIL

Aquí se probará el comportamiento de la unidad equipo central comandado desde el dispositivo móvil y se revisarán los procedimientos necesarios para ejecutar cada uno de los comandos disponibles en la unidad equipo central.

Los resultados obtenidos se irán indicando conforme avanza el proceso de explicación del diseño de pruebas realizadas.

4.1.1. PROCESO DE CONEXIÓN CON LA UNIDAD EQUIPO CENTRAL

Una vez encendido el dispositivo móvil y presionando su interruptor de bluetooth aparecerá la aplicación AUTOMATA. Al ingresar se visualiza la pantalla de presentación y nos da un mensaje de presionar una tecla, como se puede observar en la Figura IV, 1.

Figura IV, 1 Ingreso a la aplicación – Pantalla Principal
Fuente: [Elaboración Propia]

Inicia la búsqueda del dispositivo Bluetooth, al encontrar el dispositivo visualiza la MAC ADRESS y procede a elegir un puerto serial. Este proceso de indica en la Figura IV, 2.

Figura IV, 2 Buscando Dispositivos para elegir puerto serial
Fuente: [Elaboración Propia]

Una vez conectado con el puerto aparece una pantalla que nos permite elegir los movimientos del autómata con las teclas 2-Adelante, 8-Retroceso, 6-Derecha, 4-Izquierda. Este proceso de indica en la Figura IV, 3.

Figura IV, 3 Buscando Dispositivos para elegir puerto serial
Fuente: [Elaboración Propia]

4.1.1.1. PROCESO DE ENVÍO DE COMANDOS

Aquí se verán los procedimientos requeridos para encender la comunicación, los canales de salida, leer estados.

Con el mensaje de ingreso y detección de bluetooth en la pantalla del dispositivo móvil se puede ingresar las combinaciones siguientes:

Primera Tecla	Teclas Siguientes	Resultados
1	2	Enciende Canal de Salida Nº1
1	4	Enciende Canal de Salida Nº2
1	6	Enciende Canal de Salida Nº3
1	8	Enciende Canal de Salida Nº4

Tabla IV, 1 Comandos de encendido de Canal de Salida
Fuente: [Elaboración Propia]

Con el mensaje de ingreso de comando en la pantalla del dispositivo se puede ingresar las combinaciones siguientes

Primera Tecla	Resultados
5	Recibe estados de comunicación en la unidad central o maestro

Tabla IV, 2 Comando de Lectura de estado de Salida /Entrada
Fuente: [Elaboración Propia]

4.1.2. PROCESO DE PRUEBA DE RECEPCION CON LA UNIDAD EQUIPO CENTRAL

En primer lugar se ejecuta el programa prueba.exe

Controlador

Figura IV, 4 Programa ejecutable
Fuente: [Elaboración Propia]

En el cual se reciben las señales del puerto serial COM· (entrada del modulo Bluetooth) y las procesa según la elección de los movimientos.

Si se elige la tecla "2" del celular en el programa aparecerá la siguiente pantalla: ADELANTE.

Si se elige la tecla "8" del celular en el programa aparecerá la siguiente pantalla: RETROCESO.

Este proceso de indica en la Figura IV, 4.

Figura IV, 4 Movimiento ADELANTE – Movimiento RETROCESO
Fuente: [Elaboración Propia]

Si se elige la tecla "6" del celular en el programa aparecerá la siguiente pantalla: DERECHA.

Si se elige la tecla “4” del celular en el programa aparecerá la siguiente pantalla: IZQUIERDA

Este proceso de indica en la Figura IV, 5

.

Figura IV, 4 Movimiento DERECHA– Movimiento IZQUIERDA
Fuente: [Elaboración Propia]

Y por ultimo si elegimos la tecla “5” del celular en el programa aparecerá la siguiente pantalla: STOP

Figura IV, 5 Movimiento DERECHA– Movimiento IZQUIERDA
Fuente: [Elaboración Propia]

4.2. PRUEBA DE HIPÓTESIS

Para realizar una prueba de hipótesis se sigue un procedimiento.

- Determinar la hipótesis Nula "Ho" y Alternativa "Ha".
- Determinar el nivel de significancia. Este nivel representa la probabilidad de rechazar una hipótesis nula verdadera, matemáticamente se puede considerar cualquier valor ente cero y uno; pero para estudios de pruebas de hipótesis normalmente está entre 0.05 y 0.1. Este nivel se determina y se basa en las características del estudio y el riesgo que se considere aceptable de cometer el error tipo I.

  Nivel de significancia del estudio para el ejemplo: α =0.1
- Calcular los intervalos que implican ese nivel de significancia. Para dicho nivel de significancia los valores de Z son Z = =+/- 11,52

Ho = 128 us

Ha = x

a. Determinar el nivel de significancia

$\alpha = 0.1$

Nivel de significancia= +/-((1-0.1)/10)*128

Nivel de significancia=+/- 11,52

Rango mediante nivel de significancia

Por lo tanto:

X: (116.48 – 139.52)

b. Estadístico de la prueba

Calculando el estadístico de la prueba. El estadístico Z se calcula de la siguiente manera:

$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$ Se calcula la siguiente desviación estándar

$z = \frac{\bar{x} - \mu}{\sigma_{\bar{x}}}$ Se calcula el valor de Z tipificado.

μ	Promedio considerado por la hipótesis nula
$\tilde{x}$	Media de la muestra tomada.
σ	Desviación estándar de la muestra.
$\sigma_{\bar{x}}$	Desviación estándar tipificada
Z	Valor de Z tipificado

Estadístico de la prueba=+/-16.013

Rango mediante nivel estadístico

Por lo tanto:

X: (111.98 – 144.01)

Determinación hipótesis – Transmisión de Serial

Ho = 9600 bps

Ha = x

a. Determinar el nivel de significancia

α = 0.1

Nivel de significancia= +/-((1-0.1)/10)*9600

Nivel de significancia=+/- 864

Rango mediante nivel de significancia

Por lo tanto:

X: (8736 – 10464)

b. Estadístico de la prueba

Estadístico de la prueba=+/-610,940259

Rango mediante nivel estadístico

Por lo tanto:

X: (8989,06–10210,94)

CAPITULO V

Resumen

En este presente capitulo se podrá encontrar las conclusiones relacionadas con el desarrollo y culminación de la presente tesis, y las recomendaciones que se dan para el desarrollo posterior de aplicaciones similares en forma y fondo

.

5.1. CONCLUSIONES

1. El proyecto se diseñó utilizando diferentes lenguajes de programación entre ellos el J2ME que permitió realizar una programación estable y de fácil comprensión en un interfaz de comunicación inalámbrica entre el teléfono móvil y el servidor de recepción da datos utilizando sistema embebido. Para manejar el Standar Bluetooth se utilizo el paquete J2ME con su librería java. Bluetooth que nos permitió trabajar con el perfil de puerto serie (SPP) y otras funciones que facilito la tarea de programación del celular.

2. El diseño e implementación en el equipo central (Servidor) fue desarrollado utilizando el Lenguaje Visual Basic orientado a objetos lo que permitió controlar las señales de puerto serie a través de sus componentes MSComm y Timer para leer los puertos de forma continua,

3. En la programación del microprocesador se utilizo la función HSEROUT que envía uno o varios datos en serie y permite configurar el registro, la velocidad y la tasa de transmisión en un sistema embebido para recepción de datos y ejecutarlos.

4. Con los resultados obtenidos en las pruebas realizadas se puede comprobar que el modelo de sistema funciona en tiempo real ya que el intervalo en el envío y recepción está en función del tiempo entre el teléfono móvil y el autómata utilizando un equipo central como conexión entre los dos dispositivos, se demuestra que existe flujo de información.

5. Podemos decir que se ha logrado un nivel básico de comunicación entre distintos sistemas, lo que demuestra que es posible la comunicación entre teléfonos móviles y microcontroladores usando un equipo que funcione como nexo entre ellos, además de tener una comunicación completamente sin cables.

5.2. RECOMENDACIONES

Luego del diseño del modelo y aplicado en una versión preliminar, hay varias líneas a considerar para trabajos futuros, se recomienda:

1. Consiste en incluir la tutoría inteligente automatizada, dotando al entorno de agentes de forma que colaboren en la selección y sugerencia de conocimientos.

2. Se recomienda contar con políticas de seguridad, respaldos de información y administración de red, armar un plan de contingencia en caso de que se presente alguna eventualidad tales como virus, desastres físicos y lógicos, etc.

3. Dado que se realizo una versión reducida del entorno, es realizar una versión completa del mismo, con todos los contenidos de la materia, en este caso Programación.

4. Se puede mejorar la visualización en los estados de entrada y salida de información donde se halla instalado la unidad central (servidor).

5. Continuar con esta línea de investigación para aprovechar al máximo las posibilidades de comunicación inalámbrica con propósitos de seguridad y ejecución de ciertas tareas a distancia, habida cuenta que la disponibilidad de la telefonía móvil es amplia.

BIBLIOGRAFÍA

1. Referencias Escritas

[MUÑOZ,1998] Muñoz Razo Carlos (1998) "Como elaborar y asesorar una investigación" de tesis México: 1ª ed. Prentice Hall Hispanoamearicana S.A.

[ANGULO- 2009] Angulo Martin (2009) "Microcontroladores PIC" La solución de un Chip (pp 231-290) Madrid

[POLO- 2008] Polo Jesus (2008) "Arquitectura de microcontroladores" (pp 191 - 250) Mexico

[OLLERO- 2001] Ollero(2001) "Robótica manipuladores y robots móviles" Barcelona, Marcombo

[ANGULO- 2004] Angulo Jose Maria, Romero Susana, Angulo Ignacio (2004) "Diseño Practico con microcontroladores para todos".

[SABINO-1994] Sabino Carlos (1994) "Como hacer una tesis" Venezuela – Caracas.

[DOGAN-2007] Dogan Ibrahin(2007) "Programación de Microcontroladores PIC" Barcelona – Marcombo

[MARTINEZ- 1999] Angulo Usateggui – Angulo Martinez (1999) "Microcontroladores PIC. Diseño practico de aplicaciones". Editorial Mc Graw Hill.

PEATMAN-2005 Peatman John (2005) "Introducción a los microcontroladores"edición UMSNH-FIE

GARCIA-2008 Garcia Breijo Eduardo(2008)"Compilador C CCS y simulador PROTEUS para microcontroladores PIC" Barcelona-Marcombo

[SEDRA-2006] Sedra Adel(2006) "Circuitos Microelectronicos" Mexico

2. Referencias Electrónicas

[WIKIPEDIA 2011]	http://es.wikipedia.org (Última consulta Septiembre del 2011)
[HISPAVILA 2011]	Manual Práctico de PROTEUS VSM http://www.hispavila.com\3ds\chipspic\manualproteus.html (Última consulta Noviembre del 2011)
[ELECTRONICA 2011]	http://www.casadomo.com (Última consulta Junio del 2011)
[MEXATRONICA 2011]	Conexión y configuración del modulo Bluetooth http://www.Mexatronica.com/?p=26 (Última Consulta Noviembre del 2011)
[MICROS Y MACROS 2011]	Enlace de Basic con PROTEUS http://micros.mforos.com/.../9295693-como-enlazo-proteus-con-visual-bas. (Última consulta Noviembre del 2011)
[AUTOMATA 2011]	Conceptos de Autómata http://kefamare.galeon.com/automata.htm (Última Consulta)
[MICROCHIP 2011]	Microchip - Grabadores http://microchip.com (Última consulta Octubre 2011)
[TDOROBOT 2011]	Concepto de robots, motores y confoiguraciones http://www.todorobot.com.ar/productos/motores/motores.stpper (Última Consulta Septiembre del 2011)
[NIPLE 2011]	Entorno Visual de Desarrollo para microcontroladores PIC www.niplesoft.net (Última consulta Noviembre del 2011)
[MIKROELEKTR 2011]	Creating the First Project in mikroBasic for PIC www.mikroe.com

(Última consulta noviembre del 2011)

[LARROSA 2011] Programación y diseño de dispositivos mediante microcontroladores PIC

http://mimosa.pntic.mec.es/flarrosa/intropic.pdf

(Última Consulta octubre del 2011)

ANEXO A

CODIGO FUENTE CELULAR

```
**************************************
**************************************

import processing.core.*;
import processing.buetooth.*;


public class carro extends PMIDlet
  {
      //M/u00e1maquina de estados
      final int ESTADO_INICIO=0;

      final int ESTADO_BUSCA=1;
      final int
ESTADO_CONECTADO=2;
      int estado;
      //librer/u00eda bluetooth
      Bluetooth bt;
      //servicios descubiertos
      Service[] servicios;
      //mensaje de estado
      String mens;
      //Connexi/u00f3n srial
      Client c;
      //fuente
      PFont font;

////////////////////////////////

      public void setup()
          {
           font =loadFont();
           textFont(font);
           //inicializa objeto
bluetooth
           bt=new
Bluetooth(this,bluetooth.UUID_SERIAL
PORT);

estado=ESTADO_INICIO;
          }
////////////////////////////////
      public void destroy()
          {
           bt.stop();
          }
      public void draw()
          {

background(0xFF,0xFF,0xFF);//rgb-
blanco
           //Dibuja la pantalla
dependiendo de qu//u009 estado
tienela m/u00e1maquina
           switch(estado)
```

```
                    {
                        case
ESTADO_INICIO:
                        int
c1=color(50,185,0);
                        fill(c1);
                        rect(1,1,width-
2,height-2);
                        int
c2=color(255,235,0);
                        fill(c2);
                        rect(15,15,width-
30,height-30);
                        //PImage b;
                        //iMAGES MUST
BE IN THE "data"directory to load
correctly
                       //
b=loadImage("escudo.PNG");
                        //image(b,30,30);
                        fill(0);

textAlign(CENTER);

text("ESPOCH\nAPLICACION
CARRO\n",2,4,width-4,height-4);
                        textAling(LEFT);
                        teXt("Presione
una tecla\n",1,100,width-2,height-2);
                        break;
                        case
ESTADO_BUSCA:
                        fill(0);
                        textAlign(LEFT);
                        //si no se han
encontrado servicios
                        if(servicios ==
null)

text("APLICACION
CARRO\nB\u00fasqueda\n"+mens,2,2,
width-4,hwight-4);
                                //si ya hay al
menos un servicio
                        else
                          {
                            String lista =
"elija un puerto serial:\n";
                            for(int
i=0,conteo=length(servicios);i<conteo;
++i)
                            lista +=i+".-
"+servicios[i].divice.name+"\n";//hace
una lista con nombres de dispisitivos
hallados

text(lista,2,2,width-4,height-4);
                          }
                        break;
```

```
                    case
ESTADO_CONECTADO:
                    fill(0);

textAling(CENTER);
                    text("Presione
2,4,6,8 para direccionar el
auto\n",2,2,width-4,height-4);
                    //PImage t;
                    //Images must be
in the "data directory to load correctly"

//t=loadImage("imagenflechacelular.PN
G");
                  //image(t,25,25);
                    break;
                    }
               }
////////////////////////////////////
       public void keyPressed()
               {
                //M/u00e1quina de
estados
                switch(estado)
                    {
                    case
ESTADO_INICIO:
        servicios_null;//limpia los
servicios
                        bt.find(); //comienza
bsquedas de bluetooth

      estado=ESTADO_BUSCA:
                        mens="Buscando
Dispositivos Bluetooth....\n";
                        break;

                        case
ESTADO_BUSCA:
                        //si ya hay al
menos un dispisitivo encontrado
                        if (servicios!=null)
                            {
                             //verfica
que haya presionado algo del 0 al 9
                             if
((key>='0')&& (key<='9'))
                             {
                              int i=key -
'0'; convierte ascii a entero
                             //si la
tecla presionada existe como puerto
disponible

if(i<length(servicios))
                                 {
```

```
mens="Conectado..";

c=servicios[i].connect();

estado=ESTADO_CONECTADO;
                                    }
                                }
                            }
                    break;


                    case
ESTADO_CONECTADO:
                      //busqueda por el
puerto serial que el usurip presiona
                    c.write(key);
                      c.flush();
                      break;
                      }
                }
 ////////////////////////////////////////
        public void libraryEvent(Object
library, int event,Object data)
            {
                //si la librria no fue
bluettoth
                if(library!=bt)
                  return;
                switch(event)
                    {
                      //si encontro un
disposiito
                      case
Bluetooth.EVENT_DISCOVER_DEVIC
E:
mens="dispositivo:"+((Device)data).ad
dress;
                      break;
                      // ya se
encontrnaron todos losdispositivos
posibles
                      case
Bluetooth.EVENT_DISCOVER_DEVIC
E_COMPLETED:
                      mens="Se
encontraron"+length((Device[])data)+"d
ispositivos,\n"+"bucando puertos
seriales...";
                      break;
                      //ENCONTRO UN
PUERTO SERIAL
                      case
Bluetooth.EVENT_DISCOVER_SERVI
CE:
                      mens="Puerto
serial
```

```
en"+((Service[])data)[0].device.address
;
                    break;
                            //busqueda de
puertos seriales temrinados


                            case
Bluetooth.EVENT_DISCOVER_SERVI
CE_COMPLETED:


servicios=(Service[])data;//juanta los
puertos encontrados en una arrelgo
                            mens="elija un
puerto";
                            break;
                            //ya se encontro
al cliente


                            case
Bluetooth.EVENT_CLIENT_CONNEC
TED:
                            C=(Client)data;
                            mens="Conexion
exitosa :D";
                        }
                }
}
```

ANEXO B

Código Fuente Microcontrolador PIC

```
*****************************************
*****************************************
        CMCON =7
        TRISA=%00000000
        TRISB=%00000010
        DEFINE HSER_RCSTA 90h
        DEFINE HSER_TXSTA 20h
        DEFINE HSER_BAUD 2400
        ***************
        RETRO VAR PORTB.6
        AVANZA VAR PORTB.5
        DERECHA VAR PORTB.4
        IZQUIERDA VAR PORTB.3
        C VAR BYTE 'M = MOVER
VEHICULO
        M VAR BYTE 'M = MOVER
VEHICULO
        AUX VAR BYTE
        C=0
        High PORTB.0
        Pause 100
        LOW PORTB.0
        PAUSE 100
        HIGH PORTB.0
        PAUSE 100
        LOW PORTB.0
        PAUSE 100     'A = AVANCE
                      'R =RETRO
                      ' I =IZQUIERDA
                      'D = DERECHA
        HIGH AVANZA
        PAUSE 100
        HIGH RETRO
        PAUSE 100

        HIGH IZQUIERDA
        PAUSE 100
        HIGH DERECHA
        PAUSE 100
        INICIO:
        ' PRMERA PULSADA
        HSERIN[M]
        IF M ="A" AND C=0 THEN
        LOW AVANZA
        HIGH RETRO
        HIGH IZQUIERDA
        HIGH DERECHA
        HIGH PORTB.0
```

```
PAUSE 300
LOW PORTB.0
C=1
AUX=M
M=0
GOTO INICIO
END IF
IF M="R" AND C=0 THEN
HIGH AVANZA
LOW RETRO
HIGH IZQUIERDA
HIGH DERECHA
HIGH PORTB.0
PAUSE 300
LOW PORTB.0
C =1
AUX=M
M=0
GOTO INICIO
END IF
IF M="I" AND C=0 THEN
HIGH AVANZA
HIGH RETRO
LOW IZQUIERDA
HIGH DERECHA
HIGH PORTB.0
PAUSE 300
LOW PORTB.0
C=1
AUX =M
M=0
GOTO INICIO
END IF
IF M="D" AND C=0 THEN
HIGH AVANZA
HIGH RETRO
HIGH IZQUIERDA
LOW DERECHA
HIGH PORTB.0
PAUSE 300
LOW PORTB.0
C=1
AUX= M
M=0
GOTO INICIO
END IF
IF M="P" AND C=0 THEN
HIGH AVANZA
HIGH RETRO
HIGH IZQUIERDA
```

```
        HIGH DERECHA
        HIGH PORTB.0
        PAUSE 300
        LOW PORTB.0
        C=1
        AUX=M
        M=0
        GOTO INICIO
        END IF
        'SEGUNDA PULSADA
        IF AUX= "A" AND C>=1  then
        SELECT CASE M
        CASE "A"
            LOW AVANZA '0 = ACTIVA
LOGICA INVERSA
            HIGH RETRO
            HIGH IZQUIERDA
            HIGH DERECHA
            AUX ="A"
            C=1
            CASE "R"
            HIGH AVANZA '0 = ACTIVA
LOGICA INVERSA
            LOW RETRO
            HIGH IZQUIERDA
            HIGH DERECHA
            AUX ="R"
            C=1
            CASE "I"
            LOW AVANZA '0 = ACTIVA
LOGICA INVERSA

            HIGH RETRO
            LOW IZQUIERDA
            HIGH DERECHA
            C=1
            CASE "D"
            LOW AVANZA '0 = ACTIVA
LOGICA INVERSA
            HIGH RETRO
            HIGH IZQUIERDA
            LOW DERECHA
            C=1
            CASE"P"
            HIGH AVANZA
            HIGH RETRO
            HIGH IZQUIERDA
            HIGH DERECHA
            C=0
            AUX =0
            END SELECT
            M=0
```

```
        HIGH PROTB.0
        PAUSE 300
        LOW PORTB.0
        IF C!=0 THEN
        C=C+1
        END IF
        'M=0
        GOTO INICIO
        END IF
        '************************
        IF AUX ="R" AND C>=1
THEN
        SELECT CASE M
        CASE "A"
        LOW AVANZA '0 = ACTIVA
LOGICA INVERSA
        HIGH RETRO
        HIGH IZQUIERDA
        HIGH DERECHA
        AUX "A"
        C=1
        CASE "R"
        HIGH AVANZA '0= ACTIVA
LOGICA INVERSA
        LOW RETRO
        HIGH IZQUIERDA
        GIGH DERECHA
        AUX ="R"
        C=1
        CASE "I"
        HIGH AVANZA'0= ACTIVA
LOGICA INVERSA
        LOW RETRO
        LOW IZQUIERDA
        HIGH DERECHA
        C=1
        CASE "D"
        HIGH AVANZA '0= ACTIVA
LOGICA INVERSA
        LOW RETRO
        HIGH IZQUIERDA
        LOW DERECHA
        C=1
        CASE "P"
        HIGH AVANZA
        HIGH RETRO
        HIGH IZQUIERDA
        HIGH DERECHA
        C=0
        AUX =0
        END SELECT
        M=0
```

```
        HIGH PORTB.0
        PAUSE 30O
        LOW PORTB.0
        IF C!=0 THEN
        C=C+1
        END IF
        'M=0
        GOTO INICIO
        END IF
        IF AUX ="I"  AND C=1 THEN
        SELECT CASE M
        CASE "A"
        LOW AVANZA '0=ACTIVA
LOGICA INVERSA
        HIGH RETRO
        LOW IZQUIERDA
        HIGH DERECHA
        AUX ="A"
        C=1
        CASE "R"
        HIGH AVANZA '0=ACTIVA
LOGICA INVERSA
        LOW RETRO
        LOW IZQUIERDA
        HIGH DERECHA
        AUX="R"
        C=1
        'CASE "D"
        ' HIGH AVANZA '0 = CTIVA
LOGICA INVERSA
        ' HIGH RETRO
        ' HIGH IZQUIERDA
        'LOW DERECHA
        CASE "P"
        HIGH AVANZA
        HIGH RETRO
        HIGH IZQUIERDA
        HIGH DERECHA
        C=0
        AUX =0
        END SELECT
        M=0
        HIGH PORTB.0
        PAUSE 300
        LOW PORTB.0
        IF C!=0 THEN
         C=C+1
         END IF
         'm=0
         GOTO INICIO
        END IF
```

```
            if Aux = "I" and c =1 then
            select case m
            case "A"
              low avanza '0= ACTIVA
LOGICA INVERSA
              high retro
              low izquierda
              high derecha
              aux ="A"
              C=1
              case "R"
              high avanza '0 = activa
logica inversa
              LOW RETRO
              LOW IZQUIERDA
              HIGH DERECHA
              AUX ="R"
              c=1
            'case "D"
            ' high avnza '0= activa
logica inversa
            ' high retro
            ' high izquierda
            ' low derecha
            case "P"
            high avanza
            high retro
            high izquierda
            high derecha
            c=0
            aux=0
            end select
            m=0
            high PORTB.0
            PAUSE 300
            LOW PORTB.0
            IF C!=O THEN
            C=C+1
            END IF
            ' M=0
            GOTO INICIO
            END IF
            IF AUX ="D" AND C=1
THEN
            SELECT CASE M
            CASE "A"
            LOW AVANZA '0=ACTIVA
LOGICA INVERSA
            HIGH RETRO
            HIGH IZQUIERDA
            LOW DERECHA
            AUX = "A"
```

```
            CASE "R"
            HIGH AVANZA'0= ACTIVA
LOGICA INVERSA
            LOW RETRO
            HIGH IZQUIERDA
            LOW DERECHA
            AUX ="R"
'           C=1
'           CASE "I"
'           HIGH AVANZA '0=
ACTIVA LOGICA INVERSA
            ' HIGH RETRO
            ' LOW IZQUIERDA
            ' HIGH DERECHA
            CASE "P"
            HIGH AVANZA
            HIGH RETRO
            HIGH IZQUIERDA
            HIGH DERECHA
            C=0
            AUX =0
            END SELECT
            M=0
            HIGH PORTB.0
            PAUSE 300
            LOW PORTB.0
            IF C!=0 THEN
            C=C+1
            END IF
            'M=0
            GOTO INICIO
            END IF
            ' FINAL PULSACIONES
            GOTO INICIO
```

ANEXO C

****

****

MICROCHIP

PIC16F627A/628A/648A

18-pin FLASH-Based 8-Bit CMOS Microcontrollers

High Performance RISC CPU:

- Operating speeds from DC - 20 MHz
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect and Relative Addressing modes
- 35 single word instructions
  - All instructions single cycle except branches

Special Microcontroller Features:

- Internal and external oscillator options
  - Precision Internal 4 MHz oscillator factory calibrated to ±1%
  - Low Power Internal 37 kHz oscillator
  - External Oscillator support for crystals and resonators.
- Power saving SLEEP mode
- Programmable weak pull-ups on PORTB
- Multiplexed Master Clear/Input-pin
- Watchdog Timer with independent oscillator for reliable operation
- Low voltage programming
- In-Circuit Serial Programming™ (via two pins)
- Programmable code protection
- Brown-out Reset
- Power-on Reset
- Power-up Timer and Oscillator Start-up Timer
- Wide operating voltage range. (2.0 - 5.5V)
- Industrial and extended temperature range
- High Endurance FLASH/EEPROM Cell
  - 100,000 write FLASH endurance
  - 1,000,000 write EEPROM endurance
  - 100 year data retention

Low Power Features:

- Standby Current:
  - 100 nA @ 2.0V, typical
- Operating Current:
  - 12 µA @ 32 kHz, 2.0V, typical
  - 120 µA @ 1 MHz, 2.0V, typical
- Watchdog Timer Current
  - 1 µA @ 2.0V, typical
- Timer1 oscillator current:
  - 1.2 µA @ 32 kHz, 2.0V, typical
- Dual Speed Internal Oscillator:
  - Run-time selectable between 4 MHz and 37 kHz
  - 4 µs wake-up from SLEEP, 3.0V, typical

Peripheral Features:

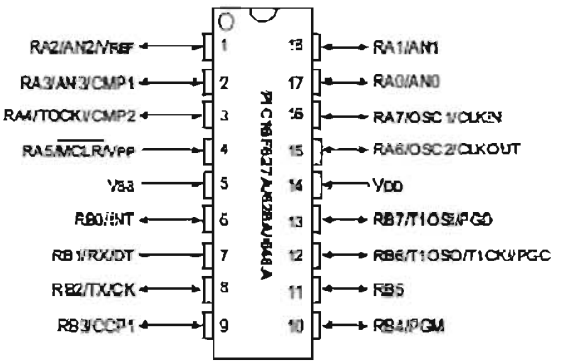
- 16 I/O pins with individual direction control
- High current sink/source for direct LED drive
- Analog comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (V_{REF}) module
  - Selectable internal or external reference
  - Comparator outputs are externally accessible
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Timer1: 16-bit timer/counter with external crystal/clock capability
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Capture, Compare, PWM module
  - 16-bit Capture/Compare
  - 10-bit PWM
- Addressable Universal Synchronous/Asynchronous Receiver/Transmitter USART/SCI

Presentado por:

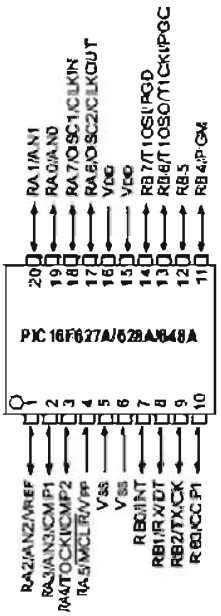
Device	Program Memory	Data Memory		I/O	CCP (PWM)	USART	Comparators	Timers 8/16-bit
	FLASH (words)	SRAM (bytes)	EEPROM (bytes)					
PIC16F627A	1024	224	128	16	1	Y	2	2/1
PIC16F628A	2048	224	128	16	1	Y	2	2/1
PIC16F648A	4096	256	256	16	1	Y	2	2/1

Pin Diagrams

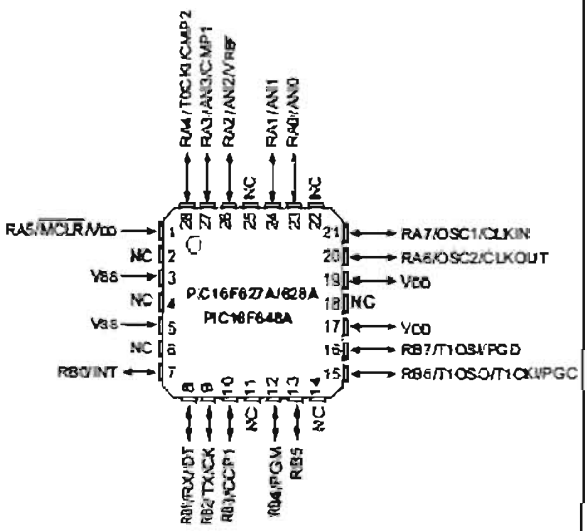
PDIP, SOIC



SSOP



28-Pin QFN



1.0 GENERAL DESCRIPTION

The PIC16F627A/628A/648A are 18-Pin FLASH-based members of the versatile PIC16CXX family of low cost, high performance, CMOS, fully-static, 8-bit microcontrollers.

All PICmicro® microcontrollers employ an advanced RISC architecture. The PIC16F627A/628A/648A have enhanced core features, eight-level deep stack, and multiple internal and external interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 14-bit wide instruction word with the separate 8-bit wide data. The two-stage instruction pipeline allows all instructions to execute in a single-cycle, except for program branches (which require two cycles). A total of 35 instructions (reduced instruction set) are available, complemented by a large register set.

PIC16F627A/628A/648A microcontrollers typically achieve a 2:1 code compression and a 4:1 speed improvement over other 8-bit microcontrollers in their class.

PIC16F627A/628A/648A devices have integrated features to reduce external components, thus reducing system cost, enhancing system reliability and reducing power consumption.

The PIC16F627A/628A/648A has 8 oscillator configurations. The single-pin RC oscillator provides a low cost solution. The LP oscillator minimizes power consumption, XT is a standard crystal, and INTOSC is a self-contained precision two-speed internal oscillator. The HS is for High-Speed crystals. The EC mode is for an external clock source.

The SLEEP (Power-down) mode offers power savings. Users can wake-up the chip from SLEEP through several external interrupts, internal interrupts and RESETS.

A highly reliable Watchdog Timer with its own on-chip RC oscillator provides protection against software lock-up.

Table 1-1 shows the features of the PIC16F627A/628A/648A mid-range microcontroller families.

A simplified block diagram of the PIC16F627A/628A/648A is shown in Figure 3-1.

The PIC16F627A/628A/648A series fits in applications ranging from battery chargers to low power remote sensors. The FLASH technology makes customizing application programs (detection levels, pulse generation, timers, etc.) extremely fast and convenient. The small footprint packages makes this microcontroller series ideal for all applications with space limitations. Low cost, low power, high performance, ease of use and I/O flexibility make the PIC16F627A/628A/648A very versatile.

1.1 Development Support

The PIC16F627A/628A/648A family is supported by a full-featured macro assembler, a software simulator, an in-circuit emulator, a low cost in-circuit debugger, a low cost development programmer and a full-featured programmer. A Third Party "C" compiler support tool is also available.

TABLE 1-1: PIC16F627A/628A/648A FAMILY OF DEVICES

		PIC16F627A	PIC16F628A	PIC16F648A	PIC16LF627A	PIC16LF628A	PIC16LF648A
Clock	Maximum Frequency of Operation (MHz)	20	20	20	4	4	4
Memory	FLASH Program Memory (words)	1024	2048	4096	1024	2048	4096
	RAM Data Memory (bytes)	224	224	256	224	224	256
	EEPROM Data Memory (bytes)	128	128	256	128	128	256
Peripherals	Timer module(s)	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2
	Comparator(s)	2	2	2	2	2	2
	Capture/Compare/PWM modules	1	1	1	1	1	1
	Serial Communications	USART	USART	USART	USART	USART	USART
	Internal Voltage Reference	Yes	Yes	Yes	Yes	Yes	Yes
	Interrupt Sources	10	10	10	10	10	10
	I/O Pins	16	16	16	16	16	16

2.0 PIC16F627A/628A/648A DEVICE VARIETIES

A variety of frequency ranges and packaging options are available. Depending on application and production requirements, the proper device option can be selected using the information in the PIC16F627A/628A/648A Product Identification System, at the end of this data sheet. When placing orders, please use this page of the data sheet to specify the correct part number.

2.1 FLASH Devices

FLASH devices can be erased and re-programmed electrically. This allows the same device to be used for prototype development, pilot programs and production.

A further advantage of the electrically erasable FLASH is that it can be erased and reprogrammed in-circuit, or by device programmers, such as Microchip's PICSTART® Plus, or PRO MATE® II programmers.

2.2 Quick-Turnaround-Production (QTP) Devices

Microchip offers a QTP Programming Service for factory production orders. This service is made available for users who chose not to program a medium to high quantity of units and whose code patterns have stabilized. The devices are standard FLASH devices but with all program locations and configuration options already programmed by the factory. Certain code and prototype verification procedures apply before production shipments are available. Please contact your Microchip Technology sales office for more details.

2.3 Serialized Quick-Turnaround-Production (SQTP℠) Devices

Microchip offers a unique programming service where a few user-defined locations in each device are programmed with different serial numbers. The serial numbers may be random, pseudo-random or sequential.

Serial programming allows each device to have a unique number, which can serve as an entry-code, password or ID number.

3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC16F627A/628A/648A family can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC16F627A/628A/648A uses a Harvard architecture, in which program and data are accessed from separate memories using separate busses. This improves bandwidth over traditional von Neumann architecture where program and data are fetched from the same memory. Separating program and data memory further allows instructions to be sized differently than 8-bit wide data word. Instruction opcodes are 14-bits wide making it possible to have all single word instructions. A 14-bit wide program memory access bus fetches a 14-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions (35) execute in a single-cycle (200 ns @ 20 MHz) except for program branches.

Table 3-1 lists device memory sizes (FLASH, Data and EEPROM).

TABLE 3-1: DEVICE MEMORY LIST

Device	Memory		
	FLASH Program	RAM Data	EEPROM Data
PIC16F627A	1024 x 14	224 x 8	128 x 8
PIC16F628A	2048 x 14	224 x 8	128 x 8
PIC16F648A	4096 x 14	256 x 8	256 x 8
PIC16LF627A	1024 x 14	224 x 8	128 x 8
PIC16LF628A	2048 x 14	224 x 8	128 x 8
PIC16LF648A	4096 x 14	256 x 8	256 x 8

The PIC16F627A/628A/648A can directly or indirectly address its register files or data memory. All Special Function Registers, including the program counter, are mapped in the data memory. The PIC16F627A/628A/648A have an orthogonal (symmetrical) instruction set that makes it possible to carry out any operation, on any register, using any Addressing mode. This symmetrical nature and lack of 'special optimal situations' make programming with the PIC16F627A/628A/648A simple yet efficient. In addition, the learning curve is reduced significantly.

The PIC16F627A/628A/648A devices contain an 8-bit ALU and working register. The ALU is a general purpose arithmetic unit. It performs arithmetic and Boolean functions between data in the working register and any register file.

The ALU is 8-bit wide and capable of addition, subtraction, shift and logical operations. Unless otherwise mentioned, arithmetic operations are two's complement in nature. In two-operand instructions, typically one operand is the working register (W register). The other operand is a file register or an immediate constant. In single operand instructions, the operand is either the W register or a file register.

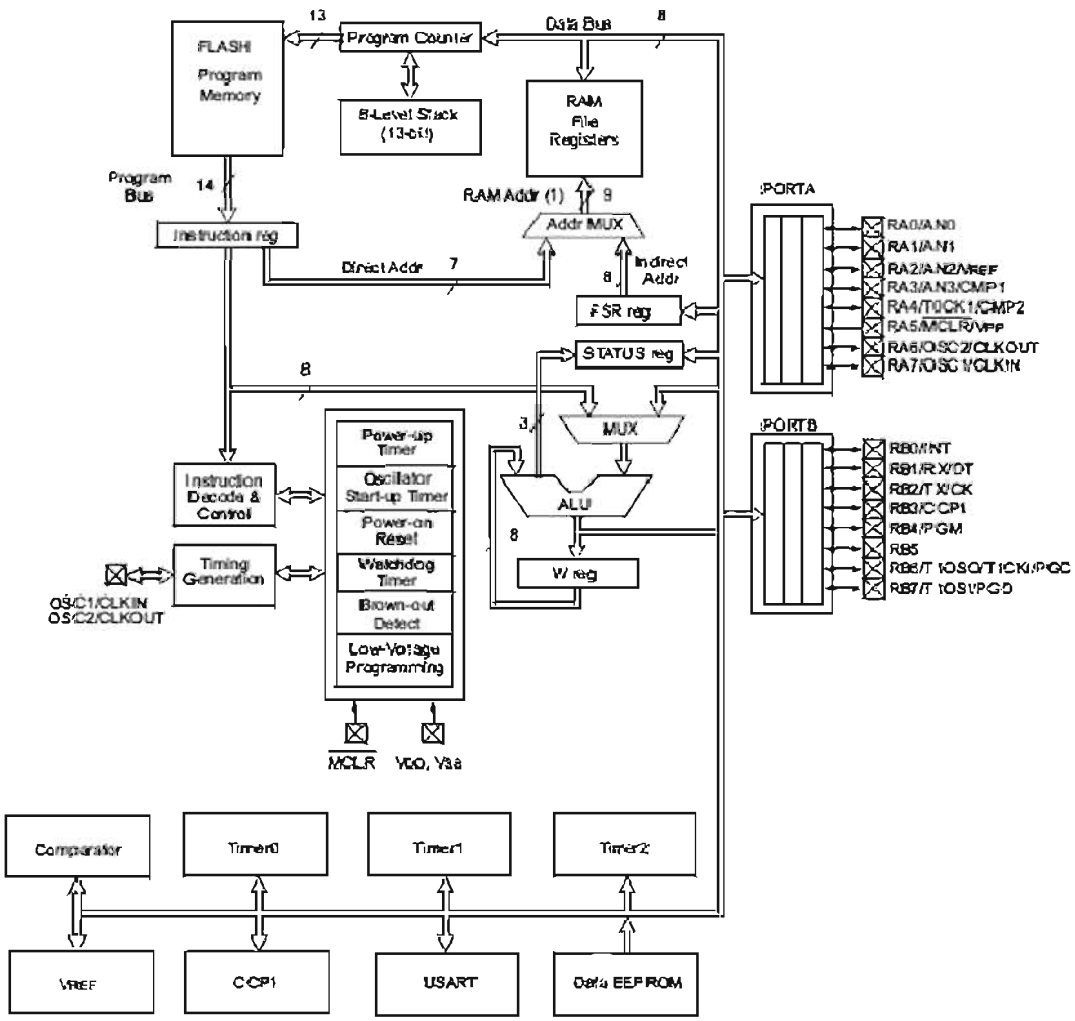
The W register is an 8-bit working register used for ALU operations. It is not an addressable register.

Depending on the instruction executed, the ALU may affect the values of the Carry (C), Digit Carry (DC), and Zero (Z) bits in the STATUS register. The C and DC bits operate as a $\overline{\text{Borrow}}$ and $\overline{\text{Digit Borrow}}$ out bit, respectively, bit in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

A simplified block diagram is shown in Figure 3-1, and a description of the device pins in Table 3-2.

Two types of data memory are provided on the PIC16F627A/628A/648A devices. Non-volatile EEPROM data memory is provided for long term storage of data such as calibration values, look up table data, and any other data which may require periodic updating in the field. These data are not lost when power is removed. The other data memory provided is regular RAM data memory. Regular RAM data memory is provided for temporary storage of data during normal operation. Data are lost when power is removed.

FIGURE 3-1: BLOCK DIAGRAM



Note: Higher order bits are from the STATUS register.

TABLE 3-2: PIC16F627A/628A/648A PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
RA0/AN0	RA0	ST	CMOS	Bi-directional I/O port
	AN0	AN	—	Analog comparator input
RA1/AN1	RA1	ST	CMOS	Bi-directional I/O port
	AN1	AN	—	Analog comparator input
RA2/AN2/VREF	RA2	ST	CMOS	Bi-directional I/O port
	AN2	AN	—	Analog comparator input
	VREF	—	AN	VREF output
RA3/AN3/CMP1	RA3	ST	CMOS	Bi-directional I/O port
	AN3	AN	—	Analog comparator input
	CMP1	—	CMOS	Comparator 1 output
RA4/T0CKI/CMP2	RA4	ST	OD	Bi-directional I/O port
	T0CKI	ST	—	Timer0 clock input
	CMP2	—	OD	Comparator 2 output
RA5/MCLR/VPP	RA5	ST	—	Input port
	$\overline{MCLR}$	ST	—	Master clear. When configured as $\overline{MCLR}$, this pin is an active low RESET to the device. Voltage on MCLR/VPP must not exceed VDD during normal device operation.
	VPP	—	—	Programming voltage input.
RA6/OSC2/CLKOUT	RA6	ST	CMOS	Bi-directional I/O port
	OSC2	—	XTAL	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
	CLKOUT	—	CMOS	In RC/INTOSC mode, OSC2 pin can output CLKOUT, which has 1/4 the frequency of OSC1
RA7/OSC1/CLKIN	RA7	ST	CMOS	Bi-directional I/O port
	OSC1	XTAL	—	Oscillator crystal input
	CLKIN	ST	—	External clock source input. RC biasing pin.
RB0/INT	RB0	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	INT	ST	—	External interrupt.
RB1/RX/DT	RB1	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	RX	ST	—	USART receive pin
	DT	ST	CMOS	Synchronous data I/O.
RB2/TX/CK	RB2	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	TX	—	CMOS	USART transmit pin
	CK	ST	CMOS	Synchronous clock I/O.
RB3/CCP1	RB3	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	CCP1	ST	CMOS	Capture/Compare/PWM I/O

Legend: O = Output — = Not used TTL = TTL Input
CMOS = CMOS Output I = Input OD = Open Drain Output
P = Power ST = Schmitt Trigger Input AN = Analog

TABLE 3-2: PIC16F627A/628A/648A PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
RB4/PGM	RB4	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	PGM	ST	—	Low voltage programming input pin. When lowvoltage programming is enabled, the interrupt-on-pin change and weak pull-up resistor are disabled.
RB5	RB5	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
RB6/T1OSO/T1CKI/PGC	RB6	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	T1OSO	—	XTAL	Timer1 oscillator output.
	T1CKI	ST	—	Timer1 clock input.
	PGC	ST	—	ICSP Programming Clock.
RB7/T1OSI/PGD	RB7	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	T1OSI	XTAL	—	Timer1 oscillator input.
	PGD	ST	CMOS	ICSP Data I/O
Vss	Vss	Power	—	Ground reference for logic and I/O pins
VDD	VDD	Power	—	Positive supply for logic and I/O pins

Legend: O = Output, — = Not used, TTL = TTL Input, CMOS = CMOS Output, I = Input, OD = Open Drain Output, P = Power, ST = Schmitt Trigger Input, AN = Analog

3.1 Clocking Scheme/Instruction Cycle

The clock input (OSC1/CLKIN/RA7 pin) is internally divided by four to generate four non-overlapping quadrature clocks namely Q1, Q2, Q3 and Q4. Internally, the program counter (PC) is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow is shown in Figure 3-2.

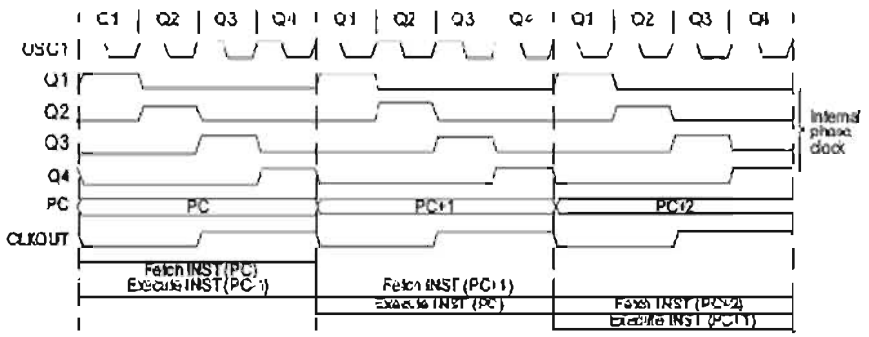
3.2 Instruction Flow/Pipelining

An instruction cycle consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO) then two cycles are required to complete the instruction (Example 3-1).

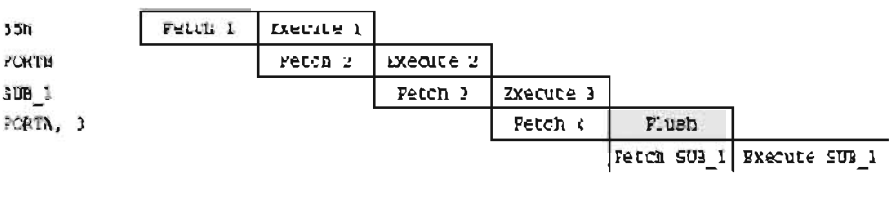
A fetch cycle begins with the program counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the Instruction Register (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3, and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

FIGURE 3-2: CLOCK/INSTRUCTION CYCLE

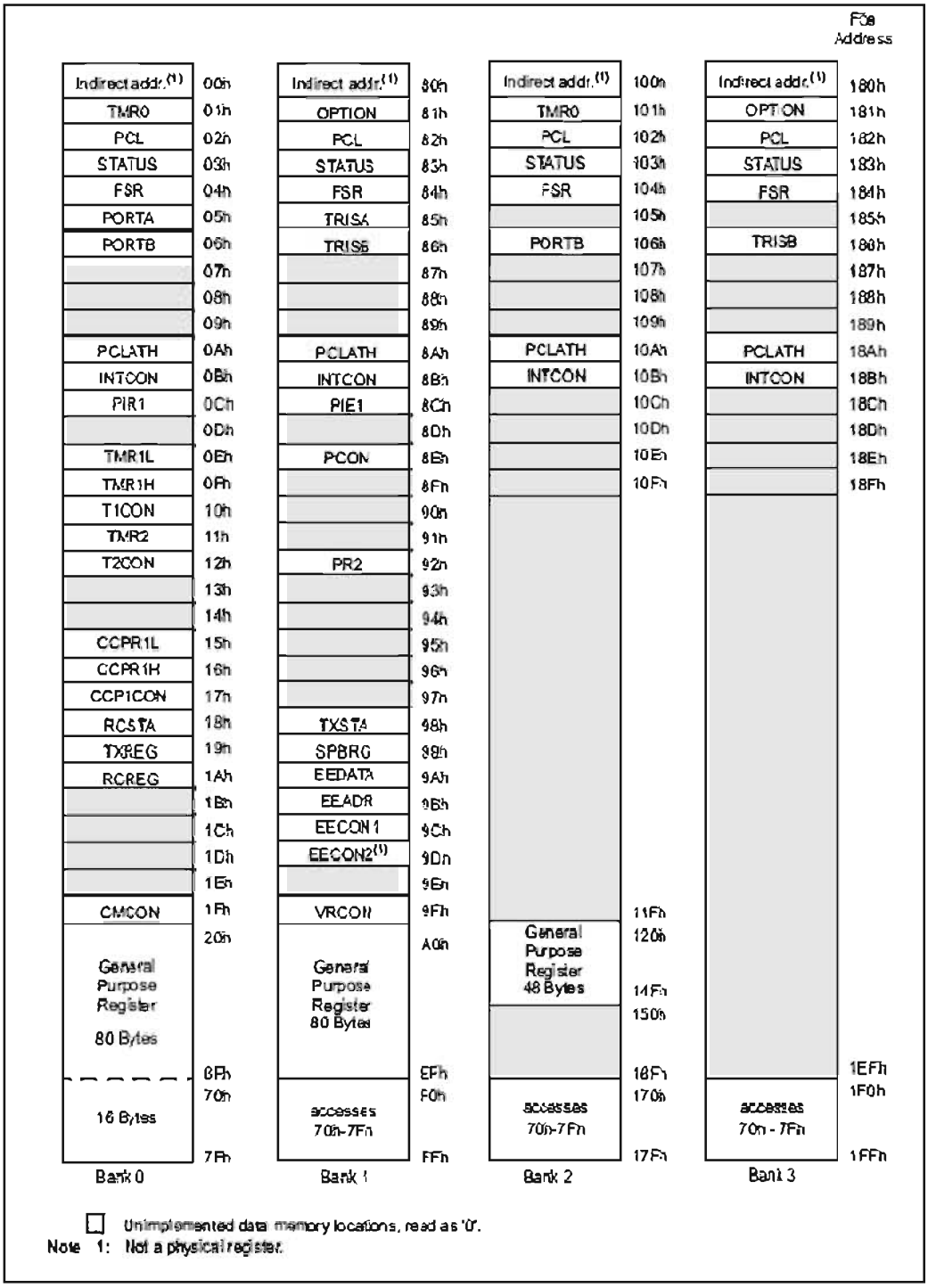


EXAMPLE 3-1: INSTRUCTION PIPELINE FLOW



All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

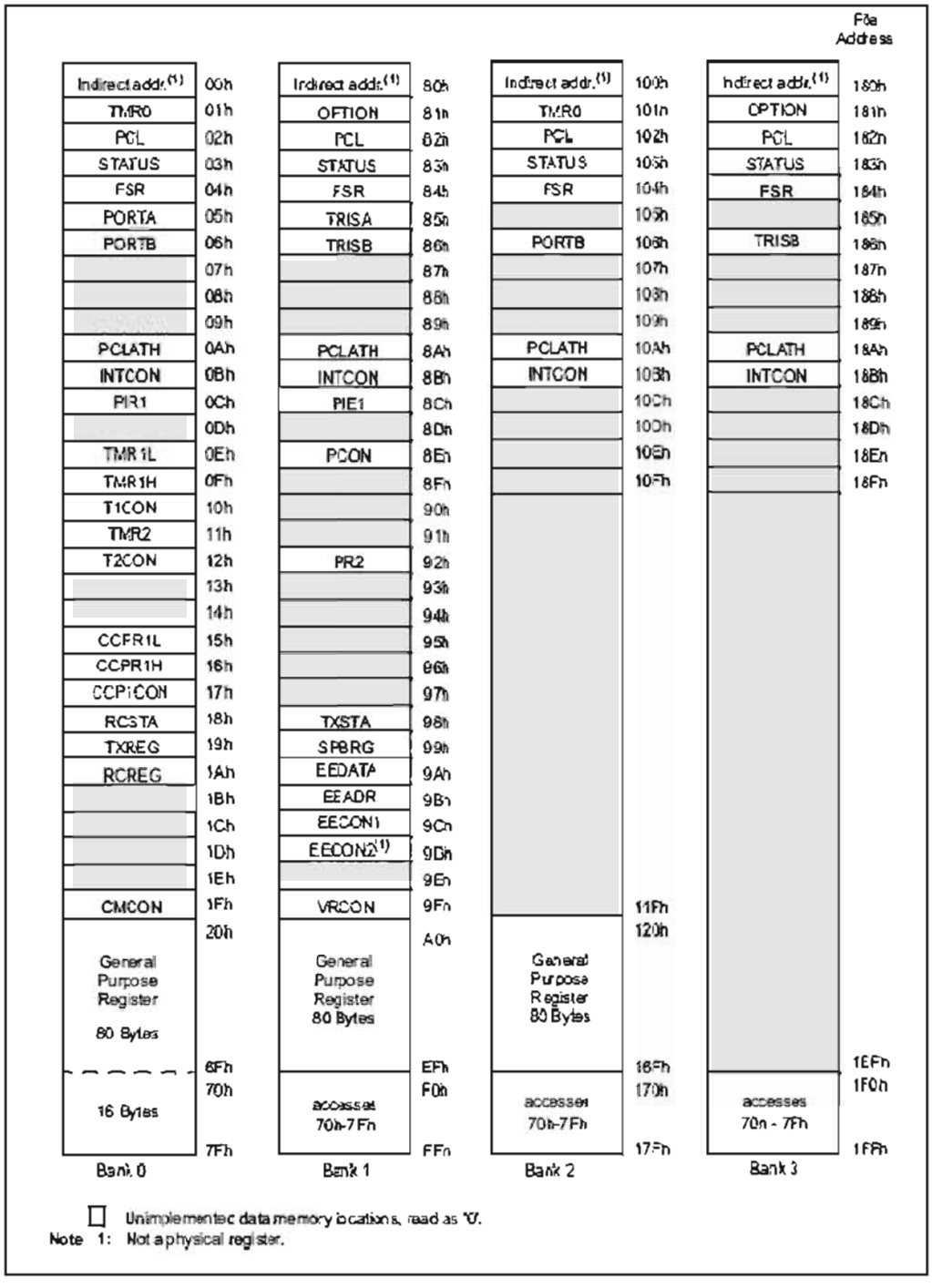
FIGURE 4-2: DATA MEMORY MAP OF THE PIC16F627A AND PIC16F628A



☐ Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

FIGURE 4-3: DATA MEMORY MAP OF THE PIC16F648A



☐ Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

4.2.2 SPECIAL FUNCTION REGISTERS

The SFRs are registers used by the CPU and Peripheral functions for controlling the desired operation of the device (Table 4-3). These registers are static RAM.

The special registers can be classified into two sets (core and peripheral). The SFRs associated with the "core" functions are described in this section. Those related to the operation of the peripheral features are described in the section of that peripheral feature.

TABLE 4-3: SPECIAL REGISTERS SUMMARY BANK0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR Reset(1)	Details on Page
Bank 0											
00h	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	28
01h	TMR0	Timer0 module's Register								xxxx xxxx	45
02h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	28
03h	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22
04h	FSR	Indirect data memory address pointer								xxxx xxxx	28
05h	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxx 0000	31
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	36
07h	—	Unimplemented								—	—
08h	—	Unimplemented								—	—
09h	—	Unimplemented								—	—
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of program counter					---0 0000	28
0Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	24
0Ch	PIR1	EEIF	CMIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF	0000 -000	26
0Dh	—	Unimplemented								—	—
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1								xxxx xxxx	48
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1								xxxx xxxx	48
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	--00 0000	48
11h	TMR2	TMR2 module's register								0000 0000	52
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	52
13h	—	Unimplemented								—	—
14h	—	Unimplemented								—	—
15h	CCPR1L	Capture/Compare/PWM register (LSB)								xxxx xxxx	55
16h	CCPR1H	Capture/Compare/PWM register (MSB)								xxxx xxxx	55
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	55
18h	RCSTA	SPEN	RX9	SREN	CREN	ADEN	FERR	OERR	RX9D	0000 000x	69
19h	TXREG	USART Transmit data register								0000 0000	78
1Ah	RCREG	USART Receive data register								0000 0000	79
1Bh	—	Unimplemented								—	—
1Ch	—	Unimplemented								—	—
1Dh	—	Unimplemented								—	—
1Eh	—	Unimplemented								—	—
1Fh	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0000	61

Legend: — = Unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: For the Initialization Condition for Registers Tables, refer to Table 14-6 and Table 14-7.

TABLE 4-4: SPECIAL FUNCTION REGISTERS SUMMARY BANK1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR Reset(1)	Details on Page
Bank 1											
80h	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	28
81h	OPTION	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23
82h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	28
83h	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22
84h	FSR	Indirect data memory address pointer								xxxx xxxx	28
85h	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 1111	31
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	36
87h	—	Unimplemented								—	—
88h	—	Unimplemented								—	—
89h	—	Unimplemented								—	—
8Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of program counter					---0 0000	28
8Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	24
8Ch	PIE1	EEIE	CMIE	RCIE	TXIE	—	CCP1IE	TMR2IE	TMR1IE	0000 -000	25
8Dh	—	Unimplemented								—	—
8Eh	PCON	—	—	—	—	OSCF	—	$\overline{POR}$	$\overline{BOR}$	---- 1-0x	27
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	—	Unimplemented								—	—
92h	PR2	Timer2 Period Register								1111 1111	52
93h	—	Unimplemented								—	—
94h	—	Unimplemented								—	—
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	71
99h	SPBRG	Baud Rate Generator Register								0000 0000	71
9Ah	EEDATA	EEPROM data register								xxxx xxxx	89
9Bh	EEADR	EEPROM address register								xxxx xxxx	90
9Ch	EECON1	—	—	—	—	WRERR	WREN	WR	RD	---- x000	90
9Dh	EECON2	EEPROM control register 2 (not a physical register)								---- ----	90
9Eh	—	Unimplemented								—	—
9Fh	VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0	000- 0000	67

Legend: — = Unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: For the Initialization Condition for Registers Tables, refer to Table 14-6 and Table 14-7.

TABLE 4-5: SPECIAL FUNCTION REGISTERS SUMMARY BANK2

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR Reset(1)	Details on Page
Bank 2											
100h	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	28
101h	TMR0	Timer0 module's Register								xxxx xxxx	46
102h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	28
103h	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22
104h	FSR	Indirect data memory address pointer								xxxx xxxx	28
105h	—	Unimplemented								—	—
106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	36
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of program counter					---0 0000	28
10Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	24
10Ch	—	Unimplemented								—	—
10Dh	—	Unimplemented								—	—
10Eh	—	Unimplemented								—	—
10Fh	—	Unimplemented								—	—
110h	—	Unimplemented								—	—
111h	—	Unimplemented								—	—
112h	—	Unimplemented								—	—
113h	—	Unimplemented								—	—
114h	—	Unimplemented								—	—
115h	—	Unimplemented								—	—
116h	—	Unimplemented								—	—
117h	—	Unimplemented								—	—
118h	—	Unimplemented								—	—
119h	—	Unimplemented								—	—
11Ah	—	Unimplemented								—	—
11Bh	—	Unimplemented								—	—
11Ch	—	Unimplemented								—	—
11Dh	—	Unimplemented								—	—
11Eh	—	Unimplemented								—	—
11Fh	—	Unimplemented								—	—

Legend: — = Unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented.

Note 1: For the Initialization Condition for Registers Tables, refer to Table 14-6 and Table 14-7.

TABLE 4-6: SPECIAL FUNCTION REGISTERS SUMMARY BANK3

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR Reset(1)	Details on Page
Bank 3											
180h	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	28
181h	OPTION	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23
182h	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	28
183h	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22
184h	FSR	Indirect data memory address pointer								xxxx xxxx	28
185h	—	Unimplemented								—	—
186h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	36
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of program counter					---0 0000	28
18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	24
18Ch	—	Unimplemented								—	—
18Dh	—	Unimplemented								—	—
18Eh	—	Unimplemented								—	—
18Fh	—	Unimplemented								—	—
190h	—	Unimplemented								—	—
191h	—	Unimplemented								—	—
192h	—	Unimplemented								—	—
193h	—	Unimplemented								—	—
194h	—	Unimplemented								—	—
195h	—	Unimplemented								—	—
196h	—	Unimplemented								—	—
197h	—	Unimplemented								—	—
198h	—	Unimplemented								—	—
199h	—	Unimplemented								—	—
19Ah	—	Unimplemented								—	—
19Bh	—	Unimplemented								—	—
19Ch	—	Unimplemented								—	—
19Dh	—	Unimplemented								—	—
19Eh	—	Unimplemented								—	—
19Fh	—	Unimplemented								—	—

Legend: — = Unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: For the Initialization Condition for Registers Tables, refer to Table 14-6 and Table 14-7.

4.2.2.1 STATUS Register

The STATUS register, shown in Register 4-1, contains the arithmetic status of the ALU, the RESET status and the bank select bits for data memory (SRAM).

The STATUS register can be the destination for any instruction, like 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 non-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 "000uuluu" (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 any STATUS bit. For other instructions not affecting any STATUS bits, see the "Instruction Set Summary".

Note 1: The C and DC bits operate as a $\overline{Borrow}$ and Digit $\overline{Borrow}$ out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

REGISTER 4-1: 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)
00 = Bank 0 (00h - 7Fh)
01 = Bank 1 (80h - FFh)
10 = Bank 2 (100h - 17Fh)
11 = Bank 3 (180h - 1FFh)

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

4.2.2.2 OPTION Register

The OPTION register is a readable and writable register which contains various control bits to configure the TMR0/WDT prescaler, the external RB0/INT interrupt, TMR0 and the weak pull-ups on PORTB.

Note:	To achieve a 1:1 prescaler assignment for TMR0, assign the prescaler to the WDT (PSA = 1). See Section 6.3.1.

REGISTER 4-2: OPTION 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

4.2.2.3 INTCON Register

The INTCON register is a readable and writable register, which contains the various enable and flag bits for all interrupt sources except the comparator module. See Section 4.2.2.4 and Section 4.2.2.5 for a description of the comparator enable and flag bits.

Note:	Interrupt flag bits get set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>).

REGISTER 4-3: 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 un-masked interrupts
0 = Disables all interrupts

bit 6 **PEIE:** Peripheral Interrupt Enable bit
1 = Enables all un-masked 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 = When 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

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

4.2.2.4 PIE1 Register

This register contains interrupt enable bits.

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

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
EEIE	CMIE	RCIE	TXIE	—	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **EEIE**: EE Write Complete Interrupt Enable Bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt

bit 6 **CMIE**: Comparator Interrupt Enable bit
1 = Enables the comparator interrupt
0 = Disables the comparator 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 **Unimplemented**: Read as '0'

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

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

4.2.2.5 PIR1 Register

This register contains interrupt flag bits.

Note:	Interrupt flag bits get 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 4-5: PIR1 REGISTER (ADDRESS: 0Ch)

R/W-0	R/W-0	R-0	R-0	U-0	R/W-0	R/W-0	R/W-0
EEIF	CMIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **EEIF**: EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation has not completed or has not been started

bit 6 **CMIF**: Comparator Interrupt Flag bit
1 = Comparator output has changed
0 = Comparator output has not changed

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 **Unimplemented**: Read as '0'

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

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

4.2.2.6 PCON Register

The PCON register contains flag bits to differentiate between a Power-on Reset, an external MCLR Reset, WDT Reset or a Brown-out Reset.

Note:	$\overline{BOR}$ is unknown on Power-on Reset. It must then be set by the user and checked on subsequent RESETS to see if $\overline{BOR}$ is cleared, indicating a brown-out has occurred. The $\overline{BOR}$ STATUS bit is a "don't care" and is not necessarily predictable if the brown-out circuit is disabled (by clearing the BOREN bit in the Configuration word).

REGISTER 4-6: PCON REGISTER (ADDRESS: 8Eh)

U-0	U-0	U-0	U-0	R/W-1	U-0	R/W-0	R/W-x
—	—	—	—	OSCF	—	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **OSCF:** INTOSC oscillator frequency
1 = 4 MHz typical
0 = 37 kHz typical

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

bit 1 **$\overline{POR}$:** Power-on Reset STATUS bit
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{BOR}$:** Brown-out Reset STATUS bit
1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

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

5.0 I/O PORTS

The PIC16F627A/628A/648A have two ports, PORTA and PORTB. Some pins for these I/O ports are multiplexed with alternate functions 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.

5.1 PORTA and TRISA Registers

PORTA is an 8-bit wide latch. RA4 is a Schmitt Trigger input and an open drain output. Port RA4 is multiplexed with the T0CKI clock input. RA5(1) is a Schmitt Trigger input only and has no output drivers. All other RA port pins have Schmitt Trigger input levels and full CMOS output drivers. All pins have data direction bits (TRIS registers) which can configure these pins as input or output.

A '1' in the TRISA register puts the corresponding output driver in a High-impedance mode. A '0' in the TRISA register puts the contents of the output latch on the selected pin(s).

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. So a write to a port implies that the port pins are first read, then this value is modified and written to the port data latch.

The PORTA pins are multiplexed with comparator and voltage reference functions. The operation of these pins are selected by control bits in the CMCON (comparator control register) register and the VRCON (voltage reference control register) register. When selected as a comparator input, these pins will read as '0's.

Note 1: RA5 shares function with V_{PP}. When V_{PP} voltage levels are applied to RA5, the device will enter Programming mode.

2: On RESET, the TRISA register is set to all inputs. The digital inputs (RA<3:0>) are disabled and the comparator inputs are forced to ground to reduce current consumption.

3: TRISA<6:7> is overridden by oscillator configuration. When PORTA<6:7> is [illegible] TRISA<6:7> bits are ignored.

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

The RA2 pin will also function as the output for the voltage reference. When in this mode, the VREF pin is a very high-impedance output. The user must configure TRISA<2> bit as an input and use high-impedance loads.

In one of the Comparator modes defined by the CMCON register, pins RA3 and RA4 become outputs of the comparators. The TRISA<4:3> bits must be cleared to enable outputs to use this function.

EXAMPLE 5-1: Initializing PORTA

```
CLRF    PORTA       ;Initialize PORTA by
                    ;setting
                    ;output data latches
MOVLW   0x07        ;Turn comparators off and
MOVWF   CMCON       ;enable pins for I/O
                    ;functions
BCF     STATUS, RP1
BSF     STATUS, RP0;Select Bank1
MOVLW   0x1F        ;Value used to initialize
                    ;data direction
MOVWF   TRISA       ;Set RA<4:0> as inputs
                    ;TRISA<5> always
                    ;read as '1'.
                    ;TRISA<7:6>
                    ;depend on oscillator
                    ;mode
```

FIGURE 5-1: BLOCK DIAGRAM OF RA0/AN0:RA1/AN1 PINS

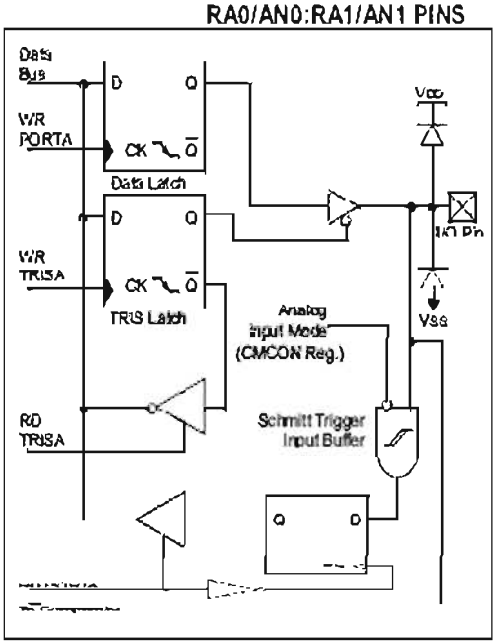


TABLE 5-1: PORTA FUNCTIONS

Name	Function	Input Type	Output Type	Description
RA0/AN0	RA0	ST	CMOS	Bi-directional I/O port
	AN0	AN	—	Analog comparator input
RA1/AN1	RA1	ST	CMOS	Bi-directional I/O port
	AN1	AN	—	Analog comparator input
RA2/AN2/VREF	RA2	ST	CMOS	Bi-directional I/O port
	AN2	AN	—	Analog comparator input
	VREF	—	AN	VREF output
RA3/AN3/CMP1	RA3	ST	CMOS	Bi-directional I/O port
	AN3	AN	—	Analog comparator input
	CMP1	—	CMOS	Comparator 1 output
RA4/T0CKI/CMP2	RA4	ST	OD	Bi-directional I/O port. Output is open drain type.
	T0CKI	ST	—	External clock input for TMR0 or comparator output
	CMP2	—	OD	Comparator 2 output
RA5/$\overline{MCLR}$/VPP	RA5	ST	-	Input port
	$\overline{MCLR}$	ST	-	Master clear. When configured as $\overline{MCLR}$, this pin is an active low RESET to the device. Voltage on $\overline{MCLR}$/VPP must not exceed VDD during normal device operation.
	VPP	HV	—	Programming voltage input.
RA6/OSC2/CLKOUT	RA6	ST	CMOS	Bi-directional I/O port
	OSC2	—	XTAL	Oscillator crystal output. Connects to crystal resonator in Crystal Oscillator mode.
	CLKOUT	—	CMOS	In RC or INTOSC mode, OSC2 pin can output CLKOUT, which has 1/4 the frequency of OSC1
RA7/OSC1/CLKIN	RA7	ST	CMOS	Bi-directional I/O port
	OSC1	XTAL	—	Oscillator crystal input. Connects to crystal resonator in Crystal Oscillator mode.
	CLKIN	ST	—	External clock source input. RC biasing pin.

Legend: O = Output, — = Not used, TTL = TTL Input, CMOS = CMOS Output, I = Input, OD = Open Drain Output, P = Power, ST = Schmitt Trigger Input, AN = Analog

TABLE 5-3: PORTB FUNCTIONS

Name	Function	Input Type	Output Type	Description
RB0/INT	RB0	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	INT	ST	—	External interrupt.
RB1/RX/DT	RB1	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	RX	ST	—	USART Receive Pin
	DT	ST	CMOS	Synchronous data I/O
RB2/TX/CK	RB2	TTL	CMOS	Bi-directional I/O port
	TX	—	CMOS	USART Transmit Pin
	CK	ST	CMOS	Synchronous Clock I/O. Can be software programmed for internal weak pull-up.
RB3/CCP1	RB3	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	CCP1	ST	CMOS	Capture/Compare/PWM I/O
RB4/PGM	RB4	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	PGM	ST	—	Low voltage programming input pin. When low voltage programming is enabled, the interrupt-on-pin change and weak pull-up resistor are disabled.
RB5	RB5	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
RB6/T1OSO/T1CKI/PGC	RB6	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	T1OSO	—	XTAL	Timer1 Oscillator Output
	T1CKI	ST	—	Timer1 Clock Input
	PGC	ST	—	ICSP Programming Clock
RB7/T1OSI/PGD	RB7	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	T1OSI	XTAL	—	Timer1 Oscillator Input
	PGD	ST	CMOS	ICSP Data I/O

Legend: O = Output, — = Not used, TTL = TTL Input, CMOS = CMOS Output, I = Input, OD = Open Drain Output, P = Power, ST = Schmitt Trigger Input, AN = Analog

TABLE 5-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB(1)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on All Other RESETS
06h, 106h	PORTB	RB7	RB6	RB5	RB4(2)	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
81h, 181h	OPTION	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: u = unchanged, x = unknown

Note 1: Shaded bits are not used by PORTB.
2: LVP Configuration Bit sets RB4 functionality.

UNIVERSITAS MAJOR PACENSIS DIVI ANDRE