ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMATICA Y ELECTRÓNICA

ESCUELA DE INGENIERIA ELECTRONICA

"CONECTIVIDAD INALÁMBRICA ENTRE UN ROBOT Y UN TELÉFONO CELULAR UTILIZANDO SISTEMAS EMBEBIDOS"

TESIS DE GRADO

Previa la obtención del título de

INGENIERA EN ELECTRONICA Y COMPUTACIÓN

Presentado por:

LUISA FERNANDA SANCHEZ OJEDA

RIOBAMBA – ECUADOR

2010

A mis padres que con su esfuerzo han logrado entregarme el apoyo necesario para cumplir mis metas. A mi hija y mi esposo quienes han sido mi fuente de superación.

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

“Yo Luisa Fernanda Sánchez Ojeda soy responsable de las ideas, doctrinas y resultados expuestos en esta tesis; y, el patrimonio intelectual de la Tesis de Grado pertenece a la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO”.

Nombre y firma del autor

ÍNDICE 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.
HSEROUT	Salida serial de hardware asíncrono.
J2ME	Java 2 Platform, Micro Edition
MIDP	Mobile Information Device Profile
PAN	red de área personal
PIC	Peripherical 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 Específico
SPP	Perfil puerto serie
UCP	Unidad Central de Proceso.
USART	Transmisor/Receptor Asíncrono Universal.
UART	adaptador de comunicación serie asíncrona
UUID	Identificador Único Universal Dispositivo

ÍNDICE GENERAL

CAPÍTULO V

CAPÍTULO VI

CAPÍTULO VII

CONCLUSIONES.

RECOMENDACIONES

RESUMEN

SUMMARY

ANEXOS

BIBLIOGRAFIA

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

INTRODUCCION

En la actualidad podemos encontrar algunas aplicaciones desarrolladas para celulares por ejemplo manejos de dispositivos electrónicos, manejo de sistemas como base de datos donde se puede acceder de forma local o remota por lo que es conveniente implementar aplicaciones que permitan realizar tareas difíciles para los seres humanos por su alto nivel de peligrosidad.

En nuestro país el tema de los Microcontroladores está dando buenos resultados y hay algunas aplicaciones desarrolladas. Están presentes en nuestro trabajo, en nuestra casa y nuestra vida diaria. Se pueden encontrar controlando el funcionamiento de los ratones y teclados de los computadores, en los teléfonos, en los hornos microondas y los televisores de nuestro hogar

Por otro lado la comunicación inalámbrica está llegando de a poco en lo que son las redes 802.11b y 802.11g. 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.

En cuanto a la tecnología de los celulares, tiene una vida de más o menos dos décadas en nuestro país y ha 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.

Gracias a toda la información y referencias de investigaciones, la hipótesis que rige esta tesis asevera que la implementación de este sistema servirá para la automatización de un robot utilizando conectividad inalámbrica entre el equipo central que ejecutara el sistema embebido y el teléfono celular que controlará al robot a través de órdenes que facilitará la realización de tareas de control.

CAPÍTULO I

GENERALIDADES

1.1 Antecedentes

El simple hecho de ser seres humanos nos hace desenvolvernos en medios donde tenemos que estar comunicados. Por eso la gran importancia de la transmisión y la recepción de información, y en la época actual donde los computadores hacen parte de la cotidianidad, es necesario establecer medios de comunicación eficaces entre ellos.

Una de las tecnologías 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 pañales y se deben de 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.

La tecnología bluetooth es extremadamente útil y confiable, la comunicación entre dispositivos móviles y PCS ya no es la misma después de la incorporación de el Standard bluetooth, lo que hace muy atractivo para los desarrolladores en aplicaciones para dispositivos móviles ya que integra la conectividad local a otro nivel.

La denominación de Sistemas embebidos refleja que son una 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 a quien lo mira.

1.2 Justificación

Debido al crecimiento de las aplicaciones en los equipos de telefonía celular en la actualidad, esta tesis será desarrollada para aprovechar la capacidad de un celular creando una interfaz capaz de controlar un robot mediante la ejecución de un sistema embebido lo que nos da un amplio campo para su aplicación, con la implementación de nuestro sistema podemos 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 equipo que actué como centro de control de datos, esto hace posible

que podamos enviar y recibir datos en cualquier dirección y así poder utilizarlo en diferentes situaciones.

1.3 Objetivos

1.3.1 Objetivo General

- Conectar de forma inalámbrica un robot y un teléfono celular a través de un sistema embebido.

1.3.2 Objetivos Específicos

- Diseñar e implementar la interfaz en el teléfono celular.
- Programar el sistema embebido en el equipo central.
- Diseñar e implementar la interfaz entre el equipo central y el robot.
- Programar el sistema embebido para el control del robot.

CAPÍTULO II

MICROCONTROLADOR PIC16F628A

2.1 MICROCONTROLADOR PIC16F628A

El pic 16f628A 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.

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 MHz y 20 MHz.

Los PIC16F62X 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, 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:

Figura II.1. Microcontrolador PIC16F628A.

- Conjunto reducido de instrucciones (RISC). Sólamente 35 instrucciones que aprender a utilizar
- Oscilador interno de 4MHz
- Las instrucciones se ejecutan en un sólo 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, si se 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 20 MHz (ciclo de máquina de 200 ns)
- Memoria de programa: 2048 locaciones de 14 bits
- Memoria de datos: Memoria RAM de 224 bytes (8 bits por registro)
- 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 Aspecto externo

EL PIC16(F)628A 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:

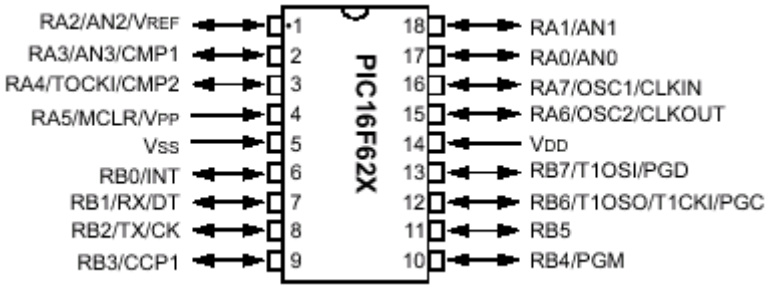


Figura II.2. Distribución de pines PIC16F628A.

PORTA: RA0-RA7:

- Los pines RA0-RA4 y RA6–RA7 son bidireccionales y manejan señales TTL
- El pin RA5 es una entrada Schmitt Trigger que sirve también para entrar en el modo de programación cuando se aplica una tensión igual a Vpp (13,4V mínimo)
- El terminal RA4 puede configurarse como reloj de entrada para el contador TMR0
- Los pines RA0-RA3 sirven de entrada para el comparador analógico

PORTB: RB0-RB7:

- Los pines RB0-RB7 son bidireccionales y manejan señales TTL
- Por software se pueden activar las resistencias de pull-up internas, que evitan el uso de resistencias externas en caso de que los terminales se utilicen como entrada (permite, en algunos casos, reducir el número de componentes externos)
- El pin RB0 se puede utilizar como entrada de pulsos para provocar una interrupción externa
- Los pines RB4-RB7 están diseñados para detectar una interrupción por cambio de estado.

  Esta interrupción puede utilizarse para controlar un teclado matricial, por poner un ejemplo

Otros pines

- VDD: Pin de alimentación positiva. De 2 a 5,5 Vcc
- VSS: Pin de alimentación negativa. Se conecta a tierra o a 0 Vcc
- MCLR: Master Clear (Reset). Si el nivel lógico de este terminal es bajo (0 Vcc), el microcontrolador permanece inactivo. Este reset se controla mediante la palabra de configuración del PIC
- OSC1/CLKIN: Entrada de oscilador externo
- OSC2/CLKOUT: Salida del oscilador. El PIC 16F628 dependiendo de cómo se configure puede proporcionar una salida de reloj por medio de este pin

2.3 Arquitectura Interna

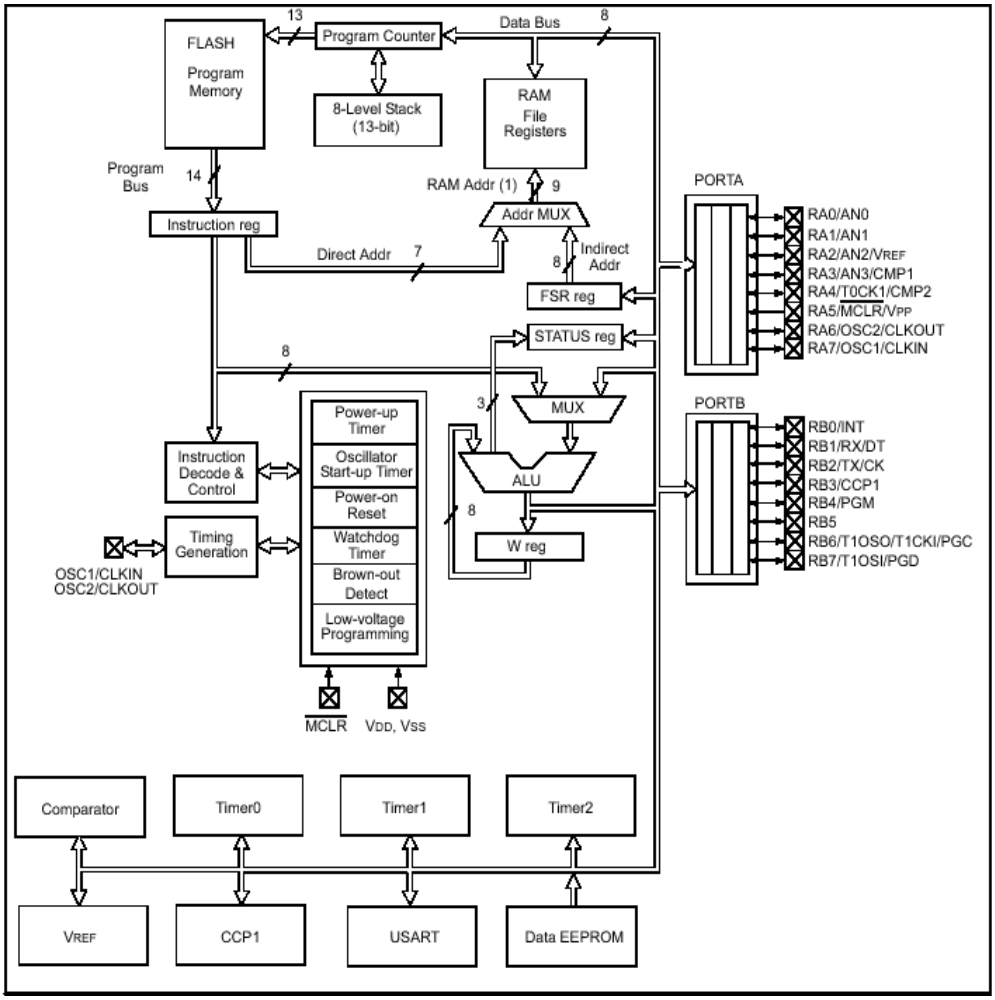


Figura II.3 Arquitectura interna PIC16F628A.

2.4 Organización de Memoria

El PIC16F628 posee un contador 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 PIC16F84 tienen la misma organización, excepto que únicamente están implementados los primeros 1Kx14, desde 0000h hasta 03FFh.

El PIC16F628 y el PIC16F627 poseen un espacio de memoria RAM de datos de 512x8, dividido en 4 bancos de 128 bytes cada uno, figura 3. Sin embargo, sólo están implementados 330 bytes, correspondiendo 224 al area 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 y 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.

CAPÍTULO III

COMUNICACIÓN INALÁMBRICA

3.1 Introducción

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.

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.

Sin embargo se pueden mezclar las redes cableadas y las inalámbricas, y de esta manera generar una "Red Híbrida" y poder resolver los últimos metros hacia la estación. Se puede considerar que el sistema cableado sea la 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.

3.2 Comunicación de Radiofrecuencia

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 transmiten 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.

Figura III.1. Transmisor y receptor de radio.

3.2.1 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.

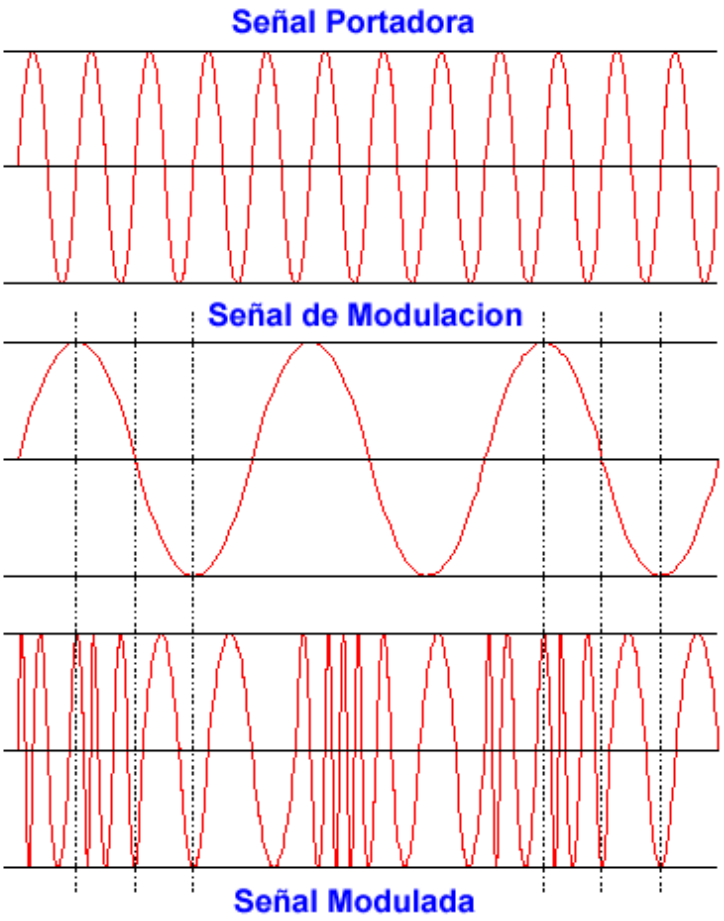


Figura III.2. Ejemplo de señal Modulada.

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 determina la rapidez o lentitud con que ocurren las variaciones en la frecuencia. Es decir, que una frecuencia mas baja de modulación provoca que ocurran variaciones a una tasa más lenta, y una frecuencia mas alta de 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.

3.3 Bluetooth

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

Estos dispositivos pueden ser computadoras de escritorio, PDAs, teléfonos móviles y en fin, las posibilidades pueden considerarse infinitas.

Bluetooth opera en la banda libre de radio ISM1 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 un ambiente libre de obstáculos.

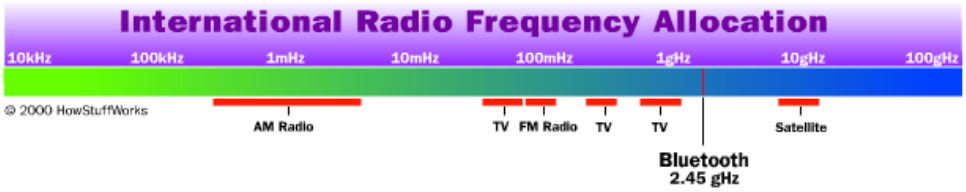


Figura III.3 Relación de Frecuencia Bluetooth

3.3.1 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. Éstas 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 ésta tecnología puede ser integrada en equipos de baja potencia y bajo coste. Éste 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.

3.3.2 El canal

Bluetooth utiliza un sistema FH/TDD (salto de frecuencia/división de tiempo duplex), 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 (pequeña red que establecen automáticamente los terminales Bluetooth para comunicarse entre si), donde una unidad actúa como maestra, controlando el tráfico de datos en la piconet que se genera entre las demás unidades, donde éstas actúan como esclavas, enviando y recibiendo señales hacia el maestro.

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.

3.3.3 Piconets:

Como hemos citado anteriormente si un equipo se encuentra dentro del radio de cobertura de otro, éstos pueden establecer conexión entre ellos. Cada dispositivo tiene una dirección única de 48 bits, basada en el estándar IEEE 802.11 para WLAN. En principio sólo 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.

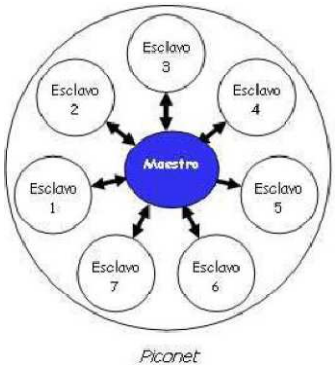


Figura III.4 Diagrama red Bluetooth

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 éste 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 sólo 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.

3.3.4 Alcance

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

3.3.4.1 Máxima salida de poder

1. Clase 1 a 100mW y 100 Metros aprox.
2. Clase 2 a 2.5mW y 10 Metros aprox.
3. Clase 3 a 1mW y 1 Metro aprox.

3.3.5 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, etc.

3.3.5.1 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 III.5. Adaptador Bluetooth para PC o Laptops.

3.3.5.2 Conversor Bluetooth a serial

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

3.3.5.2.1 Caracteristicas

- Bluetooth Spec v2.0+EDR Compliant
- Salida de poder de tipo clase 2
- Operación de bluetooth a máxima velocidad
- Operación a 3.3V
- Interface UART (adaptador de comunicación serie asíncrona.)
- Componentes externos mínimos

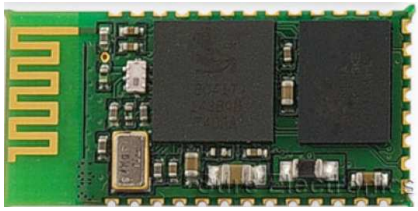


Figura III.6 .Modulo Conversor Bluetooth serial

El modulo bluetooth puede comunicarse con cualquier unidad Master como Laptop, PDA, computadoras personales y celulares.

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

Figura III.7 Comunicación Modulo Conversor Bluetooth serial

Dos módulos bluetooth son siempre unidades eslavas, por lo tanto no se pueden comunicar entre si.

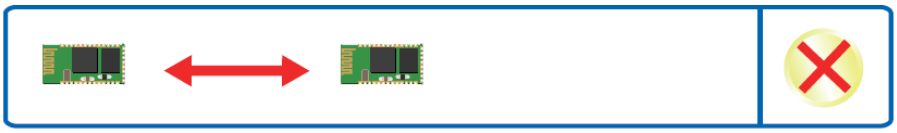


Figura III.8 Comunicación entre módulos bluetooth

3.3.5.2.2 Diagrama de bloques

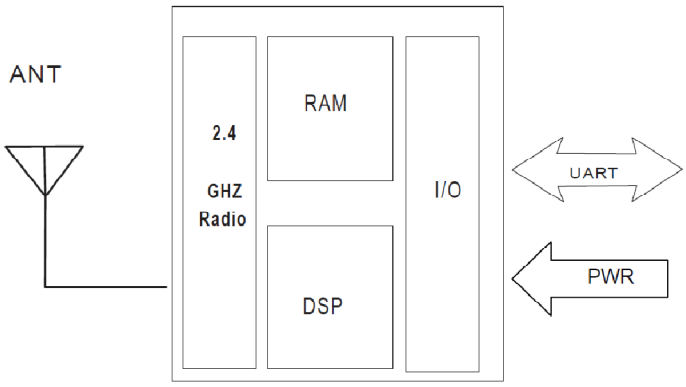


Figura III.9 Diagrama Modulo Conversor Bluetooth serial

CAPÍTULO IV

JAVA 2 MICRO EDITION: SOPORTE BLUETOOTH

4.1 Introducción

Bluetooth es, pues, 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 para 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

4.2 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. Ésta 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.

El JSR-82[1] especifica un API de alto nivel para la programación de dispositivos Bluetooth. Depende de la configuración CLDC de J2ME, y se divide en dos paquetes: javax.bluetooth y javax.obex. El primer paquete provee la funcionalidad para la realización de búsquedas de dispositivos, búsquedas de servicios y comunicación mediante flujos de datos (streams) o arrays de bytes. Por otro lado el paquete javax.obex permite la comunicación mediante el protocolo OBEX.

El objetivo de ésta especificación era definir un API estándar abierto, no propietario que pudiera ser usado en todos los dispositivos que implementen J2ME. Por consiguiente fue diseñado usando los APIs J2ME y el entorno de trabajo CLDC/MIDP.

Los APIs JSR 82 son muy flexibles, ya que permiten trabajar tanto con aplicaciones nativas Bluetooth como con aplicaciones Java Bluetooth.

El API intenta ofrecer las siguientes capacidades:

• Registro de servicios.

• Descubrimiento de dispositivos y servicios.

• Establecer conexiones RFCOMM, L2CAP y OBEX entre dispositivos.

• Usar dichas conexiones para mandar y recibir datos (las comunicaciones de voz no están soportadas).

• Manejar y controlar las conexiones de comunicación.

• Ofrecer seguridad a dichas actividades.

4.3 Perfil del puerto serie (SPP)

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 más 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.

Una aplicación que ofrezca un servicio basado en el perfil de puerto serie (SPP) es un servidor SPP. Una aplicación que inicie una conexión a un servicio SPP es un cliente SPP. Cliente y servidor residen en los extremos de una sesión RFCOMM. El servidor SPP registra su servicio en el SDDB, y como parte del proceso de registro, se añade un identificador de canal (channel identifier) al ServiceRecord por la implementación.

4.3.1 Registro del servicio del 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.microedition.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, éste devuelve un StreamConnectionNotifier que representa 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. Crear un identificador de un canal servidor RFCOMM, chanN y asignarlo.

2. chanN es añadido al ProtocolDescritorList en el registro de servicio.

3. El UUID (102030...) usado en el connection string para describir el tipo de servicio ofrecido es añadido al ServiceClassIDList.

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.

4.3.2 Establecimiento de la conexión

Antes de que un cliente SPP pueda establecer una conexión con un servicio SPP, éste 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.

```
StreamConnection con =
(StreamConnection) Connector.open(“btspp://dirección:identificador_canal”);
```

CAPÍTULO V

VISUAL BASIC 6.0

5.1 Introducción

Visual Basic 6.0 es un lenguaje de programación visual, también llamado lenguaje de 4ª generación. Esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla.

En Visual Basic 6.0, se le llama objeto a todo lo que se ve en una ventana típica de Windows; los objetos son por ejemplo un botón de comando, una caja de texto, una imagen, en general todo objeto visible que puedas ver en la pantalla,se les llaman objetos porque cada uno de ellos poseen propiedades, eventos y métodos. Un botón de comando tiene propiedades tales como: Caption (Titulo) que indica el texto que tiene el botón, también tiene las propiedades Width (Anchura) y Height (Altura) que establecen la anchura y altura del botón.

5.2 Modo de diseño y modo de ejecución

La aplicación Visual Basic de Microsoft puede trabajar de dos modos distintos: En modo diseño y en modo de ejecución. En modo diseño el usuario construye interactivamente la aplicación, colocando controles en el formulario, definiendo sus propiedades, y desarrollando funciones para gestionar los eventos.

La aplicación se prueba en modo de ejecución. En este caso el usuario actúa sobre el programa (introduce eventos) y prueba cómo responde el programa. Hay algunas propiedades de los controles que deben establecerse en modo de diseño, pero muchas otras pueden cambiarse en tiempo de ejecución desde el programa escrito en Visual Basic 6.0.

5.3 El entorno de Visual Basic 6.0

Cuando se arranca Visual Basic 6.0 aparece en la pantalla una configuración similar a la mostrada en la siguiente figura:

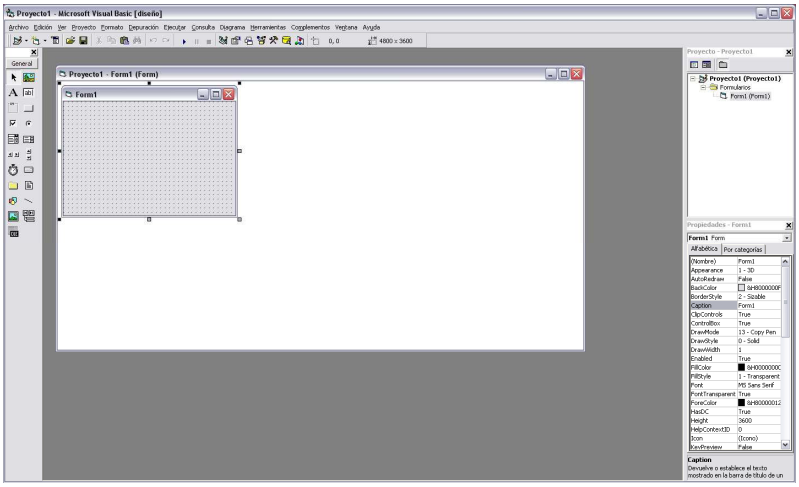


Figura V.1 Entorno de desarrollo de Visual Basic 6.0

5.3.1 La barra de herramientas no estándar (Toolbox)

Esta barra incluye los controles con los que se puede diseñar la pantalla de la aplicación. Estos controles son por ejemplo, botones, etiquetas, cajas de texto, zonas gráficas, etc.

5.3.2 Formularios (Forms)

Los formularios son las zonas de la pantalla sobre las que se diseña el programa y sobre los que se sitúan los controles o herramientas del Toolbox. Al ejecutar el programa, el Form se convertirá en la ventana de la aplicación donde aparecerán los botones, las cajas de texto, los gráficos, etc.

5.4 Creación de programas ejecutables

Una vez finalizada la programación de la nueva aplicación, la siguiente tarea suele consistir en la creación de un programa ejecutable para su distribución e instalación en cuantos ordenadores se desee, incluso aunque en ellos no este instalado Visual Basic 6.0.

Para crear un programa ejecutable se utiliza el comando Make ProjectName.exe… del menú File. De esta manera se genera un fichero cuya extensión será (.EXE). Para que este programa funcione en un ordenador solamente se necesita que el fichero MSVBVM60.DLL esté instalado en el directorio de C:\Windows\System o C:\WinNT\System32.

CAPÍTULO VI

DISEÑO DEL SISTEMA

6.1 Requerimientos del sistema

La temática del proyecto 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 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 433 Mhz. un PC, con dispositivos Bluetooth y equipos celulares.

6.2 Esquema del sistema

Este diagrama muestra el flujo de datos entre los diferentes dispositivos que participan en este sistema.

Figura VI.1 Diagrama de flujo de información.

El equipo central de control (PC) se conecta con un equipo de telefonía celular 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 usó 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 equipo celular, se utilizó una programación con soporte para JavaMicroEdition, más conocido como JME2, que permite establecer un enlace de comunicación de datos entre el celular y el PC emulando un puerto serial

Las interfaces que participan en este sistema son el puerto serial del PC, que tiene conectado un transmisor inalámbrico integrado en un control remoto que se encarga de enviar el dato inalámbricamente hasta el receptor que está en el carro

El software que se ejecuta en el equipo central está desarrollado en Visual Basic, el software es el encargado de la comunicación entre todos los dispositivos.

Finalmente podemos controlar el robot usando el equipo de telefonía celular Este es un robot con funciones básicas, hablamos de movimientos básicos como son los movimientos similares a los de un auto de juguete, ya que la idea central del proyecto es probar la conectividad entre distintos dispositivos,

6.2.1 Etapa Equipo Celular

Para esta etapa se utilizó un equipo de telefonía celular con soporte bluetooth en el cual se programará en el lenguaje JAVA 2 Microedition para establecer la comunicación entre el celular y el modulo bluetooth y poder enviar los datos de direccionamiento al vehículo (robot).

6.2.1.1 Software.- Diseño del Programa Celular

El programa está diseñado para establecer la comunicación entre el equipo celular y el modulo bluetooth que se encuentra en la placa del circuito, para ello 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;
Client 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 un puerto serial.

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

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

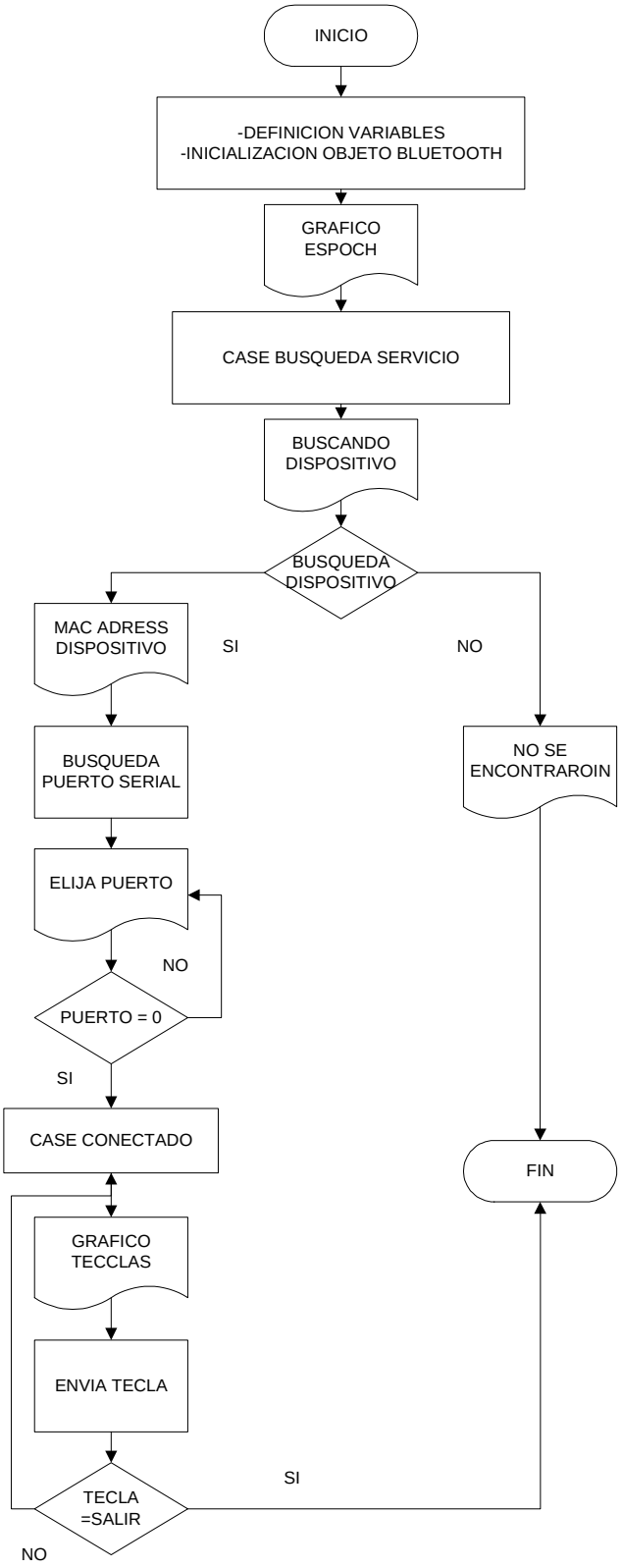


Figura VI.2 Diagrama de Flujo del celular

Los APIs Java para Bluetooth definen dos paquetes :

- javax.bluetooth
- javax.obex

En este caso se utilizó javax.bluetooth que nos permitió la 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

6.2.1.2 Hardware.- Diseño Electrónico

Se utilizo un teléfono celular NOKIA 5200 el cual envía los datos a un modulo Bluetooth Serial Converter que se encuentra en la placa y su función es recibir la señal y convertirla a UART(adaptador de comunicación serie asíncrona.), para luego ser enviada a un max 232 y de allí a un conector DB9 para conectar al puerto serial

de la PC mediante un cable USB-RS232.

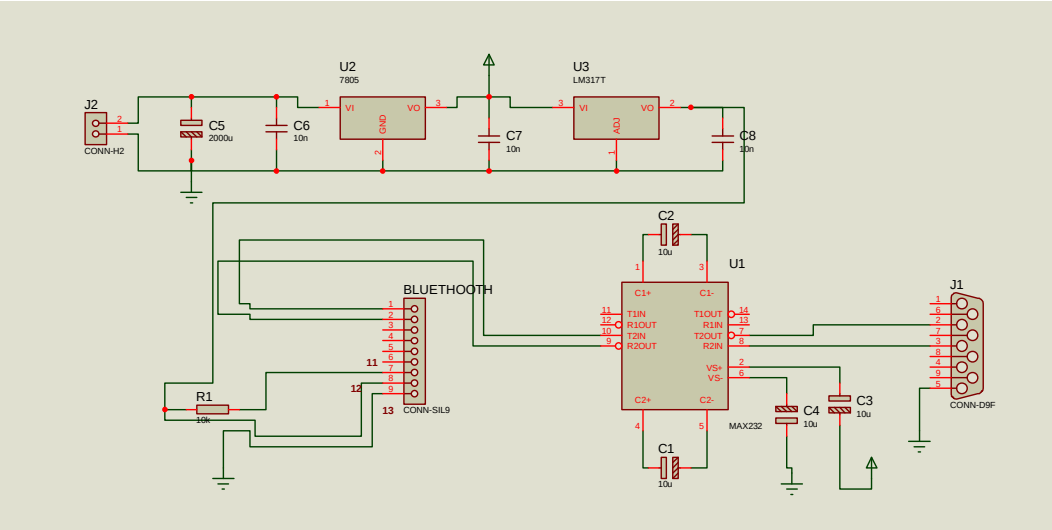


Figura VI.3 Diagrama Placa Bluetooth

6.2.2 Etapa Central Maestro

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=MSComm1.Input

4. Procesamos la información de acuerdo a los datos y enviamos al puerto de salida la instrucción necesaria para el direccionamiento del vehiculo(robot)

```
MSComm2.Output = "A"   'adelante
```

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

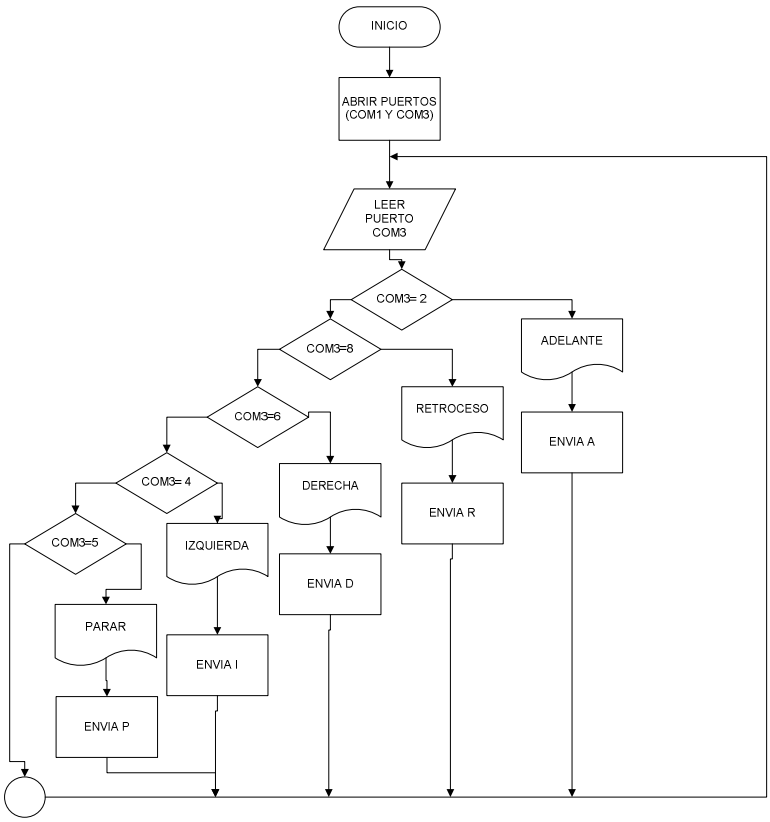


Figura VI.4 Diagrama de flujo equipo central

6.2.3 Etapa Control Vehículo (Robot)

6.2.3.1 Software

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_RCSTA 90h Determinar la habilitación del registro de transmisión
- DEFINA HSER_TXSTA 20h Determinación de la velocidad de transmisión
- DEFINA HSER_BAUD 2400 Determinar la rata de transmisión

Además se utilizo la función de selección CASE para dar paso a los distintos movimientos del carro según la lectura del puerto serial.

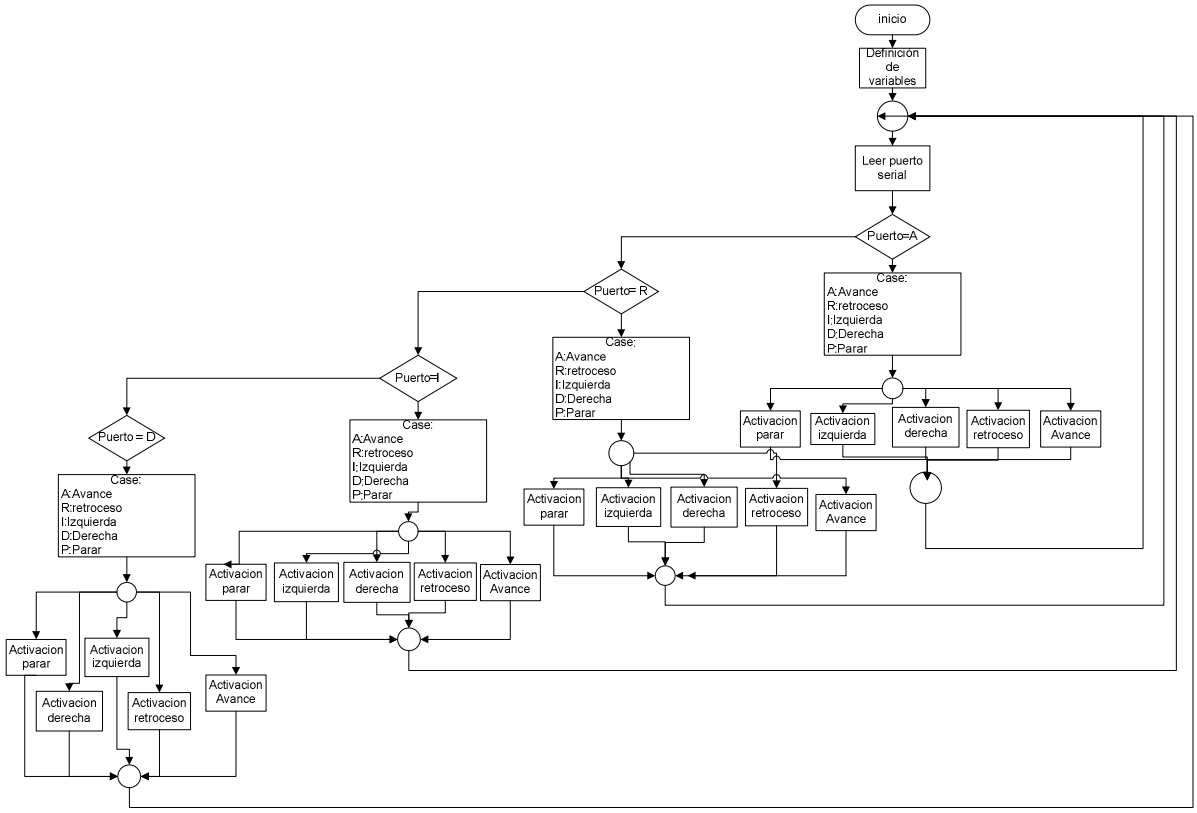


Figura VI.5 Diagrama de flujo Microcontrolador

6.2.3.2 Hardware

Se utilizo un carro de juguete de control remoto para aprovechar los transmisores inalámbricos que posee el mismo. La placa está diseñada con un conector DB9 para recibir las señales del puerto serie y enviarlas a un max 232 y de allí al microcontrolador PIC16f628A en el cual las señales son procesadas y enviadas a 4 relés que están conectadas al control remoto que envía la señal de forma inalámbrica al receptor que está integrado en el carro y produce los movimientos del mismo.

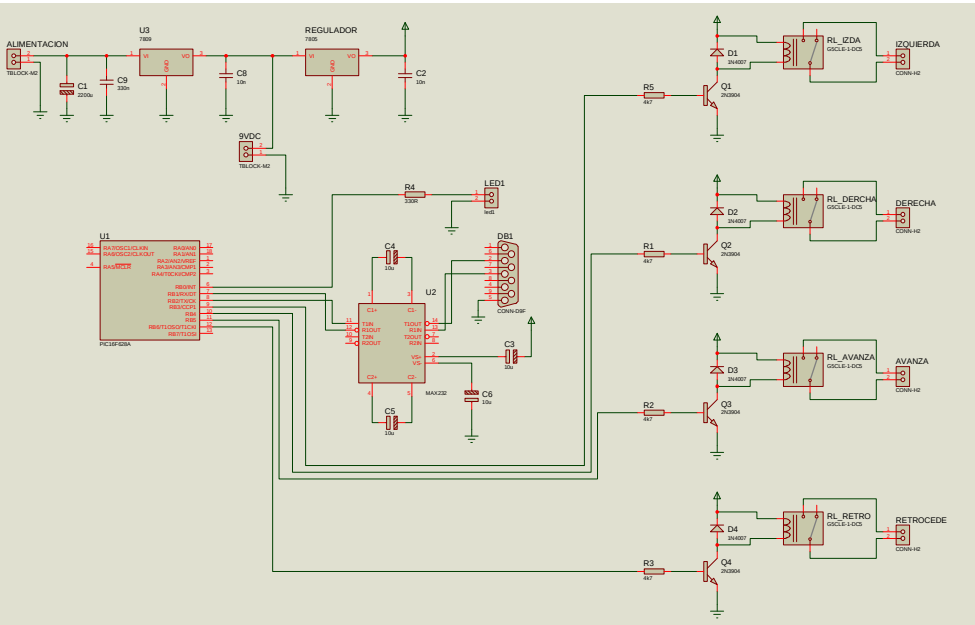


Figura VI.6 Diagrama Placa Carro

CAPÍTULO VII

PRUEBAS Y ANALISIS

7.1 Pruebas de Celular

1. Activamos el bluetooth del celular e ingresamos a la aplicación Carro

Figura VII.01 Ingreso a la aplicación

2. Al ingresar se visualiza la pantalla de presentación y nos da un mensaje de presionar una tecla.

Figura VII.02 Pantalla inicial

3. Inicia la búsqueda del dispositivo bluetooth

Figura VII.03 Buscando Dispositivos

4. Al encontrar el dispositivo visualiza la MAC ADRESS y procede a elegir un puerto

Figura VII.04 Elegir Puerto

5. Una vez Conectado con el puerto aparece una pantalla que nos permite elegir los movimientos del carro con las teclas 2-Adelante, 8- Retroceso, 6-Derecha, 4-Izquierda.

Figura VII.05 Elegir Movimiento

7.2 Pruebas Equipo Central (PC)

1. En primer lugar se ejecuta el programa prueba.exe

Figura VII.06 Programa ejecutable

2. En el cual se reciben las señales del puerto serial COM3 (entrada del modulo bluetooth) y las procesa según la elección de los movimientos

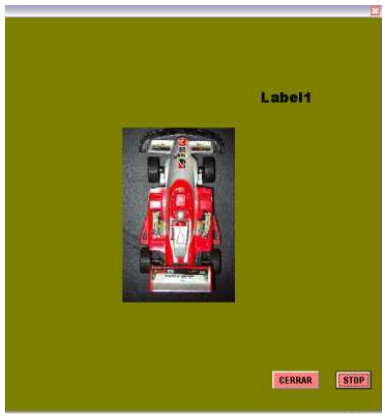


Figura VII.07 Interfaz del programa

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

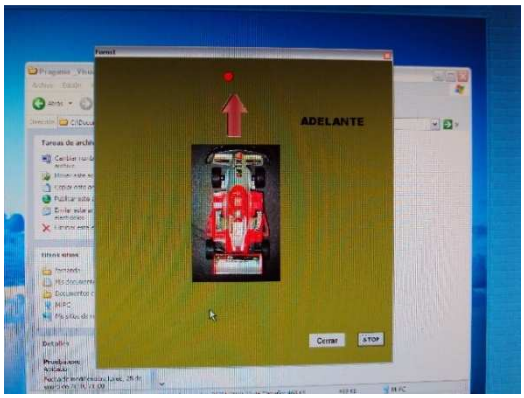


Figura VII.08 Movimiento Adelante

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

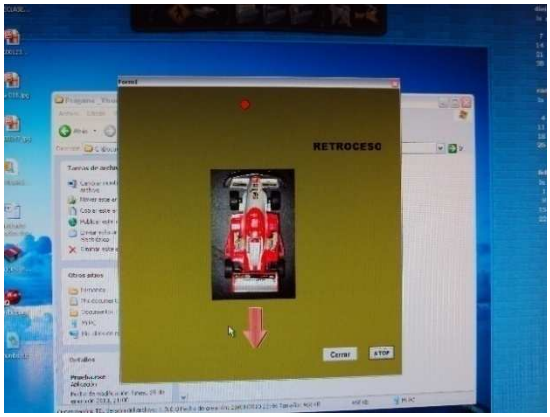


Figura VII.09 Movimiento Retroceso

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

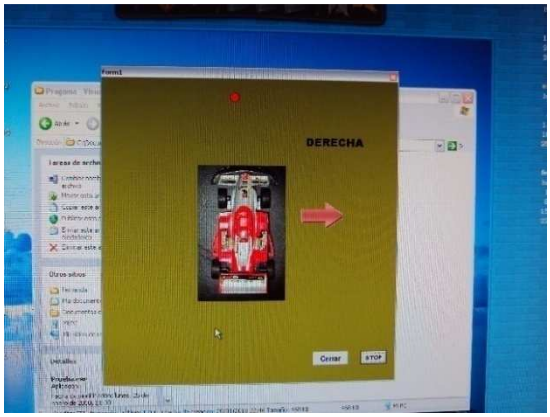


Figura VII.10 Movimiento Derecha

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

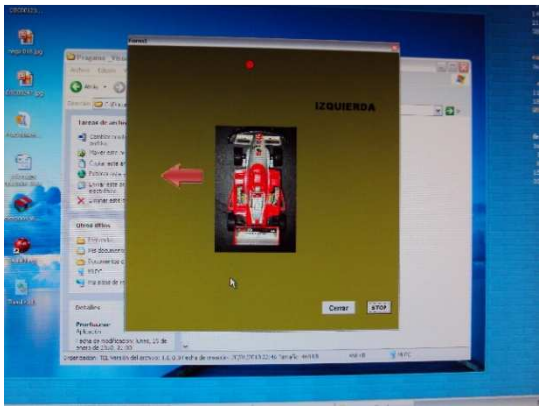


Figura VII.11 Movimiento Izquierda

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

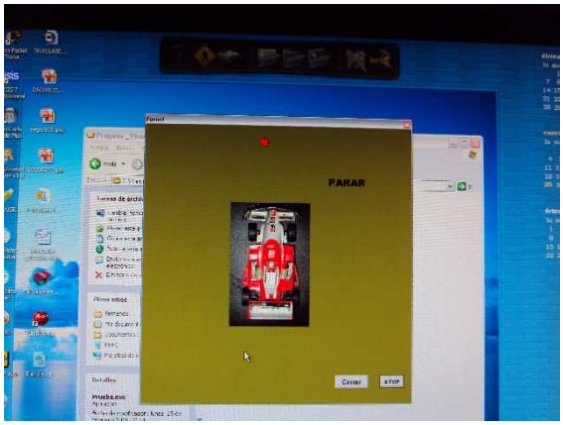


Figura VII.12 Stop

CONCLUSIONES.

1. El proyecto se diseñó utilizando diferentes lenguajes de programación entre ellos le lenguaje Viasual Basic, Lenguaje Java y Microcode que permitió realizar una programación estable y de fácil comprensión.

2. Para manejar el standar bluetooth se utilizo el paquete Java 2Microedition 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.

3. El programa de Equipo Central (PC) fue desarrollado utilizando el Lenguaje Visual Basic orientado a objetos lo que permitió controlar las señales del puerto serie a través de sus componentes MSComm y Timer para leer los puertos de forma continua.

4. En la programación del microcontrolador se utilizo la función HSEROUT que envía uno o varios datos en serie y permite configurar el registro, la velocidad y la rata de transmisión.

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

RECOMENDACIONES

1. Se debe tener cuidado con el manejo del modulo bluetooth ya que es un componente de fácil destrucción por su tamaño.

2. La aplicación del celular la podemos utilizar en cualquier equipo de telefonía que tenga soporte Java y se la carga a través de de archivo .jar

3. Para la alimentación del circuito debemos utilizar un voltaje máximo de 9 voltios debido a que los componentes utilizan voltajes bajos y los disipadores pueden calentarse mucho y dañar al circuito.

4. Para utilizar el sistema de debe instalar en la PC el driver msComm32 que nos permite manejar el Cable USB-RS232 para la entrada de datos desde el celular a la PC.

5. En la búsqueda de dispositivos desde el equipo celular se debe apagar el bluetooth de otros equipos así como de laptos ya que puede interferir en la ejecución del programa.

RESUMEN

Diseño de un sistema de conectividad inalámbrica utilizando bluetooth e interfaces que permita un flujo de datos, entre un teléfono móvil, un equipo que actúa como centro de control y un carro de juguete.

Se utilizó un equipo de telefonía móvil en el cual se desarrolló una aplicación java que permite el control del sistema. El equipo de telefonía móvil se conecta al equipo central mediante una placa que contiene un modulo bluetooth que recibe el flujo de datos y los envía a 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 carro de juguete.

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

El sistema 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 por lo que es un sistema muy atractivo para seguir investigando y desarrollando nuevas aplicaciones en base al estudio realizado.

SUMMARY

Designing a system using Bluetooth wireless connectivity and interfaces which allows a flow of data between a mobile phone, equipment that acts as a control center and toy car.

We used a mobile phone equipment in which we developed a Java application that allows control of the system. The mobile phone equipment connects to the central equipment through a plate containing a bluetooth module that receives the flow of data and sent to the PC for processing in a application developed in visual basic.

Finally, the central equipment sends data to a plate connected to the serial port which controls movement of the toy car.

We obtained a system with a basic level of wireless communication between the mobile phone equipment and a toy car with a negligible response time what allows more complex functions.

The wireless connectivity system uses the maximum capacity of a equipment mobile phone through Bluetooth technology enables a wide field applications making it a very attractive system to continue investigating and developing new applications based on the study done.

ANEXOS

Anexo A. Datasheet del PIC 16f628A

PIC16F627A/628A/648A

Data Sheet

FLASH-Based

8-Bit CMOS Microcontrollers

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

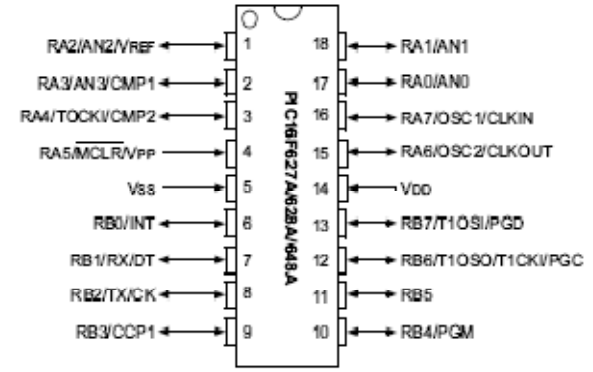
Peripheral Features:

- 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

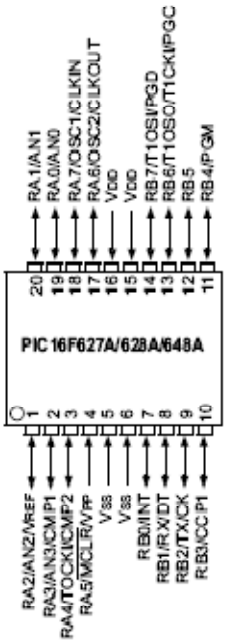
Device	Program Memory	Data Memory		I/O	CCP (PWM)	USART	Comparators	Timers 8/16-bit
	FLASH (words)	SRAM (bytes)	EEPROM (bytes)					
PIC16F627A	1024	224	128	16	1	Y	2	2/1
PIC16F628A	2048	224	128	16	1	Y	2	2/1
PIC16F648A	4096	256	256	16	1	Y	2	2/1

Pin Diagrams

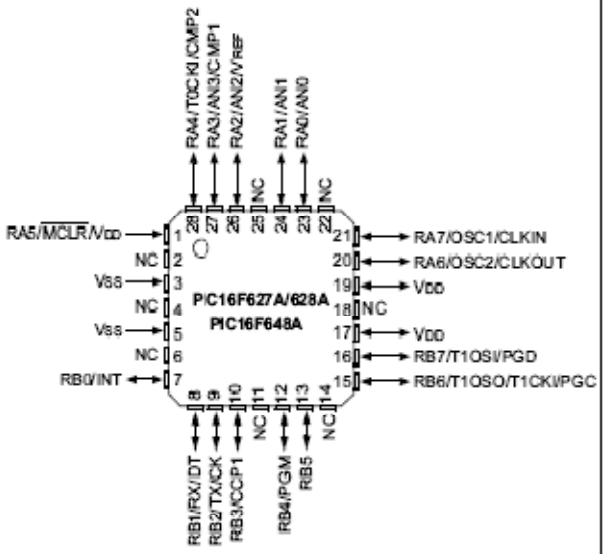
PDIP, SOIC



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
Features	Interrupt Sources	10	10	10	10	10	10
	I/O Pins	16	16	16	16	16	16
	Voltage Range (Volts)	3.0-5.5	3.0-5.5	3.0-5.5	2.0-5.5	2.0-5.5	2.0-5.5
	Brown-out Reset	Yes	Yes	Yes	Yes	Yes	Yes
	Packages	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN

All PICmicro® Family devices have Power-on Reset, selectable Watchdog Timer, selectable Code Protect and high I/O current capability.
All PIC16F627A/628A/648A Family devices use serial programming with clock pin RB6 and data pin RB7.

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.

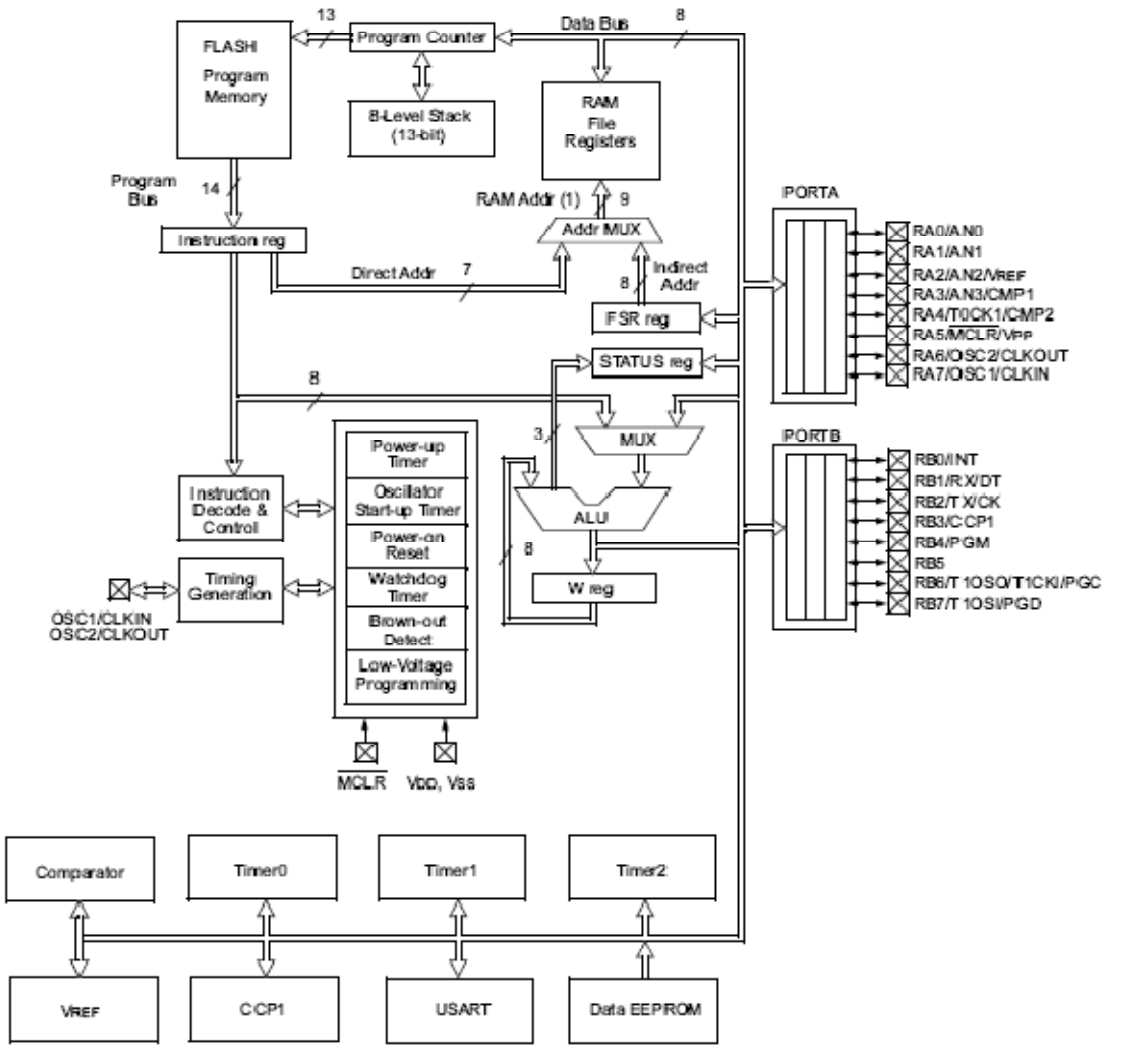
The W register is an 8-bit working register used for ALU operations. It is not an addressable register.

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 Borrow and 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 $\overline{MCLR}$/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 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
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.

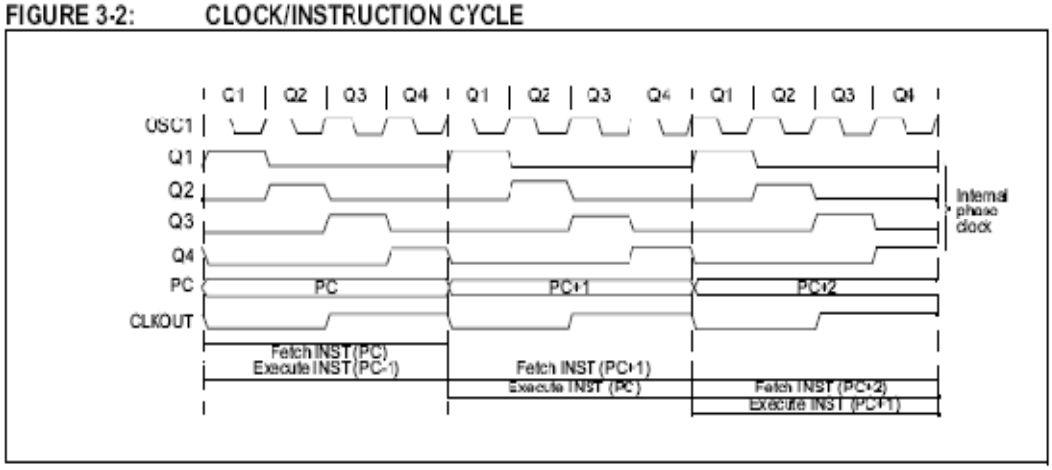
3.2 Instruction Flow/Pipelining

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).

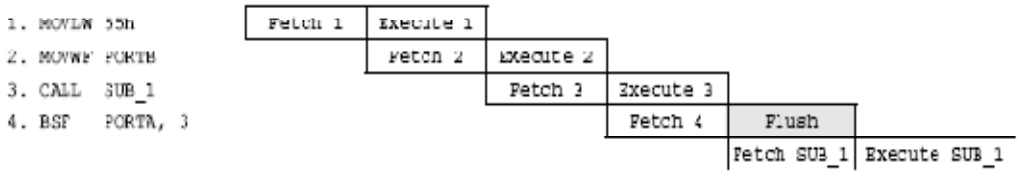
A fetch cycle begins with the program counter (PC) incrementing in Q1.

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



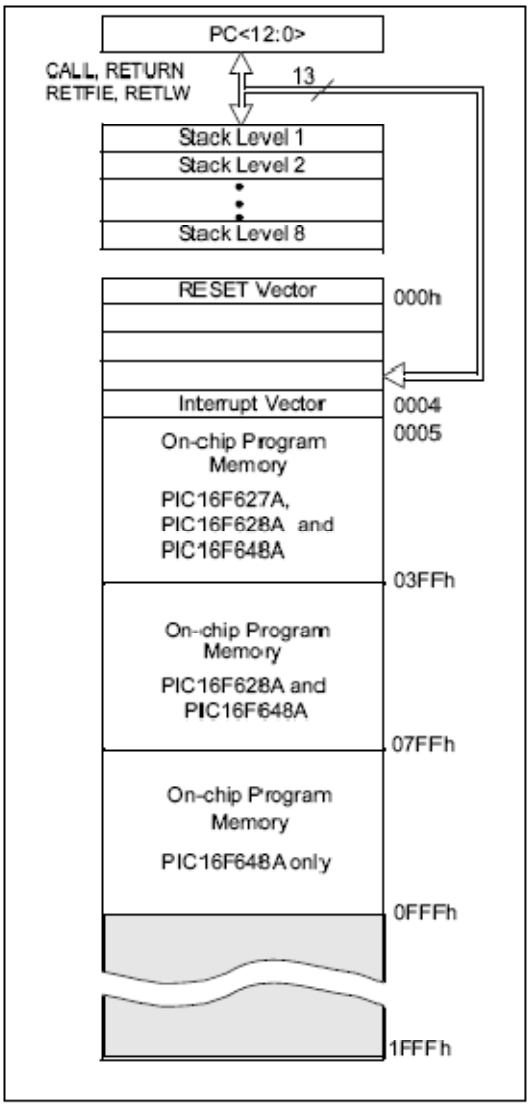
All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

4.0 MEMORY ORGANIZATION

4.1 Program Memory Organization

The PIC16F627A/628A/648A has a 13-bit program counter capable of addressing an 8K x 14 program memory space. Only the first 1K x 14 (0000h - 03FFh) for the PIC16F627A, 2K x 14 (0000h - 07FFh) for the PIC16F628A and 4K x 14 (0000h - 0FFFh) for the PIC16F648A are physically implemented. Accessing a location above these boundaries will cause a wrap-around within the first 1K x 14 space (PIC16F627A), 2K x 14 space (PIC16F628A) or 4K x 14 space (PIC16F648A). The RESET vector is at 0000h and the interrupt vector is at 0004h (Figure 4-1).

FIGURE 4-1: PROGRAM MEMORY MAP AND STACK



4.2 Data Memory Organization

The data memory (Figure 4-2 and Figure 4-3) is partitioned into four banks, which contain the general purpose registers and the Special Function Registers (SFR). The SFR's are located in the first 32 locations of each Bank. There are general purpose registers implemented as static RAM in each Bank. Table 4-1 lists the general purpose register available in each of the four banks.

TABLE 4-1: GENERAL PURPOSE STATIC RAM REGISTERS

	PIC16F627A/628A	PIC16F648A
Bank0	20-7Fh	20-7Fh
Bank1	A0h-FF	A0h-FF
Bank2	120h-14Fh, 170h-17Fh	120h-17Fh
Bank3	1F0h-1FFh	1F0h-1FFh

Addresses F0h-FFh, 170h-17Fh and 1F0h-1FFh are implemented as common RAM and mapped back to addresses 70h-7Fh.

Table 4-2 lists how to access the four banks of registers via the STATUS Register bits RP1 and RP0.

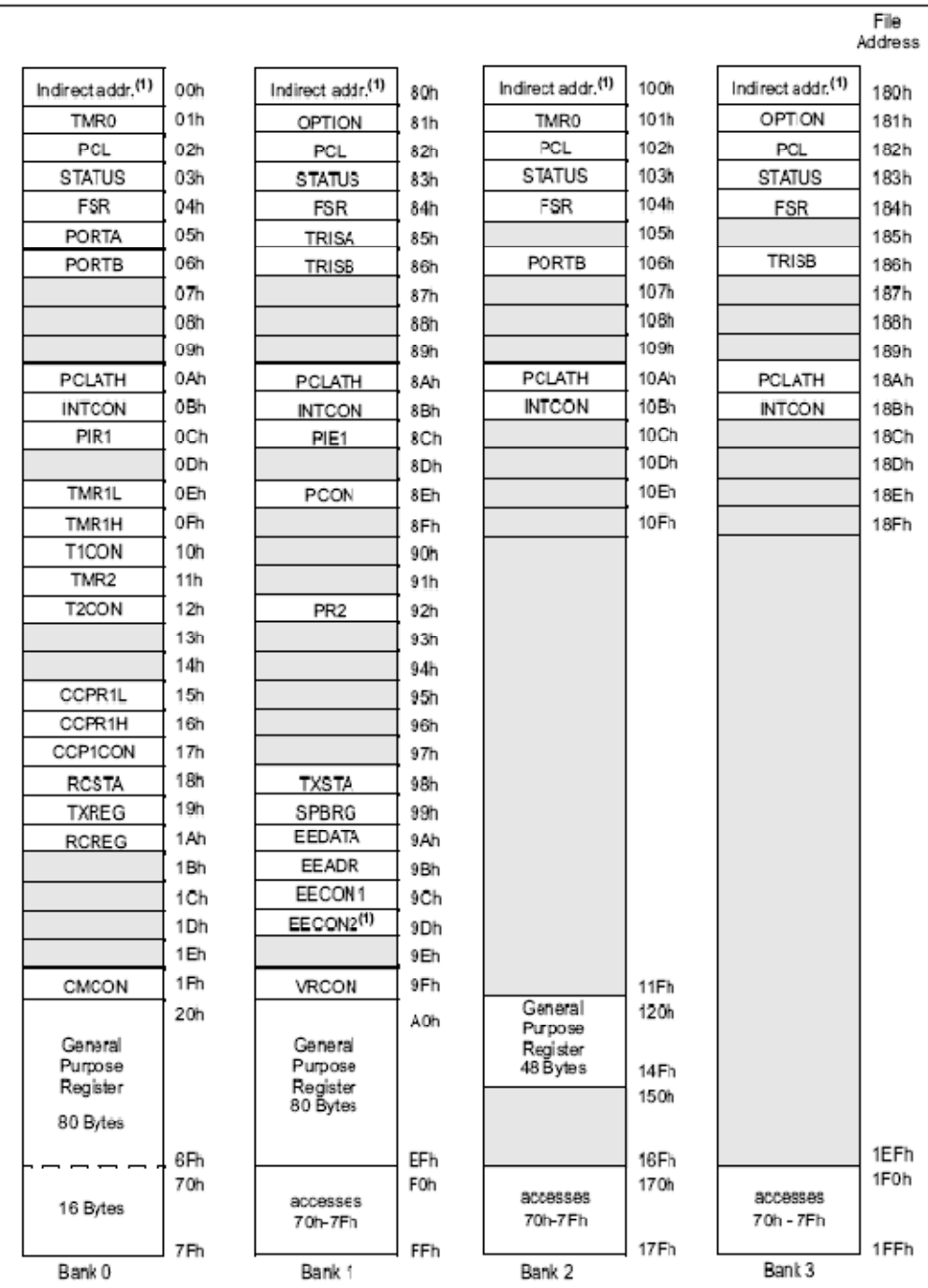
TABLE 4-2: ACCESS TO BANKS OF REGISTERS

	RP1	RP0
Bank0	0	0
Bank1	0	1
Bank2	1	0
Bank3	1	1

4.2.1 GENERAL PURPOSE REGISTER FILE

The register file is organized as 224 x 8 in the PIC16F627A/628A and 256 x 8 in the PIC16F648A. Each is accessed either directly or indirectly through the File Select Register (FSR), See Section 4.4.

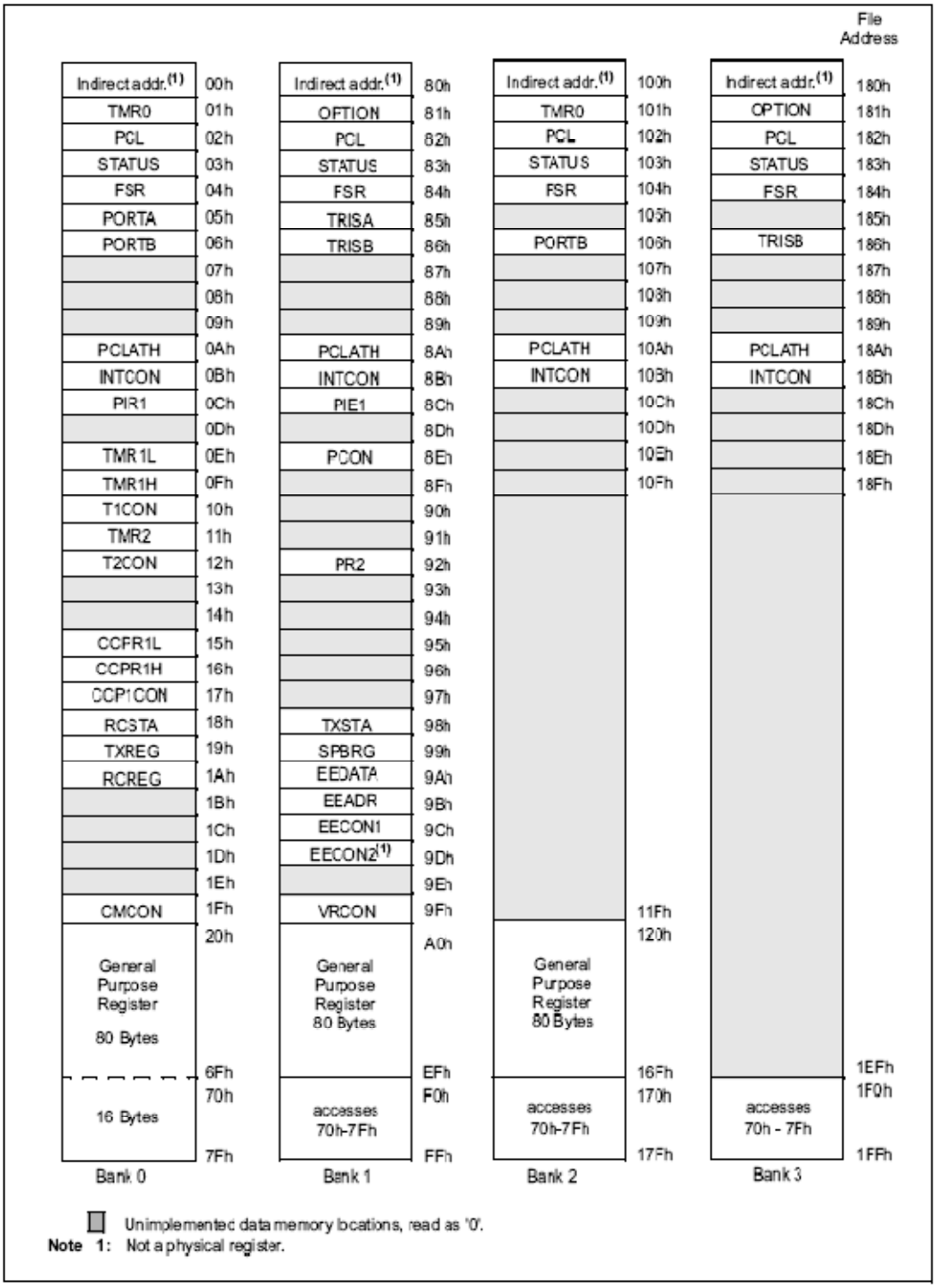
FIGURE 4-2: DATA MEMORY MAP OF THE PIC16F627A AND PIC16F628A



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	76
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	45
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 "000uu1uu" (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 $\overline{Digit\ Borrow}$ 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

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

4.2.2.3 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 $\overline{\textbf{POR}}$: 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 $\overline{\textbf{BOR}}$: 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 = Reacable 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 IPORTA 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 VPP. When VPP 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 overridden, the data reads '0' and the 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

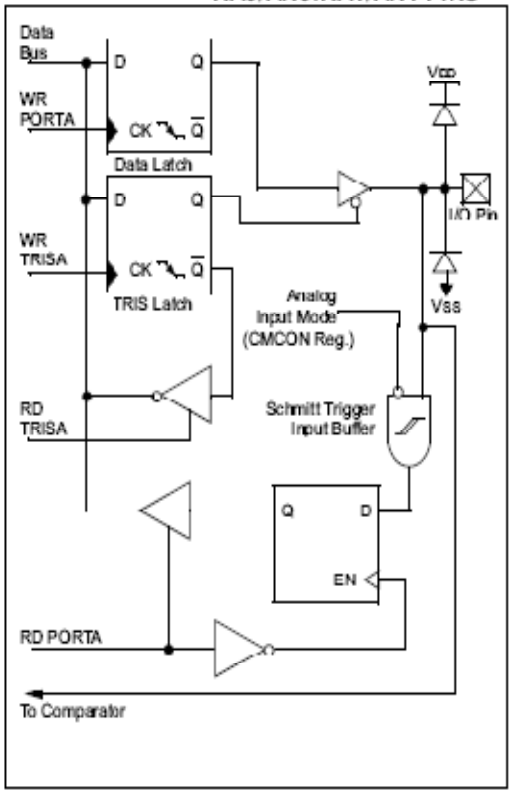


TABLE 5-1: PORTA FUNCTIONS

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/$\overline{\text{MCLR}}$/VPP	RA5	ST	—	Input port
	$\overline{\text{MCLR}}$	ST	—	Master clear. When configured as $\overline{\text{MCLR}}$, this pin is an active low RESET to the device. Voltage on $\overline{\text{MCLR}}$/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.

Anexo B. Datasheet del MAX232

19-4323; Rev 11; 2/08

MAXIM

+5V-Powered, Multichannel RS-232 Drivers/Receivers

General Description

The MAX220–MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where ±12V is not available.

These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5µW. The MAX225, MAX233, MAX235, and MAX245/MAX246/MAX247 use no external components and are recommended for applications where printed circuit board space is critical.

Applications

Portable Computers
Low-Power Modems
Interface Translation
Battery-Powered RS-232 Systems
Multidrop RS-232 Networks

Features

Superior to Bipolar

- Operate from Single +5V Power Supply (+5V and +12V—MAX231/MAX239)
- Low-Power Receive Mode in Shutdown (MAX223/MAX242)
- Meet All EIA/TIA-232E and V.28 Specifications
- Multiple Drivers and Receivers
- 3-State Driver and Receiver Outputs
- Open-Line Detection (MAX243)

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX220CPE	0°C to +70°C	16 Plastic DIP
MAX220CSE	0°C to +70°C	16 Narrow SO
MAX220CWE	0°C to +70°C	16 Wide SO
MAX220C/D	0°C to +70°C	Dice*
MAX220EPE	-40°C to +85°C	16 Plastic DIP
MAX220ESE	-40°C to +85°C	16 Narrow SO
MAX220EWE	-40°C to +85°C	16 Wide SO
MAX220EJE	-40°C to +85°C	16 CERDIP
MAX220MJE	-55°C to +125°C	16 CERDIP

Ordering Information continued at end of data sheet.

*Contact factory for dice specifications.

Selection Table

Part Number	Power Supply (V)	No. of RS-232 Drivers/Rx	No. of Ext. Caps	Nominal Cap. Value (µF)	SHDN & Three-State	Rx Active in SHDN	Data Rate (kbps)	Features
MAX220	+5	2/2	4	0.1	No	—	120	Ultra-low-power, industry-standard pinout
MAX222	+5	2/2	4	0.1	Yes	—	200	Low-power shutdown
MAX223 (MAX213)	+5	4/5	4	1.0 (0.1)	Yes	✔	120	MAX241 and receivers active in shutdown
MAX225	+5	5/5	0	—	Yes	✔	120	Available in SO
MAX230 (MAX200)	+5	5/0	4	1.0 (0.1)	Yes	—	120	5 drivers with shutdown
MAX231 (MAX201)	+5 and +7.5 to +13.2	2/2	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; same functions as MAX232
MAX232 (MAX202)	+5	2/2	4	1.0 (0.1)	No	—	120 (64)	Industry standard
MAX232A	+5	2/2	4	0.1	No	—	200	Higher slew rate, small caps
MAX233 (MAX203)	+5	2/2	0	—	No	—	120	No external caps
MAX233A	+5	2/2	0	—	No	—	200	No external caps, high slew rate
MAX234 (MAX234)	+5	4/0	4	1.0 (0.1)	No	—	120	Replaces 1488
MAX235 (MAX205)	+5	5/5	0	—	Yes	—	120	No external caps
MAX236 (MAX206)	+5	4/3	4	1.0 (0.1)	Yes	—	120	Shutdown, three-state
MAX237 (MAX207)	+5	5/3	4	1.0 (0.1)	No	—	120	Complements IBM PC serial port
MAX238 (MAX208)	+5	4/4	4	1.0 (0.1)	No	—	120	Replaces 1488 and 1489
MAX239 (MAX209)	+5 and +7.5 to +13.2	3/5	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; single-package solution for IBM PC serial port
MAX240	+5	5/5	4	1.0	Yes	—	120	DIP or flatpack package
MAX241 (MAX211)	+5	4/5	4	1.0 (0.1)	Yes	—	120	Complete IBM PC serial port
MAX242	+5	2/2	4	0.1	Yes	✔	200	Separate shutdown and enable
MAX243	+5	2/2	4	0.1	No	—	200	Open-line detection simplifies cabling
MAX244	+5	8/10	4	1.0	No	—	120	High slew rate
MAX245	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, two shutdown modes
MAX246	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, three shutdown modes
MAX247	+5	8/9	0	—	Yes	✔	120	High slew rate, int. caps, nine operating modes
MAX248	+5	8/8	4	1.0	Yes	✔	120	High slew rate, selective half-chip enables
MAX249	+5	6/10	4	1.0	Yes	✔	120	Available in quad flatpack package

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX220/222/232A/233A/242/243

Supply Voltage (V_{CC}) ..... -0.3V to +6V
Input Voltages
T_{IN} ..... -0.3V to (V_{CC} + 0.3V)
R_{IN} (Except MAX220) ..... ±30V
R_{IN} (MAX220) ..... ±25V
T_{OUT} (Except MAX220) (Note 1) ..... ±15V
T_{OUT} (MAX220) ..... +13.2V
Output Voltages
T_{OUT} ..... ±15V
R_{OUT} ..... -0.3V to (V_{CC} + 0.3V)
Driver/Receiver Output Short Circuited to GND ..... Continuous
Continuous Power Dissipation (T_A = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C) 842mW
18-Pin Plastic DIP (derate 11.11mW/°C above +70°C) 889mW
20-Pin Plastic DIP (derate 8.00mW/°C above +70°C) 440mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) 696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
18-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
20-Pin Wide SO (derate 10.00mW/°C above +70°C) 800mW
20-Pin SSOP (derate 8.00mW/°C above +70°C) 640mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C) 800mW
18-Pin CERDIP (derate 10.53mW/°C above +70°C) 842mW
Operating Temperature Ranges
MAX2_ _AC_ _, MAX2_ _C_ _ ..... 0°C to +70°C
MAX2_ _AE_ _, MAX2_ _E_ _ ..... -40°C to +85°C
MAX2_ _AM_ _, MAX2_ _M_ _ ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10s) ..... +300°C

Note 1: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.
Note 2: For the MAX220, V+ and V- can have a maximum magnitude of 7V, but their absolute difference cannot exceed 13V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2-C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High	All devices except MAX220		2	1.4		V
	MAX220: V_{CC} = 5.0V		2.4			
Logic Pull-Up/Input Current	All except MAX220, normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown, MAX220			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V MAX222/242			±0.01	±10	µA
	V_{CC} = $\overline{SHDN}$ = 0V, V_{OUT} = ±15V			±0.01	±10	
Data Rate				200	116	kbps
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±22		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V_{CC} = 5V	All except MAX243 $R2_{IN}$	0.8	1.3		V
		MAX243 $R2_{IN}$ (Note 2)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	All except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	All except MAX243, V_{CC} = 5V, no hysteresis in shdn.		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Sinking V_{OUT} = V_{CC}		10	30		

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243 (continued)

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
TTL/CMOS Output Leakage Current	$\overline{SHDN}$ = V_{CC} or $\overline{EN}$ = V_{CC} ($\overline{SHDN}$ = 0V for MAX222), 0V ≤ V_{OUT} ≤ V_{CC}			±0.05	±10	µA
$\overline{EN}$ Input Threshold Low	MAX242			1.4	0.8	V
$\overline{EN}$ Input Threshold High	MAX242		2.0	1.4		V
Operating Supply Voltage			4.5		5.5	V
V_{CC} Supply Current ($\overline{SHDN}$ = V_{CC}), Figures 5, 6, 11, 19	No load	MAX220		0.5	2	mA
		MAX222/232A/233A/242/243		4	10	
	3kΩ load both inputs	MAX220		12		
		MAX222/232A/233A/242/243		15		
Shutdown Supply Current	MAX222/242	T_A = +25°C		0.1	10	µA
		T_A = 0°C to +70°C		2	50	
		T_A = -40°C to +85°C		2	50	
		T_A = -55°C to +125°C		35	100	
$\overline{SHDN}$ Input Leakage Current	MAX222/242				±1	µA
$\overline{SHDN}$ Threshold Low	MAX222/242			1.4	0.8	V
$\overline{SHDN}$ Threshold High	MAX222/242		2.0	1.4		V
Transition Slew Rate	C_L = 50pF to 2500pF, R_L = 3kΩ to 7kΩ, V_{CC} = 5V, T_A = +25°C, measured from +3V to -3V or -3V to +3V	MAX222/232A/233A/242/243	6	12	30	V/µs
		MAX220	1.5	3	30	
Transmitter Propagation Delay TLL to RS-232 (Normal Operation), Figure 1	t_{PHLT}	MAX222/232A/233A/242/243		1.3	3.5	µs
		MAX220		4	10	
	t_{PLHT}	MAX222/232A/233A/242/243		1.5	3.5	
		MAX220		5	10	
Receiver Propagation Delay RS-232 to TLL (Normal Operation), Figure 2	t_{PHLR}	MAX222/232A/233A/242/243		0.5	1	µs
		MAX220		0.6	3	
	t_{PLHR}	MAX222/232A/233A/242/243		0.6	1	
		MAX220		0.8	3	
Receiver Propagation Delay RS-232 to TLL (Shutdown), Figure 2	t_{PHLS}	MAX242		0.5	10	µs
	t_{PLHS}	MAX242		2.5	10	
Receiver-Output Enable Time, Figure 3	t_{ER}	MAX242		125	500	ns
Receiver-Output Disable Time, Figure 3	t_{DR}	MAX242		160	500	ns
Transmitter-Output Enable Time ($\overline{SHDN}$ Goes High), Figure 4	t_{ET}	MAX222/242, 0.1µF caps (includes charge-pump start-up)		250		µs
Transmitter-Output Disable Time ($\overline{SHDN}$ Goes Low), Figure 4	t_{DT}	MAX222/242, 0.1µF caps		600		ns
Transmitter + to - Propagation Delay Difference (Normal Operation)	t_{PHLT} - t_{PLHT}	MAX222/232A/233A/242/243		300		ns
		MAX220		2000		
Receiver + to - Propagation Delay Difference (Normal Operation)	t_{PHLR} - t_{PLHR}	MAX222/232A/233A/242/243		100		ns
		MAX220		225		

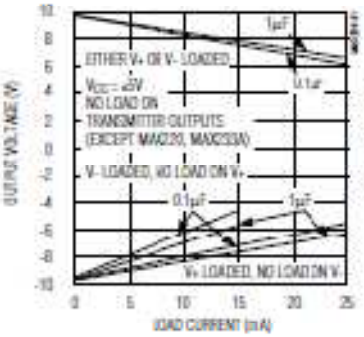
Note 3: MAX243 $R2_{OUT}$ is guaranteed to be low when $R2_{IN}$ is ≥ 0V or is floating.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

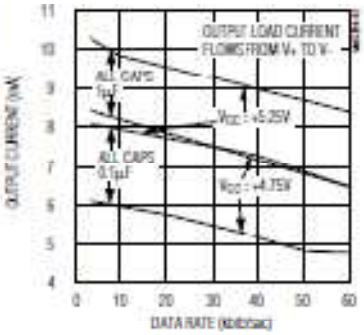
Typical Operating Characteristics

MAX220/MAX222/MAX232A/MAX233A/MAX242/MAX243

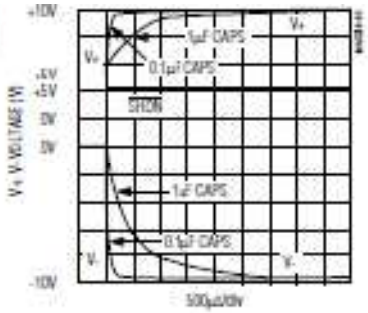
OUTPUT VOLTAGE vs. LOAD CURRENT



AVAILABLE OUTPUT CURRENT vs. DATA RATE



MAX222/MAX242 ON-TIME EXITING SHUTDOWN



ABSOLUTE MAXIMUM RATINGS—MAX223/MAX230–MAX241

V_{CC} ..... -0.3V to +6V
V+ ..... (V_{CC} - 0.3V) to +14V
V- ..... +0.3V to -14V
Input Voltages
T_{IN} ..... -0.3V to (V_{CC} + 0.3V)
R_{IN} ..... ±30V
Output Voltages
T_{OUT} ..... (V+ + 0.3V) to (V- - 0.3V)
R_{OUT} ..... -0.3V to (V_{CC} + 0.3V)
Short-Circuit Duration, T_{OUT} ..... Continuous
Continuous Power Dissipation (T_A = +70°C)
14-Pin Plastic DIP (derate 10.00mW/°C above +70°C) 800mW
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C) 842mW
20-Pin Plastic DIP (derate 11.11mW/°C above +70°C) 889mW
24-Pin Narrow Plastic DIP
(derate 13.33mW/°C above +70°C) 1.07W
24-Pin Plastic DIP (derate 9.09mW/°C above +70°C) 500mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
20-Pin Wide SO (derate 10.00mW/°C above +70°C) 800mW
24-Pin Wide SO (derate 11.76mW/°C above +70°C) 941mW
28-Pin Wide SO (derate 12.50mW/°C above +70°C) 1W
44-Pin Plastic FP (derate 11.11mW/°C above +70°C) 889mW
14-Pin CERDIP (derate 9.09mW/°C above +70°C) 727mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C) 800mW
20-Pin CERDIP (derate 11.11mW/°C above +70°C) 889mW
24-Pin Narrow CERDIP
(derate 12.50mW/°C above +70°C) 1W
24-Pin Sidebraze (derate 20.0mW/°C above +70°C) 1.6W
28-Pin SSOP (derate 9.52mW/°C above +70°C) 762mW
Operating Temperature Ranges
MAX2_ _C_ _ ..... 0°C to +70°C
MAX2_ _E_ _ ..... -40°C to +85°C
MAX2_ _M_ _ ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10s) ..... +300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX223/MAX230–MAX241

(MAX223/230/232/234/236/237/238/240/241, V_{CC} = +5V ±10; MAX233/MAX235, V_{CC} = 5V ±5%, C1–C4 = 1.0µF; MAX231/MAX239, V_{CC} = 5V ±10%; V+ = 7.5V to 13.2V; T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to ground		±5.0	±7.3		V
V_{CC} Power-Supply Current	No load, T_A = +25°C	MAX232/233		5	10	mA
		MAX223/230/234–238/240/241		7	15	
		MAX231/239		0.4	1	
V+ Power-Supply Current		MAX231		1.8	5	mA
		MAX239		5	15	
Shutdown Supply Current	T_A = +25°C	MAX223		15	50	µA
		MAX230/235/236/240/241		1	10	
Input Logic Threshold Low	T_{IN}; EN, $\overline{SHDN}$ (MAX233); $\overline{EN}$, SHDN (MAX230/235–241)				0.8	V
Input Logic Threshold High	T_{IN}		2.0			V
	EN, $\overline{SHDN}$ (MAX223); $\overline{EN}$, SHDN (MAX230/235/236/240/241)		2.4			
Logic Pull-Up Current	T_{IN} = 0V			1.5	200	µA
Receiver Input Voltage Operating Range			-30		30	V

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ELECTRICAL CHARACTERISTICS—MAX223/MAX230–MAX241 (continued)

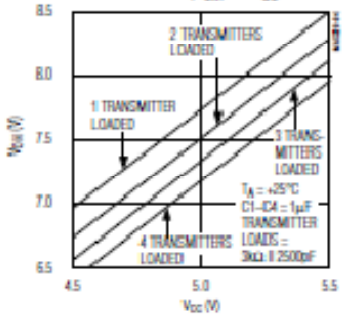
(MAX223/230/232/234/236/237/239/240/241, V_{CC} = +5V ±10; MAX233/MAX235, V_{CC} = 5V ±5%, C1–C4 = 1.0µF; MAX231/MAX239, V_{CC} = 5V ±10%; V+ = 7.5V to 13.2V; T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS			MIN	TYP	MAX	UNITS
RS-232 Input Threshold Low	T_A = +25°C, V_{CC} = 5V	Normal operation SHDN = 5V (MAX223) SHDN = 0V (MAX235/236/240/241)		0.8	1.2		V
		Shutdown (MAX223) SHDN = 0V, EN = 5V ($R4_{IN}$, $R5_{IN}$)		0.6	1.5		
RS-232 Input Threshold High	T_A = +25°C, V_{CC} = 5V	Normal operation SHDN = 5V (MAX223) SHDN = 0V (MAX235/236/240/241)			1.7	2.4	V
		Shutdown (MAX223) SHDN = 0V, EN = 5V ($R4_{IN}$, $R5_{IN}$)			1.5	2.4	
RS-232 Input Hysteresis	V_{CC} = 5V, no hysteresis in shutdown			0.2	0.5	1.0	V
RS-232 Input Resistance	T_A = +25°C, V_{CC} = 5V			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 1.6mA (MAX231/232/233, I_{OUT} = 3.2mA)					0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1mA			3.5	V_{CC} - 0.4		V
TTL/CMOS Output Leakage Current	0V ≤ R_{OUT} ≤ V_{CC}; EN = 0V (MAX223); EN = V_{CC} (MAX235–241)				0.05	±10	µA
Receiver Output Enable Time	Normal operation	MAX223			600		ns
		MAX235/236/239/240/241			400		
Receiver Output Disable Time	Normal operation	MAX223			900		ns
		MAX235/236/239/240/241			250		
Propagation Delay	RS-232 IN to TTL/CMOS OUT, C_L = 150pF	Normal operation			0.5	10	µs
		SHDN = 0V (MAX223)	t_{PHLS}		4	40	
			t_{PLHS}		6	40	
Transition Region Slew Rate	MAX223/MAX230/MAX234–241, T_A = +25°C, V_{CC} = 5V, R_L = 3kΩ to 7kΩ, C_L = 50pF to 2500pF, measured from +3V to -3V or -3V to +3V			3	5.1	30	V/µs
	MAX231/MAX232/MAX233, T_A = +25°C, V_{CC} = 5V, R_L = 3kΩ to 7kΩ, C_L = 50pF to 2500pF, measured from +3V to -3V or -3V to +3V				4	30	
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V			300			Ω
Transmitter Output Short-Circuit Current					±10		mA

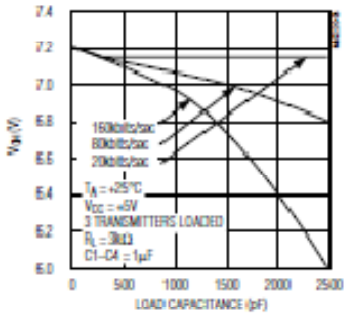
Typical Operating Characteristics

MAX223/MAX230–MAX241

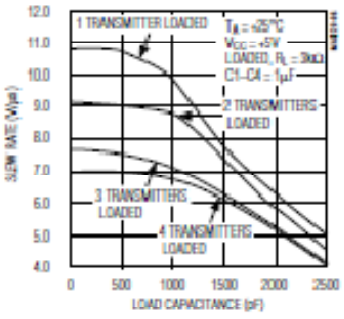
TRANSMITTER OUTPUT VOLTAGE (V_{OH}) vs. V_{CC}



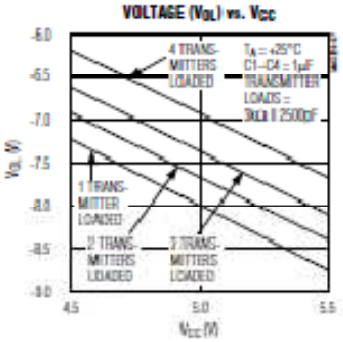
TRANSMITTER OUTPUT VOLTAGE (V_{OH}) vs. LOAD CAPACITANCE AT DIFFERENT DATA RATES



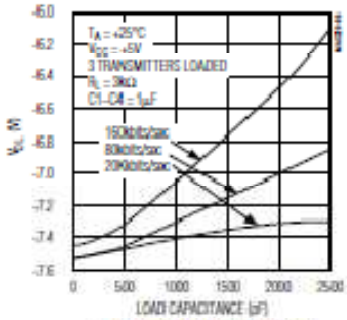
TRANSMITTER SLEW RATE vs. LOAD CAPACITANCE



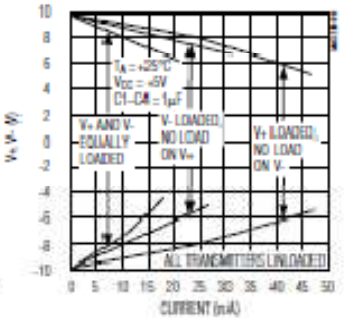
TRANSMITTER OUTPUT VOLTAGE (V_{OL}) vs. V_{CC}



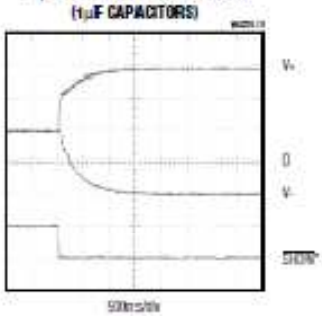
TRANSMITTER OUTPUT VOLTAGE (V_{OL}) vs. LOAD CAPACITANCE AT DIFFERENT DATA RATES



TRANSMITTER OUTPUT VOLTAGE (V_+, V_-) vs. LOAD CURRENT



V_+, V_- WHEN EXITING SHUTDOWN (1µF CAPACITORS)



*SHUTDOWN POLARITY IS REVERSED FOR NON MAXIM PARTS

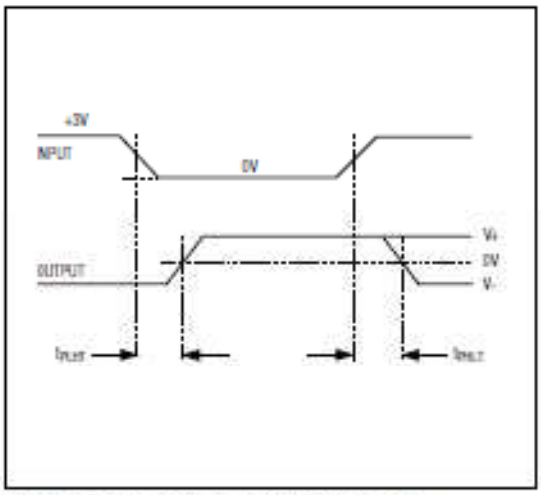


Figure 1. Transmitter Propagation-Delay Timing

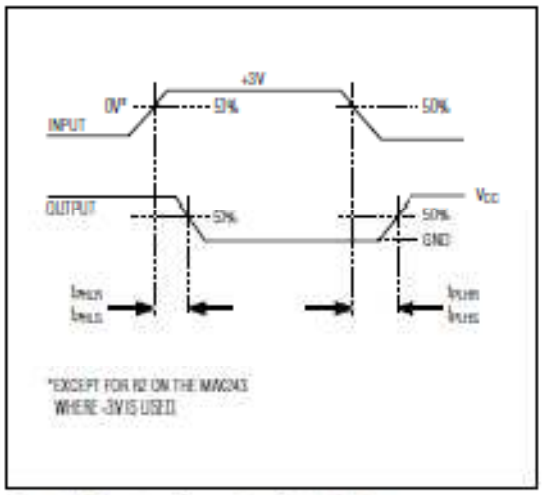


Figure 2. Receiver Propagation-Delay Timing

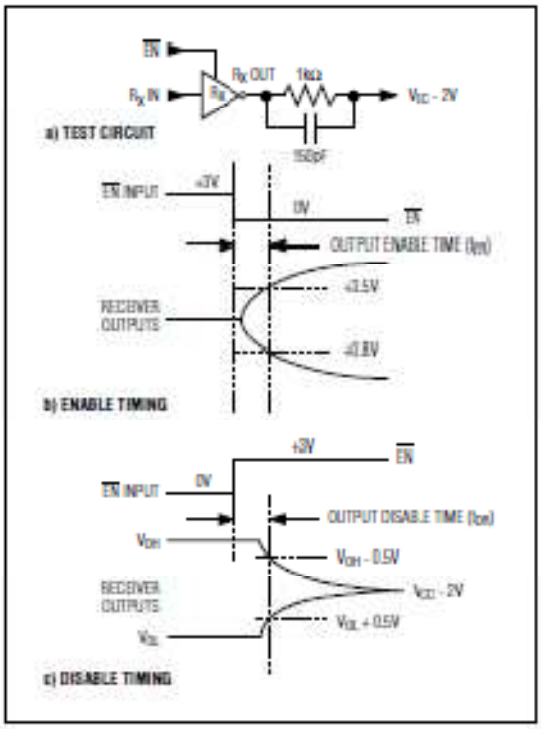


Figure 3. Receiver-Output Enable and Disable Timing

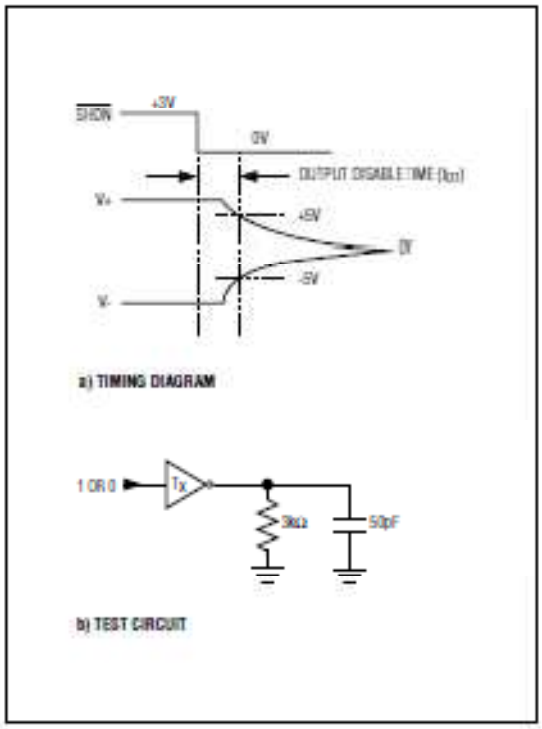
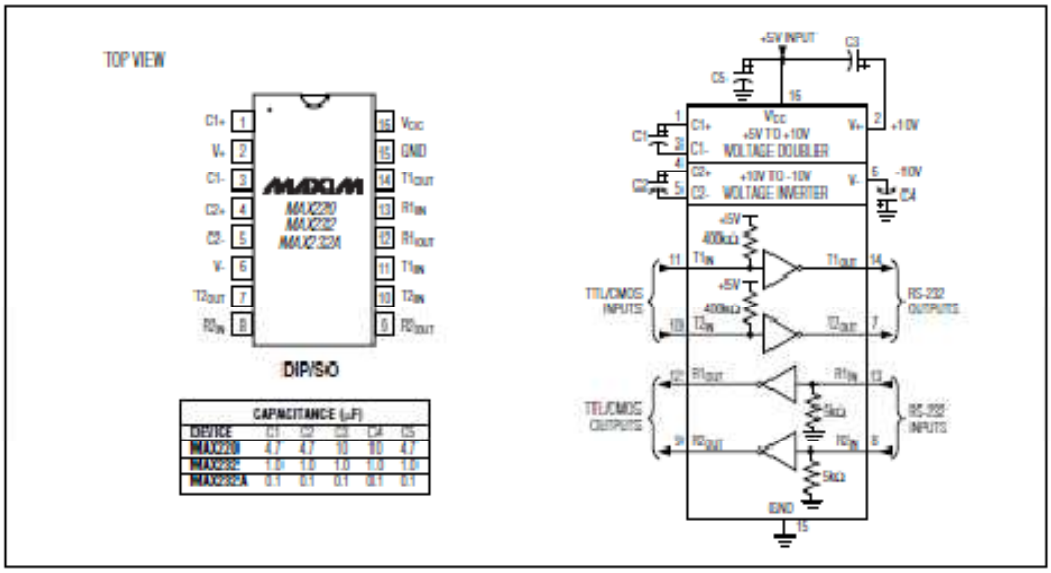


Figure 4. Transmitter-Output Disable Timing



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure 5. MAX220/MAX232/MAX232A Pin Configuration and Typical Operating Circuit

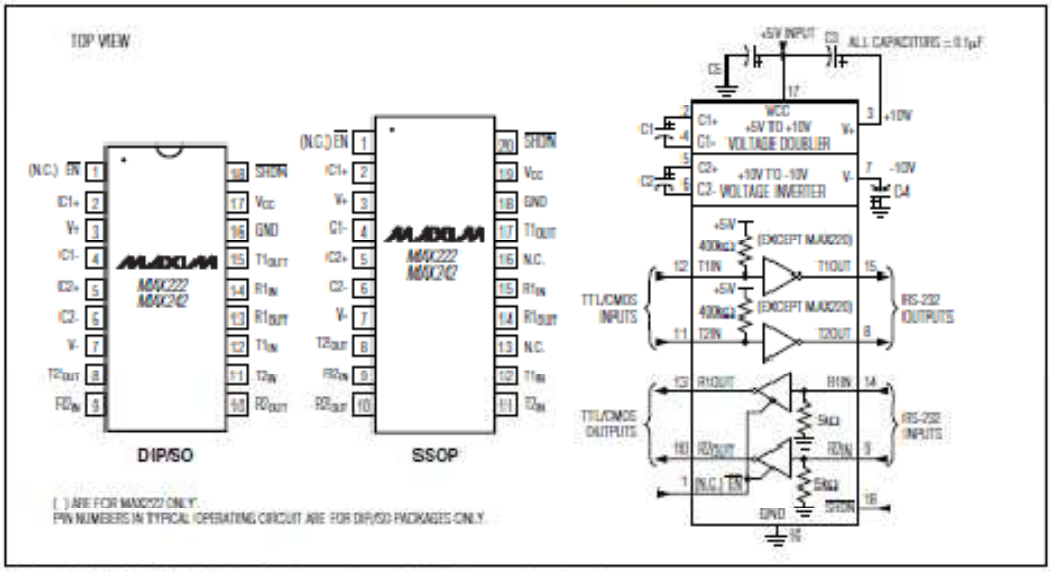


Figure 6. MAX222/MAX242 Pin Configurations and Typical Operating Circuit

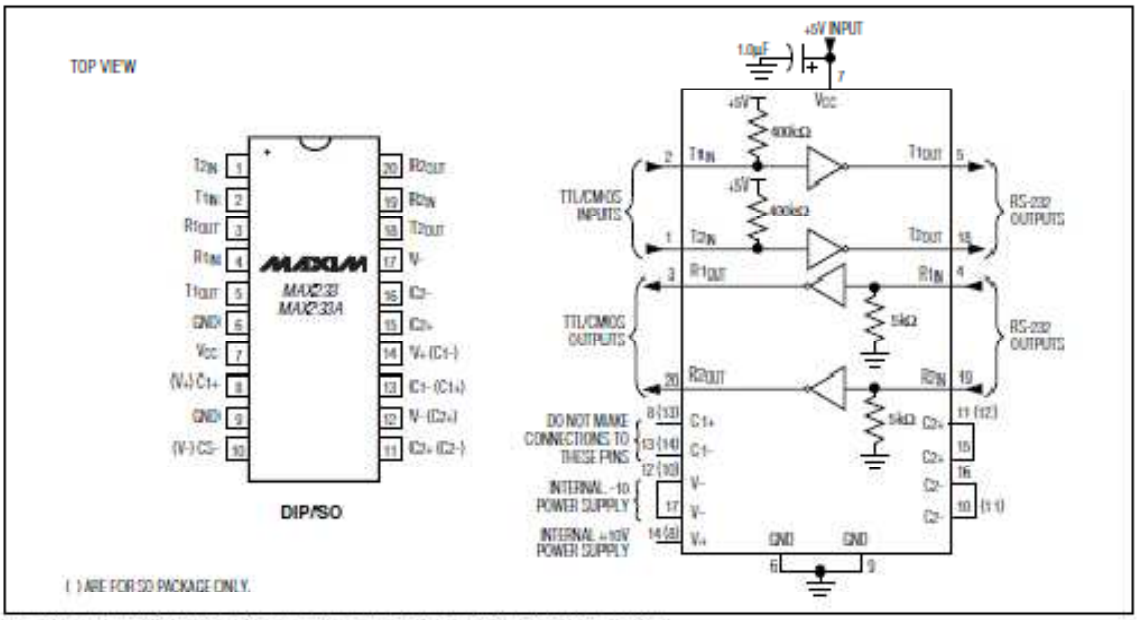


Figure 11. MAX233/MAX233A Pin Configuration and Typical Operating Circuit

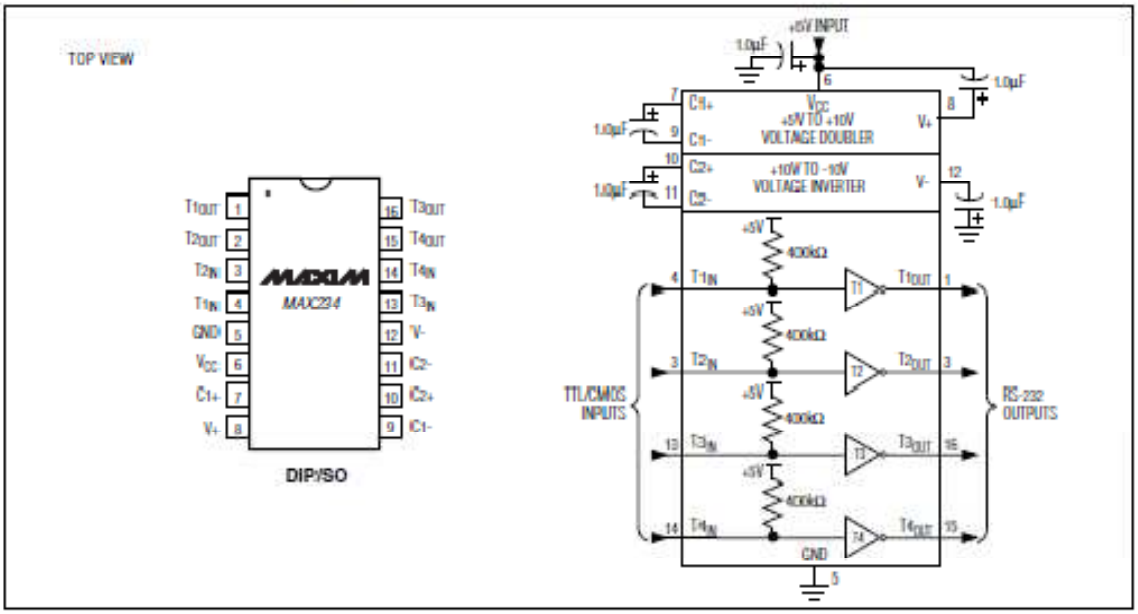


Figure 12. MAX234 Pin Configuration and Typical Operating Circuit

Anexo C. Código Fuente Celular

```
import processing.core.*; import processing.bluetooth.*; public class carro extends
PMIDlet{

//M\u00e1quina 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;
//Conexi\u00f3n serial
Client c;
//Fuente
PFont font;

//////////////////////////////////////////////////////////////////
public void setup()
{
  font=loadFont();
  textFont(font);

  //Inicializa objeto Bluetooth
  bt = new Bluetooth(this, Bluetooth.UUID_SERIALPORT);

  estado=ESTADO_INICIO;

}
//////////////////////////////////////////////////////////////////


public void destroy()
{
  bt.stop();
}

public void draw()
{
  background(0xFF, 0xFF, 0xFF); //RGB - blanco
```

```
  //Dibuja la pantalla dependiendo de qu\u00e9 estado tiene la m\u00e1quina
  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);
      textAlign(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,height-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].device.name+"\n"; //Hace una lista con nombres de
dispositivos hallados
        text(lista,2,2,width-4,height-4);
        }
      break;

    case ESTADO_CONECTADO:
      fill(0);
      textAlign(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.PNG");
     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 la b\u00fasqueda de dispositivos bluetooth
      estado=ESTADO_BUSCA;
      mens="Buscando Dispositivo Bluetooth...\n";
      break;

    case ESTADO_BUSCA:
      //Si ya hay al menos un dispositivo encontrado
      if(servicios!=null)
      {
        //Verifica que haya presionado algo del 0 al 9
        if((key>='0') && (key<='9'))

        {
          int i = key - '0'; //Conversi\u00f3n ASCII a entero
          //Si la tecla presionada existe como puerto disponible
          if(i < length(servicios))
          {
            mens="Conectando...";
            c = servicios[i].connect();
            estado= ESTADO_CONECTADO;

          }
        }
      }
      break;

    case ESTADO_CONECTADO:
      //Env\u00eda por el puerto serial la tecla que el usuario presion\u00f3
      c.write(key);
      c.flush();
      break;
  }
}
//////////////////////////////////////////////////////////////////


//Aqu\u00ed llega cuando alguna librer\u00eda genera un evento
```

```
public void libraryEvent(Object library, int event, Object data)
{
  //Si la librer\u00eda no fue bluetooth, sal de aqu\u00ed
  if(library!=bt)
    return;
  switch(event)
  {
    //Se encontr\u00f3 un dispositivo
    case Bluetooth.EVENT_DISCOVER_DEVICE:
      mens="Dispositivo: " + ((Device)data).address;
      break;

    //Ya se encontraron todos los dispositivos posibles
    case Bluetooth.EVENT_DISCOVER_DEVICE_COMPLETED:
      mens="Se encontraron " + length((Device[])data) + " dispositivos,\n" +
        "buscando puertos seriales...";
      break;

    //Encontr\u00f3 un puerto serial
    case Bluetooth.EVENT_DISCOVER_SERVICE:
      mens="Puerto serial en " + ((Service[])data)[0].device.address;
      break;

    //B\u00fasqueda de puertos seriales terminada
    case Bluetooth.EVENT_DISCOVER_SERVICE_COMPLETED:
      servicios=(Service[])data; //Junta todos los puertos encontrados en un arreglo
      mens="B\u00fasqueda terminada. Elija un puerto";
      break;

    //Ya se conect\u00f3 al cliente
    case Bluetooth.EVENT_CLIENT_CONNECTED:
      c=(Client)data;
      mens="Conexi\u00f3n exitosa :D";
      break;
  }
}

}
```

Anexo D. Código Fuente Microcontrolador

```
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:
'PRIMERA 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
 EndIF

      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
EndIF


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
EndIF

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
EndIF


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
EndIF


'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 PORTB.0
       Pause 300
       Low PORTB.0
       IF C!=0 Then
       C=C+1
       EndIF
      ' M=0
       GoTo INICIO
       EndIF

;**********************************
**********
      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
            High 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 300
            Low PORTB.0
            IF C!=0 Then
                    C=C+1
              EndIF
               '  M=0
              GoTo INICIO
      EndIF

      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=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!=0 Then
              C=C+1
              EndIF
        '  M=0
            GoTo INICIO

           EndIF

           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"
                    C=1
             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
           EndIF
       '  M=0
         GoTo INICIO
         EndIF

'FINAL PULSACIONES

GoTo INICIO




End
```

BIBLIOGRAFIA

LIBROS

- POLO, JESÚS. Arquitectura de microcontroladores, Documento Inédito. Diciembre 2008, pp 191-250
- ANGULO, J.Mª; EUGENIO, Martín y ANGULO Ignacio. Microcontroladores Pic: la solución en un chip. Madrid: Paraninfo, 1997. pp 231-290.
- RODRIGUEZ BUCARELLY, CARLOS. Visual Basic. Programación Orientada a objetos. México: MC GRAW-HILL, 2002. 110 p.
- DEITEL, HARVEY. Como Programar en Java. Traducido del ingles por Alfonso Vidal. 5ªed. MEXICO: PEARSON., 2004. pp 125-180
- BORCHES,PEDRO. Java2 Microedition: Soporte Bluetooth. Madrid: 2004. 57 p
- GALVEZ, SERGIO. J2ME España .Universidad Málaga España: CEAC, 1992.pp174-225

BIBLIOGRAFÍA DE INTERNET

Microcontrolador pic16f628a
http://www.pictronic.com,

(2009/03/10)

Java Bluetooth

http://wireless.java.sun.com/midp/articles/bluetooth2/

(2009/11/10)

Librerías en java

www.benhui.net

(2009/11/20)

Foros de Java
www.javahispano.org: Tu lenguaje, tu comunidad

(2009/07/25)

Instalación Netbeans

http--www_netbeans_org-kb-60-mobility-indexpantalla de inicio secion.htm

(2009/08/10)